

# 35th International Symposium on Computational Geometry

SoCG 2019, June 18–21, 2019, Portland, Oregon, USA

Edited by

Gill Barequet

Yusu Wang



*Editors*

**Gill Barequet**

Technion – Israel Inst. of Technology, Haifa, Israel  
barequet@cs.technion.ac.il

**Yusu Wang**

The Ohio State University, Ohio, USA  
yusu@cse.ohio-state.edu

*ACM Classification 2012*

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

**ISBN 978-3-95977-104-7**

*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/978-3-95977-104-7>.

*Publication date*

June, 2019

*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 3.0 Unported license (CC-BY 3.0):  
<https://creativecommons.org/licenses/by/3.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.SoCG.2019.0

ISBN 978-3-95977-104-7

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 (*Chair*, Gran Sasso Science Institute and Reykjavik University)
- Christel Baier (TU Dresden)
- Mikolaj Bojanczyk (University of Warsaw)
- Roberto Di Cosmo (INRIA and University Paris Diderot)
- Javier Esparza (TU München)
- Meena Mahajan (Institute of Mathematical Sciences)
- Dieter van Melkebeek (University of Wisconsin-Madison)
- Anca Muscholl (University Bordeaux)
- Luke Ong (University of Oxford)
- Catuscia Palamidessi (INRIA)
- Thomas Schwentick (TU Dortmund)
- Raimund Seidel (Saarland University and Schloss Dagstuhl – Leibniz-Zentrum für Informatik)

**ISSN 1868-8969**

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



## ■ Contents

Foreword	
<i>Gill Barequet and Yusu Wang</i> .....	0:xi
Conference Organization	
.....	0:xiii–0:xiv
Additional Reviewers	
.....	0:xv–0:xvi

### Invited Talk

A Geometric Data Structure from Neuroscience	
<i>Sanjoy Dasgupta</i> .....	1:1–1:1
Some Geometric and Computational Challenges Arising in Structural Molecular Biology	
<i>Bruce R. Donald</i> .....	2:1–2:2

### Regular Paper

A New Lower Bound for Semigroup Orthogonal Range Searching	
<i>Peyman Afshani</i> .....	3:1–3:14
Independent Range Sampling, Revisited Again	
<i>Peyman Afshani and Jeff M. Phillips</i> .....	4:1–4:13
An Efficient Algorithm for Generalized Polynomial Partitioning and Its Applications	
<i>Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl</i> .....	5:1–5:14
Efficient Algorithms for Geometric Partial Matching	
<i>Pankaj K. Agarwal, Hsien-Chih Chang, and Allen Xiao</i> .....	6:1–6:14
Connecting the Dots (with Minimum Crossings)	
<i>Akanksha Agrawal, Grzegorz Guśpiel, Jayakrishnan Madathil, Saket Saurabh, and Meirav Zehavi</i> .....	7:1–7:17
General Techniques for Approximate Incidences and Their Application to the Camera Posing Problem	
<i>Dror Aiger, Haim Kaplan, Efi Kokiopoulou, Micha Sharir, and Bernhard Zeisl</i> ...	8:1–8:14
Circumscribing Polygons and Polygonizations for Disjoint Line Segments	
<i>Hugo A. Akitaya, Matias Korman, Mikhail Rudoy, Diane L. Souvaine, and Csaba D. Tóth</i> .....	9:1–9:17
Morphing Contact Representations of Graphs	
<i>Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, and Vincenzo Roselli</i> .....	10:1–10:16
When Convexity Helps Collapsing Complexes	
<i>Dominique Attali, André Lieutier, and David Salinas</i> .....	11:1–11:15

35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang



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

Optimal Algorithm for Geodesic Farthest-Point Voronoi Diagrams <i>Luis Barba</i> .....	12:1–12:14
Upward Book Embeddings of st-Graphs <i>Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani</i> .....	13:1–13:22
Bounded Degree Conjecture Holds Precisely for $c$ -Crossing-Critical Graphs with $c \leq 12$ <i>Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera</i> .....	14:1–14:15
Preconditioning for the Geometric Transportation Problem <i>Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov</i> .....	15:1–15:14
The One-Way Communication Complexity of Dynamic Time Warping Distance <i>Vladimir Braverman, Moses Charikar, William Kuszmaul, David P. Woodruff, and Lin F. Yang</i> .....	16:1–16:15
Walking the Dog Fast in Practice: Algorithm Engineering of the Fréchet Distance <i>Karl Bringmann, Marvin Künnemann, and André Nusser</i> .....	17:1–17:21
Polyline Simplification has Cubic Complexity <i>Karl Bringmann and Bhaskar Ray Chaudhury</i> .....	18:1–18:16
A Spanner for the Day After <i>Kevin Buchin, Sariel Har-Peled, and Dániel Oláh</i> .....	19:1–19:15
Computing Shapley Values in the Plane <i>Sergio Cabello and Timothy M. Chan</i> .....	20:1–20:19
On the Metric Distortion of Embedding Persistence Diagrams into Separable Hilbert Spaces <i>Mathieu Carrière and Ulrich Bauer</i> .....	21:1–21:15
Convex Polygons in Cartesian Products <i>Jean-Lou De Carufel, Adrian Dumitrescu, Wouter Meulemans, Tim Ophelders, Claire Pennarun, Csaba D. Tóth, and Sander Verdonschot</i> .....	22:1–22:17
Smallest $k$ -Enclosing Rectangle Revisited <i>Timothy M. Chan and Sariel Har-Peled</i> .....	23:1–23:15
Dynamic Geometric Data Structures via Shallow Cuttings <i>Timothy M. Chan</i> .....	24:1–24:13
Lower Bounds for Electrical Reduction on Surfaces <i>Hsien-Chih Chang, Marcos Cossarini, and Jeff Erickson</i> .....	25:1–25:16
Maintaining the Union of Unit Discs Under Insertions with Near-Optimal Overhead <i>Pankaj K. Agarwal, Ravid Cohen, Dan Halperin, and Wolfgang Mulzer</i> .....	26:1–26:15
Almost Tight Lower Bounds for Hard Cutting Problems in Embedded Graphs <i>Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay</i> .....	27:1–27:16

The VC Dimension of Metric Balls Under Fréchet and Hausdorff Distances <i>Anne Driemel, Jeff M. Phillips, and Ioannis Psarros</i> .....	28:1–28:16
Dual Circumference and Collinear Sets <i>Vida Dujmović and Pat Morin</i> .....	29:1–29:17
A Product Inequality for Extreme Distances <i>Adrian Dumitrescu</i> .....	30:1–30:12
Topological Data Analysis in Information Space <i>Herbert Edelsbrunner, Žiga Virk, and Hubert Wagner</i> .....	31:1–31:14
Cubic Planar Graphs That Cannot Be Drawn On Few Lines <i>David Eppstein</i> .....	32:1–32:15
Counting Polygon Triangulations is Hard <i>David Eppstein</i> .....	33:1–33:17
Topologically Trivial Closed Walks in Directed Surface Graphs <i>Jeff Erickson and Yipu Wang</i> .....	34:1–34:17
Packing Disks into Disks with Optimal Worst-Case Density <i>Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer</i> .....	35:1–35:19
Semi-Algebraic Colorings of Complete Graphs <i>Jacob Fox, János Pach, and Andrew Suk</i> .....	36:1–36:12
Chunk Reduction for Multi-Parameter Persistent Homology <i>Ulderico Fugacci and Michael Kerber</i> .....	37:1–37:14
The Crossing Tverberg Theorem <i>Radoslav Fulek, Bernd Gärtner, Andrey Kupavskii, Pavel Valtr, and Uli Wagner</i> .	38:1–38:13
$\mathbb{Z}_2$ -Genus of Graphs and Minimum Rank of Partial Symmetric Matrices <i>Radoslav Fulek and Jan Kynčl</i> .....	39:1–39:16
An Experimental Study of Forbidden Patterns in Geometric Permutations by Combinatorial Lifting <i>Xavier Goaoc, Andreas Holmsen, and Cyril Nicaud</i> .....	40:1–40:16
Journey to the Center of the Point Set <i>Sariel Har-Peled and Mitchell Jones</i> .....	41:1–41:14
Preprocessing Ambiguous Imprecise Points <i>Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann</i> ...	42:1–42:16
Rods and Rings: Soft Subdivision Planner for $\mathbb{R}^3 \times S^2$ <i>Ching-Hsiang Hsu, Yi-Jen Chiang, and Chee Yap</i> .....	43:1–43:17
3-Manifold Triangulations with Small Treewidth <i>Kristóf Huszár and Jonathan Spreer</i> .....	44:1–44:20
Algorithms for Metric Learning via Contrastive Embeddings <i>Diego Ihara, Neshat Mohammadi, and Anastasios Sidiropoulos</i> .....	45:1–45:14
Exact Computation of the Matching Distance on 2-Parameter Persistence Modules <i>Michael Kerber, Michael Lesnick, and Steve Oudot</i> .....	46:1–46:15

Probabilistic Smallest Enclosing Ball in High Dimensions via Subgradient Sampling <i>Amer Krivošija and Alexander Munteanu</i> .....	47:1–47:14
A Weighted Approach to the Maximum Cardinality Bipartite Matching Problem with Applications in Geometric Settings <i>Nathaniel Lahn and Sharath Raghvendra</i> .....	48:1–48:13
The Unbearable Hardness of Unknotting <i>Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer</i> .....	49:1–49:19
On Grids in Point-Line Arrangements in the Plane <i>Mozhgan Mirzaei and Andrew Suk</i> .....	50:1–50:11
On Weak $\epsilon$ -Nets and the Radon Number <i>Shay Moran and Amir Yehudayoff</i> .....	51:1–51:14
Dynamic Planar Point Location in External Memory <i>J. Ian Munro and Yakov Nekrich</i> .....	52:1–52:15
Efficient Algorithms for Ortho-Radial Graph Drawing <i>Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf</i> .....	53:1–53:14
On the Chromatic Number of Disjointness Graphs of Curves <i>János Pach and István Tomon</i> .....	54:1–54:17
Computing Persistent Homology of Flag Complexes via Strong Collapses <i>Jean-Daniel Boissonnat and Siddharth Pritam</i> .....	55:1–55:15
Ham-Sandwich Cuts and Center Transversals in Subspaces <i>Patrick Schnider</i> .....	56:1–56:15
Distribution-Sensitive Bounds on Relative Approximations of Geometric Ranges <i>Yufei Tao and Yu Wang</i> .....	57:1–57:14
DTM-Based Filtrations <i>Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda</i> .....	58:1–58:15
A Divide-and-Conquer Algorithm for Two-Point $L_1$ Shortest Path Queries in Polygonal Domains <i>Haitao Wang</i> .....	59:1–59:14
Near-Optimal Algorithms for Shortest Paths in Weighted Unit-Disk Graphs <i>Haitao Wang and Jie Xue</i> .....	60:1–60:13
Searching for the Closest-Pair in a Query Translate <i>Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan</i> .....	61:1–61:15
On the Complexity of the $k$ -Level in Arrangements of Pseudoplanes <i>Micha Sharir and Chen Ziv</i> .....	62:1–62:15

**Multimedia Exposition**

Packing Geometric Objects with Optimal Worst-Case Density  
*Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer* ..... 63:1–63:6

Properties of Minimal-Perimeter Polyominoes  
*Gill Barequet and Gil Ben-Shachar* ..... 64:1–64:4

A Manual Comparison of Convex Hull Algorithms  
*Maarten Löffler* ..... 65:1–65:2

Fréchet View – A Tool for Exploring Fréchet Distance Algorithms  
*Peter Schäfer* ..... 66:1–66:5





## ■ Foreword

The 35th International Symposium on Computational Geometry (SoCG) was held in Portland, Oregon, United States, June 18-21, 2019, as part of the Computational Geometry Week. Altogether, 166 papers have been submitted to SoCG 2019. After a thorough review process, in which each paper has been evaluated by three or more independent reviewers, the Program Committee accepted 60 papers for presentation at SoCG. These proceedings contain extended abstracts of the accepted papers, limited to 500 lines plus references. If any supporting material (e.g., proofs or experimental details) does not fit in the line limit, the full paper is available at a public repository, which is referenced in the extended abstract.

The Best Paper Award goes to the paper “*Almost tight lower bounds for hard cutting problems in embedded graphs*” by Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay. The Best Student Presentation Award will be determined and announced at the symposium, based on ballots cast by the attendees.

A selection of papers, recommended by the Program Committee, have been invited to forthcoming special issues of *Discrete & Computational Geometry* and the *Journal of Computational Geometry*, dedicated to the best papers of the symposium.

In addition to the technical papers, there were four submissions to the multimedia exposition. All submissions were reviewed, and all four were accepted for presentation. One submission was split into two parts, hence, five multimedia segments were eventually accepted. The extended abstracts that describe these submissions are included in this proceedings volume. The multimedia content can be found at <http://www.computational-geometry.org>.

We thank the authors of all submitted papers and multimedia presentations. We are most grateful to the members of the SoCG Program Committee, the Multimedia Committee, and 275 additional reviewers for their dedication and expertise that ensured the high quality of the papers in these proceedings. We would also like to thank the Proceedings Chair, Matias Korman, for his meticulous work preparing the final proceedings. Many other people contributed to the success of SoCG 2019 and the entire CG Week. We especially thank the local organizers, all members of the Workshop and YRF Committees, and the Computational Geometry Steering Committee.

**Gill Barequet**

Program Committee, co-chair  
Technion—Israel Inst. of Technology

**Yusu Wang**

Program Committee, co-chair  
The Ohio State University

**Christiane Schmidt**

Multimedia Committee, chair  
Linköping University





## ■ Conference Organization

### SoCG Program Committee

- Hee-Kap Ahn, Pohang Univ. of Science and Technology, South Korea
- Alexandr Andoni, Columbia University, USA
- Sunil Arya, Hong Kong Univ. of Science and Technology, China
- Gill Barequet (co-chair), Technion—Israel Inst. of Technology, Israel
- Mark de Berg, TU Eindhoven, Netherlands
- Prosenjit Bose, Carleton University, Canada
- Frédéric Cazals, INRIA Sophia Antipolis-Méditerranée, France
- Tamal K. Dey, The Ohio State University, USA
- Kyle Fox, Univ. of Texas at Dallas, USA
- Joachim Gudmundsson, Univ. of Sydney, Australia
- Chaya Keller, Ben Gurion University, Israel
- Stephen Kobourov, Univ. of Arizona, USA
- Francis Lazarus, CNRS Grenoble, France
- Clément Maria, INRIA Sophia Antipolis-Méditerranée, France
- Tillmann Miltzow, Utrecht University, Netherlands
- Zuzana Patáková, Inst. of Science and Technology, Austria
- Amit Patel, Colorado State University, USA
- Raimund Seidel, Saarland University, Germany
- Christian Sohler, TU Dortmund, Germany, and Google, Switzerland
- Noam Solomon, Harvard University, USA
- Subhash Suri, Univ. of California at Santa Barbara, USA
- Kasturi Varadarajan, Univ. of Iowa, USA
- Birgit Vogtenhuber, Graz Univ. of Technology, Austria
- Bei Wang, Univ. of Utah, USA
- Yusu Wang (co-chair), The Ohio State University, USA

### SoCG Proceedings Chair

- Matias Korman, Tufts University, USA

### Multimedia Program Committee

- Aaron T. Becker, Univ. of Houston, USA
- Michael Biro, Univ. of Connecticut, USA
- Michael Hemmer, Google X Lab, USA
- Linda Kleist, TU Braunschweig, Germany
- Wolfgang Mulzer, FU Berlin, Germany
- Valentin Polishchuk, Linköping University, Sweden
- Christiane Schmidt (chair), Linköping University, Sweden
- Adam Sheffer, CUNY Baruch College, USA

35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang



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

**Workshop Program Committee**

- Maike Buchin, Technical University Dortmund, Germany
- Olivier Devillers (chair), Loria, France
- Ileana Streinu, Smith College, USA
- Alexander Wolff, Universität Würzburg, Germany

**Young Researchers Forum Program Committee**

- Kyle Fox, UT Dallas, USA
- Radoslav Fulek, IST Austria, Austria
- Jie Gao, Stony Brook University, USA
- Panos Giannopoulos, Middlesex University, United Kingdom
- Irina Kostitsyna, TU Eindhoven, Netherlands
- Steve Oudot (chair), INRIA, France
- Jeff Phillips, Univ. of Utah, USA
- Christian Sohler, TU Dortmund, Germany and Google Zürich, Switzerland
- Jonathan Spreer, Univ. of Sydney, Australia

**Local Organizing Committee**

- John Hershberger, Mentor Graphics, a Siemens Business, USA
- Bala Krishnamoorthy, Washington State University, USA
- Amir Nayyeri, Oregon State University, USA
- Gordon Wilfong, Nokia Bell Labs, USA
- William Maxwell (webmaster), Oregon State University, USA

**Steering Committee (2018-)**

- Mark de Berg, TU Eindhoven, Netherlands
- Erin Chambers (Secretary), Saint Louis University, USA
- Michael Hoffmann, ETH Zürich, Switzerland
- Joe Mitchell (Treasurer), State University of New York at Stony Brook, USA
- Bettina Speckmann, TU Eindhoven, Netherlands
- Monique Teillaud (Chair), INRIA Nancy - Grand Est, France

## ■ Additional Reviewers

Mikkel Abrahamsen	Chao Chen	Luisa Gargano
Michal Adamaszek	Siu-Wing Cheng	Mereke van Garderen
Henry Adams	Victor Chepoi	Alexey Glazyrin
Peyman Afshani	Augustin Chevallier	Marc Glisse
Oswin Aichholzer	Man-Kwun Chiu	Xavier Goaoc
José Luis Álvarez	Jongmin Choi	Omer Gold
Eric Andres	Aruni Choudhary	Douglas Gonçalves
Patrizio Angelini	Tobias Christiani	Lee-Ad Gottlieb
Boris Aronov	Josef Cibulka	Nicolas Grelier
Alan Arroyo	Vincent Cohen-Addad	Bernd Gärtner
Andrei Asinowski	David Cohen-Steiner	Masahiro Hachimori
Dominique Attali	Éric Colin de Verdière	Dan Halperin
Yakov Babichenko	David Conlon	Sariel Har-Peled
Jasine Babu	Radu Curticapean	Lenwood Heath
Sang Won Bae	Marco Cuturi	John Hershberger
Ainesh Bakshi	Ovidiu Daescu	Michael Hoffmann
Martin Balko	Sanjoy Dasgupta	Andreas Holmsen
Sayan Bandyopadhyay	Minati De	Ivor van der Hoog
Aritra Banik	Vincent Despré	Zengfeng Huang
Imre Barany	Olivier Devillers	Alfredo Hubard
Luis Barba	Emilio Di Giacomo	Clemens Huemer
Ulrich Bauer	Michael Dinitz	Kristóf Huszár
Gil Ben-Shachar	Gabor Domokos	John Iacono
Sergey Bereg	Anne Driemel	Tanmay Inamdar
Marcin Bienkowski	Guillaume Ducoffe	Ravi Janardan
Carla Binucci	Vida Dujmovic	Dominik Kaaser
Håvard Bakke Bjerkevik	Adrian Dumitrescu	Haim Kaplan
Thomas Bläsius	Christian Duncan	Matthew Katz
Jean-Daniel Boissonnat	Patrick Eades	Phillip Keldenich
Édouard Bonnet	Herbert Edelsbrunner	Nathan Keller
Karl Bringmann	Alon Efrat	Michael Kerber
Mickaël Buchet	Marek Eliáš	Jisu Kim
Kevin Buchin	Ioannis Emiris	Mincheol Kim
Johnatahn Bush	David Eppstein	Woojin Kim
Sergio Cabello	Jeff Erickson	Rolf Klein
Jean Cardinal	Esther Ezra	Linda Kleist
Paz Carmi	Ruy Fabila-Monroy	Fabian Klute
Jean-Lou De Carufel	Brittany Terese Fasy	Christian Konrad
Jérémie Chalopin	Sándor Fekete	Matias Korman
Erin Chambers	Marek Filakovský	Grigorios Koumoutsos
Hubert T.-H. Chan	Guilherme D. da Fonseca	Marc van Kreveld
Timothy M. Chan	Pierre Fraigniaud	Klaus Kriegel
Hsien-Chih Chang	Tobias Friedrich	Amer Krivosija
Yi-Jun Chang	Radoslav Fulek	Neeraj Kumar
Steven Chaplick	Jie Gao	Andrey Kupavskii

35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang



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

Jan Kynčl	Pavel Pataák	Luís Fernando Schultz Xavier
Marvin Künnemann	Aduri Pavan	da Silveira
Claudia Landi	Lehilton L. C. Pedrosa	Francesco Silvestri
Elmar Langetepe	Xavier Pennec	Primoz Skraba
Silvio Lattanzi	Gabriel Peyre	Michiel Smid
Hung Le	Jeff Phillips	Shakhar Smorodinsky
Erik Jan van Leeuwen	Vincent Pilaud	Pablo Soberón
David Letscher	Alexander Pilz	Jonathan Spreer
Joshua Levine	Rom Pinchasi	Frank Staals
Leo Liberti	Sylvain Pion	Anastasios Stefanou
Andre Lieutier	Valentin Polishchuk	Fabian Stehn
Chih-Hung Liu	Ioannis Psarros	Raphael Steiner
Roi Livni	Sergey Pupyrev	Andrew Suk
Daniel Lokshtanov	Dömötör Pálvölgyi	Narayanaswamy
Anna Lubiw	Miao Qiao	Sundararajan
Ben Lund	Kent Quanrud	Peter Synak
Maarten Löffler	Luis Rademacher	Martin Tancer
Sepideh Mahabadi	Sharath Raghvendra	Raphaël Tinarrage
Mehdi Makhul	Benjamin Raichel	Csaba Tóth
Dorian Mazauric	Rajiv Raman	Géza Tóth
Alex McCleary	Alexander Ravsky	Christopher Tralie
Saeed Mehrabi	Saurabh Ray	Katharine Turner
Piotr Micek	André van Renssen	Torsten Ueckerdt
Samuel Micka	Bruce Richter	Ryuhei Uehara
Matúš Mihalák	Oliver Roche-Newton	Seeun William Umboh
Joshua Mirth	Günter Rote	Pavel Valtr
Joseph Mitchell	Eva Rotenberg	Antoine Vigneron
Bojan Mohar	Natan Rubin	Magnus Wahlström
Debajyoti Mondal	Aviad Rubinstein	Erik Waingarten
Pat Morin	Paweł Rzażewski	Bartosz Walczak
Dmitriy Morozov	Eric Samperton	Haitao Wang
Guy Moshkovitz	Kanthi Sarpatwar	Osamu Watanabe
David Mount	Nadja Scharf	Rémi Watrigant
Wolfgang Mulzer	Anna Schenfisch	Oren Weimann
Elizabeth Munch	Lena Schlipf	Gordon Wilfong
Alexander Munteanu	Christiane Schmidt	Charles Wolf
Nabil Mustafa	Patrick Schnider	Sampson Wong
Abhinandan Nath	Hendrik Schrezenmaier	David Woodruff
Amir Nayyeri	Benjamin Schweinart	Ge Xia
Frank Nielsen	Chris Schwiegelshohn	Jinhui Xu
Bengt J. Nilsson	Eric Sedgwick	Ke Yi
Martin Nöllenburg	Micha Sharir	Yelena Yuditsky
Eunjin Oh	Don Sheehy	Norbert Zeh
Steve Oudot	Chan-Su Shin	Meirav Zehavi
Peter Palfrader	Aaron Sidford	Shira Zerbib
Irene Parada	Anastasios Sidiropoulos	Peilin Zhong
Nikos Parotsidis	Rodrigo Silveira	Sandra Zilles
Salman Parsa		

# A Geometric Data Structure from Neuroscience

Sanjoy Dasgupta

Department of Computer Science and Engineering,  
University of California San Diego, CA, USA  
<http://cseweb.ucsd.edu/~dasgupta/>  
dasgupta@eng.ucsd.edu

---

## Abstract

---

An intriguing geometric primitive, “expand-and-sparsify”, has been found in the olfactory system of the fly and several other organisms. It maps an input vector to a much higher-dimensional sparse representation, using a random linear transformation followed by winner-take-all thresholding.

I will show that this representation has a variety of formal properties, such as locality preservation, that make it an attractive data structure for algorithms and machine learning. In particular, mimicking the fly’s circuitry yields algorithms for similarity search and for novelty detection that have provable guarantees as well as having practical performance that is competitive with state-of-the-art methods.

This talk is based on work with Saket Navlakha (Salk Institute), Chuck Stevens (Salk Institute), and Chris Tosh (Columbia).

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms;  
Theory of computation → Data structures design and analysis

**Keywords and phrases** Geometric data structure, algorithm design, neuroscience

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.1

**Category** Invited Talk



© S. Dasgupta;

licensed under Creative Commons License CC-BY

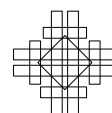
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 1; pp. 1:1–1:1

Leibniz International Proceedings in Informatics



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







# Some Geometric and Computational Challenges Arising in Structural Molecular Biology

Bruce R. Donald

Department of Computer Science, Department of Chemistry, and Department of Biochemistry  
Duke University and and Duke University Medical Center, Durham, NC, USA

[www.cs.duke.edu/brd/](http://www.cs.duke.edu/brd/)

[brd+socg19@cs.duke.edu](mailto:brd+socg19@cs.duke.edu)

---

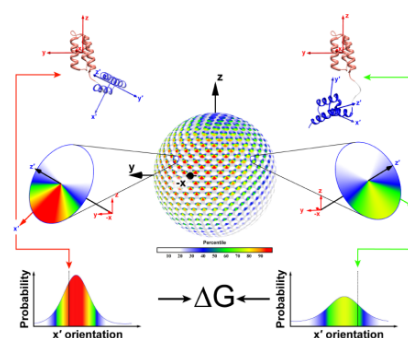
## Abstract

---

Computational protein design is a transformative field with exciting prospects for advancing both basic science and translational medical research. New algorithms blend discrete and continuous geometry to address the challenges of creating designer proteins. I will discuss recent progress in this area and some interesting open problems.

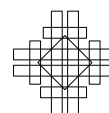
I will motivate this talk by discussing how, by using continuous geometric representations within a discrete optimization framework, broadly-neutralizing anti-HIV-1 antibodies were computationally designed that are now being tested in humans – the designed antibodies are currently in eight clinical trials, one of which is Phase 2a (NCT03721510). These continuous representations model the flexibility and dynamics of biological macromolecules, which are an important structural determinant of function.

However, reconstruction of biomolecular dynamics from experimental observables requires the determination of a conformational probability distribution. These distributions are not fully constrained by the limited geometric information from experiments, making the problem ill-posed in the sense of Hadamard. The ill-posed nature of the problem comes from the fact that it has no unique solution. Multiple or even an infinite number of solutions may exist. To avoid the ill-posed nature, the problem must be regularized by making (hopefully reasonable) assumptions.



I will present new ways to both represent and visualize correlated inter-domain protein motions (see the figure above). We use Bingham distributions, based on a quaternion fit to circular moments of a physics-based quadratic form. To find the optimal solution for the distribution, we designed an efficient, provable branch-and-bound algorithm that exploits the structure of analytical solutions to the trigonometric moment problem. Hence, continuous conformational PDFs can be determined directly from NMR measurements. The representation works especially well for multi-domain systems with broad conformational distributions. For more information please see Y. Qi et al. [1].

Ultimately, this method has parallels to other branches of geometric computing that balance discrete and continuous representations, including physical geometric algorithms, robotics, computational geometry, and robust optimization. I will advocate for using continuous distributions for protein modeling, and describe future work and open problems.



## 2:2 Geometric and Computational Challenges

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

**Keywords and phrases** Geometric computing, computational biology, Continuous Interdomain Orientation Distributions of Protein Conformations

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.2

**Category** Invited Talk

**Related Version** Y. Qi et al. [1]

---

### References

- 1 Yang Qi, Jeffrey W. Martin, Adam W. Barb, François Th  lot, Anthony K. Yan, Bruce R. Donald\*, and Terrence G. Oas\*. Continuous interdomain orientation distributions reveal components of binding thermodynamics. *Journal of Molecular Biology*, 430(18, Part B):3412 – 3426, 2018. doi:10.1016/j.jmb.2018.06.022.

---

\*Corresponding Authors

# A New Lower Bound for Semigroup Orthogonal Range Searching

Peyman Afshani

Aarhus University, Denmark

peyman@cs.au.dk

---

## Abstract

We report the first improvement in the space-time trade-off of lower bounds for the orthogonal range searching problem in the semigroup model, since Chazelle’s result from 1990. This is one of the very fundamental problems in range searching with a long history. Previously, Andrew Yao’s influential result had shown that the problem is already non-trivial in one dimension [14]: using  $m$  units of space, the query time  $Q(n)$  must be  $\Omega(\alpha(m, n) + \frac{n}{m-n+1})$  where  $\alpha(\cdot, \cdot)$  is the inverse Ackermann’s function, a very slowly growing function. In  $d$  dimensions, Bernard Chazelle [9] proved that the query time must be  $Q(n) = \Omega((\log_\beta n)^{d-1})$  where  $\beta = 2m/n$ . Chazelle’s lower bound is known to be tight for when space consumption is “high” i.e.,  $m = \Omega(n \log^{d+\epsilon} n)$ .

We have two main results. The first is a lower bound that shows Chazelle’s lower bound was not tight for “low space”: we prove that we must have  $mQ(n) = \Omega(n(\log n \log \log n)^{d-1})$ . Our lower bound does not close the gap to the existing data structures, however, our second result is that our analysis is tight. Thus, we believe the gap is in fact natural since lower bounds are proven for idempotent semigroups while the data structures are built for general semigroups and thus they cannot assume (and use) the properties of an idempotent semigroup. As a result, we believe to close the gap one must study lower bounds for non-idempotent semigroups or building data structures for idempotent semigroups. We develop significantly new ideas for both of our results that could be useful in pursuing either of these directions.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Randomness, geometry and discrete structures; Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Data Structures, Range Searching, Lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.3

**Related Version** A full version of the paper is available at [1], <https://arxiv.org/abs/1903.07967>.

**Funding** *Peyman Afshani*: supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Reserach under grant ID DFF-7014-00404.

## 1 Introduction

Orthogonal range searching in the semigroup model is one of the most fundamental data structure problems in computational geometry. In the problem, we are given an input set of points to store in a data structure where each point is associated with a weight from a semigroup  $\mathcal{G}$  and the goal is to compute the (semigroup) sum of all the weights inside an axis-aligned box given at the query time. Disallowing the “inverse” operation in  $\mathcal{G}$  makes the data structure very versatile as it is then applicable to a wide range of situations (from computing weighted sum to computing the maximum or minimum inside the query). In fact, the semigroup variant is the primary way the family of range searching problems are introduced, (see the survey [4]).

Here, we focus only on static data structures. We use the convention that  $Q(n)$ , the query time, refers to the worst-case number of semigroup additions required to produce the query answer  $S(n)$ , space, refers to the number of semigroup sums stored by the data structure. By storage, denoted by  $S^+(n)$ , we mean space but not counting the space used by the input,



© Peyman Afshani;

licensed under Creative Commons License CC-BY

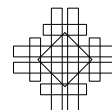
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 3; pp. 3:1–3:14

Leibniz International Proceedings in Informatics



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



i.e.,  $S(n) = n + S^+(n)$ . So we can talk about data structures with sublinear space, e.g., with 0 storage the data structure has to use the input weights only, leading to the worst-case query time of  $n$ .

### 1.1 The Previous Results

Orthogonal range searching is a fundamental problem with a very long history. The problem we study is also very interesting from a lower bound point of view where the goal is to understand the fundamental barriers and limitations of performing basic data structure operations. Such a lower bound approach was initiated by Fredman in early 80s and in a series of very influential papers (e.g., see [10, 11, 12]). Among his significant results, was the lower bound [11, 12] that showed a sequence of  $n$  insertions, deletions, and queries requires  $\Omega(n \log n)$  time to run.

Arguably, the most surprising result of these early efforts was given by Andrew Yao who in 1982 showed that even in one dimension, the static case of the problem contains a very non-trivial, albeit small, barrier. In one dimension, the problem essentially boils down to adding numbers: store an input array  $A$  of  $n$  numbers in a data structure s.t., we can add up the numbers from  $A[i]$  to  $A[j]$  for  $i$  and  $j$  given at the query time. The only restriction is that we should use only additions and not subtractions (otherwise, the problem is easily solved using prefix sums). Yao's significant result was that answering queries requires  $\Omega(\alpha(S(n), n) + n/S^+(n))$  additions, where  $\alpha(\cdot, \cdot)$  is the inverse Ackermann function. This bound implies that if one insists on using  $O(n)$  storage, the query bound cannot be reduced to constant, but even using a miniscule amount of extra storage (e.g., a  $\log^* \log^* n$  factor extra storage) can reduce the query bound to constant. Furthermore, using a bit less than  $n$  storage, e.g., by a  $\log^* \log^* n$  factor, will once again yield a more natural (and optimal) bound of  $n/S^+(n)$ . Despite its strangeness, it turns out there are data structures that can match the exact lower bound (see also [5]). After Tarjan's famous result on the union-find problem [13], this was the second independent appearance of the inverse Ackermann function in the history of algorithms and data structures.

Despite the previous attempts, the problem is still open even in two dimensions. At the moment, using range trees [7, 8] on the 1D structures is the only way to get two or higher dimensional results. In 2D for instance, we can have  $S^+(n) = O(n/\log n)$  with query bound  $Q(n) = O(\log^3 n)$ , or  $S^+(n) = O(n)$  with query bound  $Q(n) = O(\log^2 n)$ , or  $S^+(n) = O(n \log n)$  with query bound  $O(\alpha(cn, n) \log n)$ , for any constant  $c$ . In general and in  $d$  dimensions, we can build a structure with  $S^+(n) = O(n \log^{d-1} n)$  units of storage and with  $Q(n) = O(\alpha(cn, n) \log^{d-1} n)$  query bound, for any constant  $c$ . We can reduce the space complexity by any factor  $t$  by increasing the query bound by another factor  $t$ . Also, strangely, if  $t$  is asymptotically larger than  $\alpha(n, n)$ , then the inverse Ackermann term in the query bound disappears. Nonetheless, a surprising result of Chazelle [9] shows that the reverse is not true: the query bound must obey  $Q(n) = \Omega((\log_{S(n)/n} n)^{d-1})$  which implies using polylogarithmic extra storage only reduces the query bound by a  $(\log \log n)^{d-1}$  factor. Once again, using range tree with large fan out, one can build a data structure that uses  $O(n \log^{2d-2+\varepsilon} n)$  storage, for any positive constant  $\varepsilon$ , and achieves the query bound of  $O((\log_{\log n} n)^{d-1})$ . This, however leaves a very natural and important open problem: *Is Chazelle's lower bound the only barrier? Is it possible to achieve  $O(n)$  space and  $O(\log^{d-1} n)$  query time?*

**Idempotence and random point sets.** A semigroup is idempotent if for every  $x \in \mathcal{G}$ , we have  $x + x = x$ . All the previous lower bounds are in fact valid for idempotent semigroups. Furthermore, Chazelle's lower bound uses a uniform (or randomly placed) set of points which shows the lower bound does not require pathological or fragile input constructions.

Furthermore, his lower bound also holds for dominance ranges, i.e.,  $d$ -dimensional boxes in the form of  $(-\infty, a_1] \times \cdots \times (-\infty, a_d]$ . These little perks result in a very satisfying statement: problem is still difficult even when  $\mathcal{G}$  is “nice” (idempotent), and when the point set is “nice” (uniformly placed) and when the queries are simple (“dominance queries”).

## 1.2 Our Results

We show that for any data structure that uses  $S^+(n)$  storage and has query bound of  $Q(n)$ , we must have  $S^+(n) \cdot Q(n) = \Omega(n(\log n \log \log n)^{d-1})$ . This is the first improvement to the storage-time trade-off curve for the problem since Chazelle’s result in 1990. It also shows that Chazelle’s lower bound is not the only barrier. Observe that our lower bound is strong at a different corner of parameter space compared to Chazelle’s: ours is strongest when storage is small whereas Chazelle’s is strongest when the storage is large. Furthermore, we also keep most of the desirable properties of Chazelle’s lower bound: our lower bound also holds for idempotent semigroups and uniformly placed point sets. However, we have to consider more complicated queries than just dominance queries which ties to our second main result. We show that our analysis is tight: given a “uniformly placed” point set and an idempotent semigroup  $\mathcal{G}$ , we can construct a data structure that uses  $O(n)$  storage and has the query bound of  $O((\log n \log \log n)^{d-1})$ . As a corollary, we provide an almost complete understanding of orthogonal range searching queries with respect to a uniformly placed point set in an idempotent semigroup.

**Challenges.** Our results and specially our lower bound require significantly new ideas. To surpass Chazelle’s lower bound, we need to go beyond dominance queries which requires wrestling with complications that ideas such as range trees can introduce. Furthermore, in our case, the data structure can actually improve the query time by a factor  $f$  by spending a factor  $f$  extra space. This means, we are extremely sensitive to how the data structure can “use” its space. As a result, we need to capture the limits of how intelligently the data structure can spend its budget of “space” throughout various subproblems.

**Implications.** It is natural to conjecture that the uniformly randomly placed point set should be the most difficult point set for orthogonal queries. Because of this, we conjecture that our lower bounds are almost tight. This opens up a few very interesting open problems. See Section 5.

## 2 Preliminaries

**The Model of Computation.** Let  $P$  be an input set of  $n$  points with weights from a semigroup  $\mathcal{G}$ . Our model of computation is the same as the one used by the previous lower bounds, e.g., [9]. There has been quite some work dedicated to building a proper model for lower bounds in the semigroup model. We will not delve into those details and we only mention the final consequences of the efforts. The data structure stores a number of sums where each sum  $s$  is the sum of the weights of a subset  $s_P \subset P$ . With a slight abuse of the notation, we will use  $s$  to refer both to the sum as well as to the subset  $s_P$ . The number of stored sums is the space complexity of the data structure. If a sum contains only one point, then we call it a **singleton** and we use  $S^+(n)$  to denote the storage occupied by sums that are not singletons. Now, consider a query range  $r$  containing a subset  $r_P = r \cap P$ . The query algorithm must find  $k$  stored subsets  $s_1, \dots, s_k$  such that  $r_P = \cup_{i=1}^k s_i$ . For a given query  $r$ , the smallest such integer  $k$  is the query bound of the query. The query bound of the data

structure is the worst-case query bound of any query. Observe that the data structure does not disallow covering any point more than once and in fact, for idempotent semigroups this poses no problem. All the known lower bounds work in this way, i.e., they allow covering a point inside the query multiple times. However, if the semigroup is not idempotent, then covering a point more than once could lead to incorrect results. Since data structures work for general semigroups, they ensure that  $s_1, \dots, s_k$  are disjoint.

**Definitions and Notations.** A  $d$ -dimensional dominance query is determined by one point  $(x_1, \dots, x_d)$  and it is defined as  $(-\infty, x_1] \times \dots \times (-\infty, x_d]$ .

► **Definition 1.** We call a set  $P \subset \mathbb{R}^d$  **well-distributed** if the following properties hold: (i)  $P$  is contained in the  $d$ -dimensional unit cube. (ii) The volume of any rectangle that contains  $k \geq 2$  points of  $P$  is at least  $\varepsilon_d k / |P|$  for some constant  $\varepsilon_d$  that only depends on the dimension. (iii) Any rectangle that has volume  $v$ , contains at most  $\lceil v|P|/\varepsilon_d \rceil$  points of  $P$ .

► **Lemma 2** ([2, 9, 3]). For any constant  $d$  and any given value of  $n$ , there exists a well-distributed point set in  $\mathbb{R}^d$  containing  $\Theta(n)$  points.

### 3 The Lower Bound

This section is devoted to the proof of our main theorem which is the following.

► **Theorem 3.** If  $P$  is a well-distributed point set of  $n$  points in  $\mathbb{R}^d$ , any data structure that uses  $S^+(n)$  storage, and answers  $(2d - 1)$ -sided queries in  $Q(n)$  query bound requires that  $S^+(n) \cdot Q(n) = \Omega(n(\log n \log \log n)^{d-1})$ .

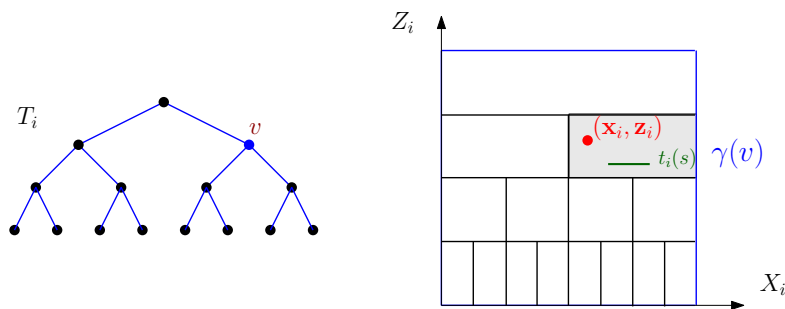
Let  $\mathcal{Q}$  be the unit cube in  $\mathbb{R}^d$ . Throughout this section, the input point set is a set  $P$  of  $n$  well-distributed points in  $\mathcal{Q}$ . Let  $\mathcal{D}$  be a data structure that answers semigroup orthogonal range searching queries on  $P$ .

#### 3.1 Definitions and Set up

We consider queries that have two boundaries in dimensions 1 to  $d - 1$  but only have an upper bound in dimension  $d$ . For simplicity, we rename the axes such that the  $d$ -th axis is denoted by  $Y$  and the first  $d - 1$  axes are denoted by  $X_1, \dots, X_{d-1}$ . Thus, each query is in the form of  $[x'_1, x_1] \times \dots \times [x'_{d-1}, x_{d-1}] \times (-\infty, y]$ . The point  $(x_1, \dots, x_{d-1}, y)$  is defined as the **dot of  $q$**  and is denoted by  $\text{Dot}(q)$ . For every  $1 \leq i \leq d - 1$ , the line segment that connects  $\text{Dot}(q)$  to the point  $(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_{d-1}, y)$  is called the  $i$ -th marker of  $q$  and it is denoted by  $t_i(s)$ .

**The tree  $T_i$ .** For each dimension  $i = 1, \dots, d - 1$ , we define a balanced binary tree  $T_i$  of height  $h = \log n$  as follows. Informally, we cut  $\mathcal{Q}$  into  $2^h$  congruent boxes with hyperplanes perpendicular to axis  $X_i$  which form the leaves of  $T_i$ . To be more specific, every node in  $T_i$  is assigned a box  $r(v) \subset \mathcal{Q}$ . The root of  $T_i$  is assumed to have depth 0 and it is assigned  $\mathcal{Q}$ . For every node  $v$ , we divide  $r(v)$  into two congruent “left” and “right” boxes with a hyperplane  $\ell(v)$ , perpendicular to  $X_i$  axis. The left box is assigned to left child of  $v$  and similarly the right box is assigned to the right child of  $v$ . We do not do this if  $r(v)$  has volume less than  $1/n$ ; these nodes become the leaves of  $T$ . Observe that all trees  $T_i$ ,  $1 \leq i \leq d - 1$  have the same height  $h$ . The volume of  $r(v)$  for a node  $v$  at depth  $j$  is  $2^{-j}$ .

**Embedding the problem in  $\mathbb{R}^{2d-1}$ .** The next idea is to embed our problem in  $\mathbb{R}^{2d-1}$ . Consistent with the previous notation, the first  $d$  axes are  $X_1, \dots, X_{d-1}$  and  $Y$ . We label the next axis  $Z_1, \dots, Z_{d-1}$ . We now represent  $T_i$  geometrically as follows. Consider  $h$  the height of  $T_i$ . For each  $i$ ,  $1 \leq i \leq d-1$ , we now define a **representative diagram**  $\Gamma_i$  which is a axis-aligned decomposition of the unit (planar) square  $Q_i$  in a coordinate system where the horizontal axis is  $X_i$  and the vertical axis is  $Z_i$ . As the first step of the decomposition, cut  $Q_i$  into  $h$  equal-sized sub-rectangles using  $h-1$  horizontal lines. Next, we will further divide each sub-rectangle into small regions and we will assign every node  $v$  of  $T_i$  to one of these regions. This is done as follows. The root  $v$  of  $T_i$  is assigned the topmost sub-rectangle as its region,  $\gamma(v)$ . Assume  $v$  is assigned a rectangle  $\gamma(v)$  as its region. We create a vertical cut starting from the middle point of the lower boundary of  $\gamma(v)$  all the way down to the bottom of the rectangle  $Q_i$ . The children of  $v$  are assigned to the two rectangles that lie immediately below  $\gamma(v)$ . See Figure 1.



■ **Figure 1** A tree and its representative diagram. The region of a node  $v$  is highlighted in grey.

**Placing the Sums.** Consider a semigroup sum  $s$  stored by the data structure  $\mathcal{D}$ . Our lower bound will also apply to semigroups that are idempotent which means without loss of generality, we can assume that our semigroup is idempotent. As a result, we can assume that each semigroup sum  $s$  stored by the data structure has the same shape as the query. Let  $b(s)$  be the smallest box that is unbounded from below (along the  $Y$  axis) that contains all the points of  $s$ . If  $s$  does not include a point,  $p$ , inside  $b(s)$ , we can just  $p$  to  $s$ . Any query that can use  $s$  must contain the box  $b(s)$  which means adding  $p$  to  $s$  can only improve things. Each sum  $s$  is placed in one node of  $T_i$  for every  $1 \leq i \leq d-1$ . The details of this placement are as follows.

A node  $v_i$  in  $T_i$  stores any sum  $s$  such that the  $i$ -th marker of  $s$ ,  $t_i(s)$ , intersects  $\ell(v_i)$  with  $v$  being the *highest* node with this property. Geometrically, this is equivalent to the following: we place  $s$  at a node  $v$  if  $\gamma(v)$  is the *lowest* region that fully contains the segment  $t_i(s)$  (or to be precise, the projection of  $t_i(s)$  onto the  $Z_i X_i$  plane). For example, in Figure 1(right), the sum  $s$  is placed at  $v$  in  $T_i$  since  $t_i(s)$ , the green line segment, is completely inside  $\gamma(v)$  with  $v$  being the lowest node of this property. Remember that  $s$  is placed at some node in each tree  $T_i$ ,  $1 \leq i \leq d-1$  (i.e., it is placed  $d-1$  times in total).

**Notations and difficult queries.** We will adopt the convention that random variables are denoted with bold math font. The difficult query is a  $2d-1$  sided query chosen randomly as follows. The query is defined as  $[\mathbf{x}'_1, \mathbf{x}_1] \times \dots \times [\mathbf{x}'_{d-1}, \mathbf{x}_{d-1}] \times (-\infty, \mathbf{y}]$  where  $\mathbf{x}'_i, \mathbf{x}_i$  and  $\mathbf{y}$  are also random variables (to be described).  $\mathbf{y}$  is chosen uniformly in  $[0, 1]$ . To choose the remaining coordinates, we do the following. We place a random point  $(\mathbf{x}_i, \mathbf{z}_i)$  uniformly

inside the representative plane  $\Gamma_i$  (i.e., choose  $\mathbf{x}_i$  and  $\mathbf{z}_i$  uniformly in  $[0, 1]$ ). Let  $\mathbf{v}_i$  be the random variable denoting the node in  $T_i$  s.t., region  $\gamma(\mathbf{v}_i)$  contains the point  $(\mathbf{x}_i, \mathbf{z}_i)$ .  $\mathbf{x}'_i$  is the  $X_i$ -coordinate of the left boundary of  $\gamma(\mathbf{v}_i)$ . Let  $\ell_i$  be the depth of  $\mathbf{v}_i$  in  $T_i$ . We denote the point  $(\mathbf{x}_1, \dots, \mathbf{x}_{d-1}, \mathbf{y})$  by  $\mathbf{q}$  and denote the  $2d - 1$  sided query by  $\text{Dom}_{\mathbf{v}_1, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$ . See Figure 1(right). Note that a query  $\text{Dom}_{v_1, \dots, v_{d-1}}(q)$  is equivalent to a dominance query defined by point  $q$  in  $r(v_1) \cap \dots \cap r(v_{d-1})$ . To simplify the presentation and to stop redefining these concepts, we will reserve the notations introduced in this paragraph to only represent the concepts introduced here.

► **Observation 4.** *A necessary condition for being able to use a sum  $s$  to answer  $\text{Dom}_{v_1, \dots, v_{d-1}}(q)$  is that  $s$  is stored at the subtree of  $v_i$ , for every  $1 \leq i \leq d - 1$ .*

**Proof.** Due to how we have placed the sums, the sums stored at the ancestors of  $v_i$  contain at least one point that lies outside  $r(v_i)$  and since  $\text{Dom}(q)$  is entirely contained inside  $r(v_i)$  those sums cannot be used to answer the query. ◀

**Subproblems.** Consider a query  $\text{Dom}(q) = \text{Dom}_{v_1, \dots, v_{d-1}}(q)$ . We now define subproblems of  $\text{Dom}(q)$ . A subproblem is represented by an array of  $d - 1$  integral indices  $\mathbf{j} = (j_1, \dots, j_{d-1})$  and it is denoted as  $\mathbf{j}$ -subproblem. The state of  $\mathbf{j}$ -subproblem of a query  $\text{Dom}_{v_1, \dots, v_{d-1}}(q)$  could either be *undefined*, or it could refer to covering a particular subset of points inside the query. In particular, given  $\text{Dom}(q)$ , a  $\mathbf{j}$ -subproblem is undefined if for some  $1 \leq i \leq d - 1$ , there is no node  $u_i \in T_i$  with the following properties:  $u_i$  has depth  $\ell_i + j_i$ ,  $u_i$  has a right sibling  $u'_i$  with  $r(u'_i)$  containing the query point  $q$ . See Figure 2. However, if such nodes  $u_i$  exist for all  $1 \leq i \leq d - 1$ , then the  $\mathbf{j}$ -subproblem of  $\text{Dom}(q)$  is **well-defined** and it refers to the problem of covering all the points inside the region  $\text{Dom}(q) \cap r(u_1) \cap \dots \cap r(u_{d-1})$ ; observe that this is equivalent to covering all the points inside the region  $r(u_1) \cap \dots \cap r(u_{d-1})$  that have  $Y$ -coordinate at most  $y$ . Further observe that for  $u_i$  to exist in  $T_i$ , it needs to pass two **checks**: (**check I**)  $\ell_i + j_i \leq h$  as otherwise, there are no nodes with depth  $\ell_i + j_i$  and (**check II**) a node  $u_i$  at depth  $\ell_i + j_i$  has a right sibling  $u'_i$  with  $r(u'_i)$  containing  $q$ . The nodes  $u_1, \dots, u_{d-1}$  are called the **defining nodes** of the  $\mathbf{j}$ -subproblem. Thus, the random variable  $\mathbf{v}_i$  defines the random variable  $\mathbf{u}_i$  where  $\mathbf{u}_i$  could be either undefined or it could be a node in  $T_i$ . Clearly, the distribution of  $\mathbf{u}_i$  is independent of the distributions of  $\mathbf{u}_j$  and  $\mathbf{v}_j$  for  $i \neq j$  as  $\mathbf{u}_i$  only depends on  $\mathbf{v}_i$ .

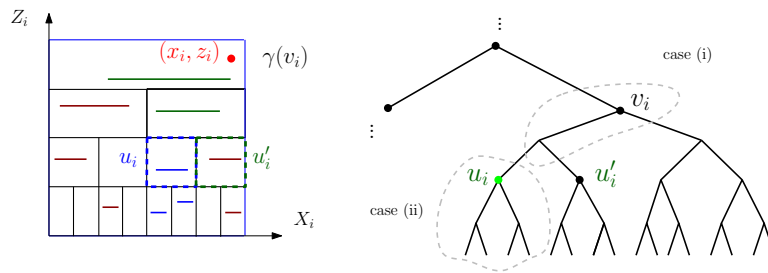
► **Observation 5.** *Consider a well-defined  $\mathbf{j} = (j_1, \dots, j_{d-1})$  subproblem of a query  $\text{Dom}_{v_1, \dots, v_{d-1}}(q)$  and its defining nodes  $u_1, \dots, u_{d-1}$ . To solve the  $\mathbf{j}$ -subproblem (i.e., to cover the points inside the subproblem), the data structure can use a sum  $s$  only if for every  $1 \leq i \leq d - 1$ , we either have case (i) where  $s$  is stored at ancestors of  $u_i$  but not the ancestors of  $v_i$  or case (ii) where  $s$  is stored at the subtree of  $u_i$ . If a sum  $s$  violates one of these two conditions for some  $i$ , then it cannot be used to answer the  $\mathbf{j}$ -subproblem. See Figure 2.*

**Proof.** See the full paper for the proof [1]. ◀

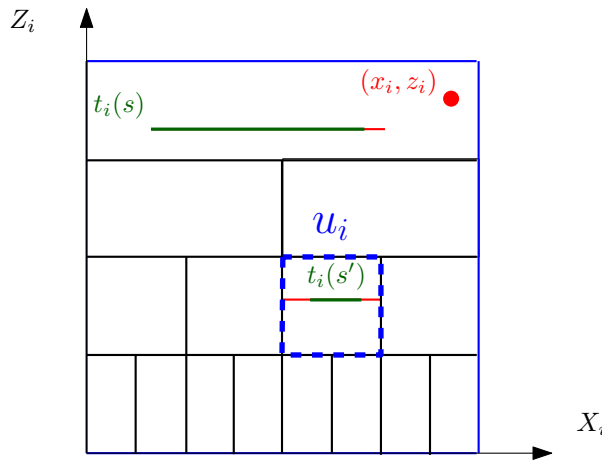
## 3.2 The Main Lemma

In this subsection, we prove a main lemma which is the heart of our lower bound proof. To describe this lemma, we first need the following notations. Consider a well-defined  $\mathbf{j}$ -subproblem of a query  $\text{Dom}_{v_1, \dots, v_{d-1}}(q)$  where  $j_i \leq \frac{h}{2}$  for  $1 \leq i \leq d - 1$ . As discussed, this subproblem corresponds to covering all the points in the region  $r(u_1) \cap \dots \cap r(u_{d-1})$  whose  $Y$ -coordinate is below  $y$ , the  $Y$ -coordinate of point  $q$ ; thus, the  $\mathbf{j}$ -subproblem of the query





■ **Figure 2**  $u_i$  is a defining node. The blue line segments correspond to  $t_i(s)$  of a sum  $s$  that is placed in the subtree of  $u_i$ . The green ones correspond to those placed at ancestors of  $u_i$  but not at ancestors of  $v_i$ . The red ones correspond to sums that cannot be used to answer the subproblem.



■ **Figure 3** The extensions of two sums that can be used to answer a subproblem.

can be represented as the problem of covering all the points inside the box  $[a_1, b_1] \times \dots \times [a_{d-1}, b_{d-1}] \times (-\infty, y]$  where  $a_i$  and  $b_i$  correspond to the left and the right boundaries of the slab  $r(u_i)$ . Let  $0 < \lambda$  be a parameter. Consider the region  $[a_1, b_1] \times \dots \times [a_{d-1}, b_{d-1}] \times [y - \beta, y]$  in which  $\beta$  is chosen such that the region contains  $\lambda$  points; as our pointset is well-distributed, this implies that the volume of the region is  $\Theta(\lambda/n)$ . We call this region **the  $\lambda$ -top box**. The  **$\lambda$ -top**, denoted by  $\text{Top}(j, \lambda)$ , is then the problem of covering all the points inside the  $\lambda$ -top box of the  $j$ -subproblem. With a slight abuse of the notation, we will use  $\text{Top}(j, \lambda)$  to refer also to the set of points inside the  $\lambda$ -top box. If there are not enough points in the  $\lambda$ -top box, the  $\lambda$ -top is undefined, otherwise, it is well-defined. These of course also depend on the query but we will not write the dependency on the query as it will clutter the notation. Furthermore, observe that when the query is random, then  $\text{Top}(j, \lambda)$  becomes a random variable which is either undefined or it is some subset of points.

**Extensions of sums.** Due to technical issues, we slightly extend the number of points each sum covers. Consider a sum  $s$  stored at a subtree of  $v_i$  such that  $s$  can be used to answer the  $j$ -subproblem. By Observation 4,  $s$  is either placed at the subtree of  $u_i$  or on the path connecting  $v_i$  to  $u_i$ . We extend the  $X_i$  range of the sum  $s$  (i.e., the projection of  $s$  on the  $X_i$ ) to include the left and the right boundary of the node  $u_i$  along the  $X_i$ -dimension. We do this for all  $d - 1$  first dimensions to obtain an extension  $e(s)$  of sum  $s$ . We allow the data structure to cover any point in  $e(s)$  using  $s$ .

► **Lemma 6** (The Main Lemma). *Consider a  $\mathbf{j} = (j_1, \dots, j_{d-1})$  subproblem of a random query  $\text{Dom}_{\mathbf{v}_1, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$ , for  $1 \leq j_i \leq h/2$ . Let  $\lambda = \frac{\delta h^{d-1}}{j_1 j_2 \dots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})}$  where  $\delta$  is a small enough constant and  $S^+(\mathcal{A})$  is the storage of the data structure. Let  $\mathbf{S}_j$  be the set of sums  $s$  such that (i)  $s$  is contained inside the query  $\text{Dom}_{\mathbf{v}_1, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$ , and (ii)  $e(s)$  covers at least  $C$  points from  $\text{Top}(\mathbf{j}, \lambda)$ , meaning,  $|e(s) \cap \text{Top}(\mathbf{j}, \lambda)| \geq C$  where  $C$  is a large enough constant.*

*With  $\Omega(1)$  probability, the  $\mathbf{j}$ -subproblem and the  $\text{Top}(\mathbf{j}, \lambda)$  are well-defined. Furthermore conditioned on both of these being well-defined, with probability  $1 - O(\sqrt{\delta/\varepsilon_d})$ , the nodes  $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$  will be sampled as nodes  $v_1, \dots, v_{d-1}$ , s.t., the following holds:  $\mathbb{E}[|\sum_{s \in \mathbf{S}_j} |e(s) \cap \text{Top}(\mathbf{j}, \lambda)|] < \frac{|\text{Top}(\mathbf{j}, \lambda)|}{C'}$  where the expectation is over the random choices of  $\mathbf{y}$  and  $C'$  is another large constant.*

Let us give some intuition on what this lemma says and why it is critical for our lower bound. For simplicity assume  $S^+(\mathcal{A}) = n$  and assume we sample  $\mathbf{v}_1, \dots, \mathbf{v}_{d-1}$  as the first step, and then sample  $\mathbf{y}$  as the last step. The above lemma implies that if we focus on one particular subproblem, the sums in the data structure cannot cover too many points; to see this consider the following. The lemma first says that after the first step, with positive constant probability,  $\mathbf{j}$ -subproblem and  $\text{Top}(\mathbf{j}, \lambda)$  are well-defined. Furthermore, here is a very high chance that our random choices will “lock us” in a “doomed” state, after sampling  $v_1, \dots, v_{d-1}$ . Then, when considering the random choices of  $\mathbf{y}$ , sums that cover at least  $C$  points in total cover a very small fraction of the points. As a result, we will need  $\Omega(\lambda/C) = \Omega(\frac{\delta \log^{d-1} n}{C j_1 j_2 \dots j_{d-1}})$  sums to cover the points inside the  $\lambda$ -top of the subproblem. Summing these values over all possible subproblems,  $j_i, 1 \leq j_i \leq h/2, 1 \leq i \leq d-1$  will create a lot of Harmonic sums of the type  $\sum_{x=1}^{h/2} x = O(\log \log n)$  which will eventually lead to our lower bound. In particular, we will have  $\sum_{j_i, 1 \leq j_i \leq h/2} \Omega(\frac{h^{d-1}}{j_1 j_2 \dots j_{d-1}}) = (\log n \log \log n)^{d-1}$ . There is however, one very big technical issue that we will deal with later: a sum can cover very few points from each subproblem but from very many subproblems! Without solving this technical issue, we only get the bound  $\max_{j_i, 1 \leq j_i \leq h/2} \Omega(\frac{h^{d-1}}{j_1 j_2 \dots j_{d-1}}) = (\log n)^{d-1}$  which offers no improvements over Chazelle’s lower bound. Thus, while solving this technical issue is important, nonetheless, it is clear that the lemma we will prove in this section is also very critical.

As this subsection is devoted to the proof of the above lemma, we will assume that we are considering a fixed  $\mathbf{j}$ -subproblem and thus the indices  $j_1, \dots, j_{d-1}$  are fixed.

### 3.2.1 Notation and Setup

By Observation 5, only a particular set of sums can be used to answer the  $\mathbf{j}$ -subproblem of a query. Consider a sum  $s$  that can be used to answer the subproblem of some query. By the observation, we must have that  $s$  must either satisfy case (i) or case (ii) for every tree  $T_i, 1 \leq i \leq d-1$ . Over all indices  $i, 1 \leq i \leq d-1$ , they describe  $2^{d-1} = O(1)$  different cases. This means that we can partition  $\mathbf{S}_j$  into  $2^{d-1}$  different **equivalent classes** s.t., for any two sums  $s_1$  and  $s_2$  in an equivalent class, either they both satisfy case (i) or they both satisfy case (ii) in Observation 5 and for any dimension  $i$ . Since  $2^{d-1}$  is a constant, it suffices to show that our lemma holds when only considering sums of particular equivalent class. In particular, let  $S'_j$  be the subset of eligible sums that all belong to one equivalent class. Now, it suffices to show that  $\mathbb{E}[|\sum_{s \in S'_j} |e(s) \cap \text{Top}(\mathbf{j}, \lambda)|] < \frac{|\text{Top}(\mathbf{j}, \lambda)|}{2^d C'}$ . since summing these over all  $2^{d-1}$  equivalent classes will yield the lemma. Furthermore, w.l.o.g and by renaming the  $X$ -axes, we can assume that there exists a fixed value  $t, 0 \leq t \leq d-1$ , such that for every sum  $s \in S'_j$ , for dimensions  $1 \leq i \leq t$ ,  $s$  satisfies case (i) in  $T_i$  and for  $t < i \leq d-1$ ,  $s$  is within case (ii). Note that if  $t = 0$ , then it implies that we have no instances of case (i) and for  $t = d-1$  we have no instances of case (ii).

**The probability distribution of subproblems.** To proceed, we need to understand the distribution of the subproblems. This is done by the following observation.

► **Observation 7.** Consider a  $j = (j_1, \dots, j_{d-1})$  subproblem of a random query  $\text{Dom}_{\mathbf{v}_1, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$  defined by random variables  $\mathbf{u}_1, \dots, \mathbf{u}_{d-1}$ . We can make the following observations. (i) the distribution of the random variable  $j_i + \ell_i$  is uniform among the integers  $j_i + 1, \dots, h + j_i$ . (ii) With probability  $j_i/h$ ,  $\mathbf{u}_i$  will be undefined because it fails (Check I). (iii) If (Check I) does not fail for  $\mathbf{u}_i$ , there is exactly 0.5 probability that  $\mathbf{u}_i$  is undefined. (iv) For a fixed  $j_i$ , the probability distribution,  $\mu_i$ , of  $\mathbf{u}_i$  is as follows: with probability  $1 - \frac{1-j_i/h}{2}$ ,  $\mathbf{u}_i$  is undefined. Otherwise,  $\mathbf{u}_i$  is a node in  $T_i$  sampled in the following way: sample a random integer (depth)  $\ell'$  uniformly among integers in  $j_i + 1, \dots, h$  and select a random node uniformly among all the nodes at depth  $\ell'$  that have a right sibling.

**Proof.** See the full paper for the proof [1]. ◀

**Partial Queries.** Observe that w.l.o.g., we can assume that we first generate the dimensions 1 to  $t$  of the query, and then the dimensions  $t + 1$  to  $d - 1$  of the query, and then the value  $\mathbf{y}$ . A partial query is one where only the dimensions 1 to  $t$  have been generated. This is equivalent to only sampling  $t$  random points  $(\mathbf{x}_i, \mathbf{z}_i)$  for  $1 \leq i \leq t$ . To be more specific, assume we have set  $\mathbf{v}_i = v_i$ , for  $1 \leq i \leq t$  where each  $v_i$  is a node in  $T_i$ . Then, the partial query is equivalent to the random query  $\text{Dom}_{v_1, \dots, v_t, \mathbf{v}_{t+1}, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$  and in which the first  $t$  coordinates of  $\mathbf{q}$  are known (not random). Thus, we can still talk about the  $j$ -subproblem of a partial query; it could be that the  $j$ -subproblem is already known to be undefined (this happens when one of the nodes  $u_i$ ,  $1 \leq i \leq t$  is known to be undefined) but otherwise, it is defined by defining nodes  $u_1, \dots, u_t$  and the random variables  $\mathbf{u}_{t+1}, \dots, \mathbf{u}_{d-1}$ ; these latter random variables could later turn out to be undefined and thus rendering the  $j$ -subproblem of the query undefined.

After sampling a partial query, we can then talk about **eligible** sums: a sum  $s$  is eligible if it could potentially be used to answer the  $j$ -subproblem once the full query has been generated. Note that the emphasis is on answering the  $j$ -subproblem. This means, there are multiple ways for a sum to be ineligible: if  $j$ -subproblem is already known to be undefined then there are no eligible sums. Otherwise, the defining nodes  $u_1, \dots, u_t$  are well-defined. In this case, if it is already known that  $s$  is outside the query, or it is already known that  $s$  cannot cover any points from the  $j$ -subproblem then  $s$  becomes ineligible. Final and the most important case of ineligibility is when  $s$  is placed at a node  $w_i$  which is a descendant of node  $u_i \in T_i$  for some  $1 \leq i \leq t$ . If this happens, even though  $s$  can be potentially used to answer the  $j$ -subproblem, it can do so from a different equivalent class, as the reader should remember that we only consider sums that are stored in the path that connects  $u_i$  to  $v_i$  for  $1 \leq i \leq t$ . If a sum passes all these, then it is eligible. Clearly, once the final query is generated, the set  $S'_j$  is going to be a subset of the eligible sums.

► **Definition 8.** Given a partial query  $\text{Dom}_{v_1, \dots, v_t, \mathbf{v}_{t+1}, \dots, \mathbf{v}_{d-1}}(\mathbf{q})$ , and considering a fixed  $j$ -subproblem, we define the potential function  $\Phi_{v_1, \dots, v_t}$  to be the number of eligible sums.

► **Lemma 9.** We have

$$\mathbb{E}(\Phi_{\mathbf{v}_1, \dots, \mathbf{v}_t} \cdot \prod_{i=1}^t \frac{h2^{\ell_i + j_i}}{j_i}) \leq O(S^+(\mathcal{D})).$$

**Proof.** See the full paper for the proof [1]. ◀

### 3:10 A New Lower Bound for Semigroup Orthogonal Range Searching

By the above lemma, we except only few eligible sums for a random partial query. Let  $\mathcal{B}\mathcal{A}\mathcal{D}_1$  be the “bad” event that the nodes  $\mathbf{v}_1, \dots, \mathbf{v}_t$  are sampled to be nodes  $v_1, \dots, v_t$  such that  $\Phi_{v_1, \dots, v_t} \cdot \prod_{i=1}^t \frac{h2^{\ell_i + j_i}}{j_i} > |S^+(\mathcal{D})|/\varepsilon$ . By Markov’s inequality and Lemma 9,  $\Pr[\mathcal{B}\mathcal{A}\mathcal{D}_1] = O(\varepsilon)$ .

Now, fix  $\mathbf{v}_1 = v_1, \dots, \mathbf{v}_t = v_t$ . In the rest of the proof we will assume these values are fixed and we are going to generate the rest of the query. Next, we define another potential function.

► **Definition 10.** *The potential  $\Psi_{w_{t+1}, \dots, w_{d-1}}$  for  $w_{t+1} \in T_{t+1}, \dots, w_{d-1} \in T_{d-1}$ , where the depth of  $w_i$  in  $T_i$  is  $d_i$  is defined as follows. First define  $\#_{x_{t+1}, \dots, x_{d-1}}$  for nodes  $x_i \in T_i$  to be the number of eligible sums  $s$  such that  $s$  is placed at  $x_i$  for  $t+1 \leq i \leq d-1$ . Given the nodes  $w_{t+1}, \dots, w_{d-1}$ , and for non-negative integers  $k_{t+1}, \dots, k_{d-1}$ , we define  $\#_{k_{t+1}, \dots, k_{d-1}}$  as the sum of all  $\#_{w'_{t+1}, \dots, w'_{d-1}}$  over all nodes  $w'_i$  where  $w'_i$  has depth  $d_i + k_i$  in  $T_i$  and  $w'_i$  is a descendant of  $w_i$ . We define the potential function as follows.*

$$\Psi_{w_{t+1}, \dots, w_{d-1}} = \sum_{k_{t+1}=0}^{\infty} \dots \sum_{k_{d-1}=0}^{\infty} \frac{\#_{k_{t+1}, \dots, k_{d-1}}}{2^{k_{t+1} + \dots + k_{d-1}}}.$$

► **Lemma 11.** *Having fixed the nodes  $v_1, \dots, v_t$ , we have,*

$$\mathbb{E}[\Psi_{\mathbf{u}_{t+1}, \dots, \mathbf{u}_{d-1}} \cdot \prod_{i=t+1}^{d-1} (h2^{\ell_i + j_i})] = O(\Phi_{v_1, \dots, v_t})$$

where  $\ell_i$  is the depth of  $\mathbf{v}_i$ ,  $\mathbf{u}_i$  is the defining node of the  $j$ -subproblem, the expectation is taken over the random choices of  $\mathbf{v}_i$ ,  $t+1 \leq i \leq d-1$  and the potential is defined to be zero if any of the nodes  $\mathbf{u}_i$  is undefined.

**Proof.** See the full paper for the proof [1]. ◀

Now we define the second bad event  $\mathcal{B}\mathcal{A}\mathcal{D}_2$  to be the event that  $\Psi_{u_{t+1}, \dots, u_{d-1}} \cdot (h2^{\ell_{t+1} + j_{t+1}}) \dots (h2^{\ell_{d-1} + j_{d-1}}) \geq \Phi_{v_1, \dots, v_t}/\varepsilon$ . By Markov’s inequality and Lemma 11,  $\Pr[\mathcal{B}\mathcal{A}\mathcal{D}_2] = O(\varepsilon)$ .

**Proof of the main lemma.** We now prove our main lemma (Lemma 6 at page 7).

Remember that we will focus on one equivalent class  $\mathbf{S}'_j$  of  $\mathbf{S}_j$ . Observe that the summation  $\sum_{s \in \mathbf{S}'_j} |e(s) \cap \text{Top}(j, \lambda)|$  counts how many times a point in  $\text{Top}(j, \lambda)$  is covered by extensions of sums that cover at least  $C$  points of the  $\text{Top}(j, \lambda)$  and this only takes into account the random choices of  $\mathbf{y}$  as the nodes  $v_1, \dots, v_{d-1}$  have been fixed. As a result,  $\mathbf{S}'_j$  is a random variable that only depends on  $\mathbf{y}$ . To make this clear, let  $\mathcal{M}_j$  be the set that includes all the sums that can be part of  $\mathbf{S}'_j$  over all the random choices of  $\mathbf{y}$ . As a result,  $\mathbf{S}'_j$  is a random subset of  $\mathcal{M}_j$ . Observe that every sum  $s \in \mathcal{M}_j$  has the property that it is stored in some node on the path from  $u_i$  to  $v_i$  for  $1 \leq i \leq t$  and at the subtree of  $u_i$  for  $t+1 \leq i \leq d-1$ . Since  $\text{Top}(j, \lambda)$  has exactly,  $\lambda$  points, we can label them from one to  $\lambda$  under some global ordering of the points (e.g., lexicographical ordering). Thus, let  $f(g)$  be the  $x$ -th point in  $\text{Top}(j, \lambda)$ ,  $1 \leq g \leq \lambda$ . Also, let  $m(g)$  be the number of sums  $s \in \mathbf{S}'_j$  s.t.,  $e(s)$  contains  $f(g)$ . Then, we can do the following rewriting:

$$\sum_{s \in \mathbf{S}'_j} |e(s) \cap \text{Top}(j, \lambda)| = \sum_{g=1}^{\lambda} m(g).$$

By linearity of expectation,

$$\mathbb{E}\left[\sum_{s \in \mathbf{S}'_j} |e(s) \cap \text{Top}(j, \lambda)|\right] = \sum_{g=1}^{\lambda} \mathbb{E}[m(g)] = \sum_{s \in \mathcal{M}_j} \sum_{g=1}^{\lambda} \Pr[s \text{ covers } f(g), s \in \mathbf{S}'_j]. \quad (1)$$

In the rest of the proof, we bound the right hand side of Eq. 1 and note that the probability is over the choices of  $\mathbf{y}$ . Consider a particular outcome of our random trials in which the random variable  $\mathbf{v}_i$  has been set to node  $v_i$ , for  $1 \leq i \leq d - 1$  in which none of the bad events  $\mathcal{B}\mathcal{W}_1$  and  $\mathcal{B}\mathcal{W}_2$  have happened. Set the parameter  $\varepsilon$  used in the definition of these bad events to  $\varepsilon = \sqrt{\delta/\varepsilon_d}$ . Thus, none of the bad events happen with probability at least  $1 - O(\sqrt{\delta/\varepsilon_d})$ , conditioned on the event that the  $j$ -subproblem of the query is defined. Note that can we assume the random variable  $\mathbf{y}$  has not been assigned yet. This is a valid assumption since the subproblem of a query only depend on the selection of the nodes  $v_1, \dots, v_{d-1}$  and not on the  $Y$ -coordinate of the query.

As  $\mathcal{B}\mathcal{W}_1$  has not occurred, we have  $\Phi_{v_1, \dots, v_t} \cdot \prod_{i=1}^t \frac{h2^{\ell_i+j_i}}{j_i} \leq |S(\mathcal{D})/\varepsilon|$ . As  $\mathcal{B}\mathcal{W}_2$  has not occurred either, we know that  $\Psi_{u_{t+1}, \dots, u_{d-1}} \cdot \prod_{i=t+1}^{d-1} (h2^{\ell_i+j_i}) < \Phi_{v_1, \dots, v_t}/\varepsilon$ . Thus,

$$\begin{aligned} \Psi_{v_{t+1}, \dots, v_{d-1}} &< \frac{\Phi_{v_1, \dots, v_t}}{\varepsilon \prod_{i=t+1}^{d-1} (h2^{\ell_i+j_i})} \leq \frac{|S^+(\mathcal{D})|}{\varepsilon \prod_{i=1}^t \frac{h2^{\ell_i+j_i}}{j_i}} \cdot \frac{1}{\varepsilon \prod_{i=t+1}^{d-1} (h2^{\ell_i+j_i})} \\ &= \frac{|S^+(\mathcal{D})| \prod_{i=1}^t j_i}{\varepsilon^2 h^{d-1} \prod_{i=1}^{d-1} 2^{j_i+\ell_i}}. \end{aligned} \tag{2}$$

**The experiment.** To bound the sum at the Eq. 1, we will use the above inequality combined with the following experiment. We select a random point  $\mathbf{p}$  from  $\text{Top}(j, \lambda)$  by sampling an integer  $\mathbf{g} \in [1, \dots, \lambda]$  and considering  $f(\mathbf{g})$ . We compute the probability that  $f(\mathbf{g})$  can be covered by the extension of a sum in  $\mathbf{S}'_j$  where the probability is computed over the choices of  $\mathbf{g}$  and the  $Y$ -coordinate of the query  $\mathbf{y}$ .

We now look at the side lengths of the box  $\text{Top}(j, \lambda)$ . The  $i$ -th side length of  $\lambda$ -top box is  $\frac{1}{2^{\ell_i+j_i}}$  for  $1 \leq i \leq d - 1$ ; this is because the  $j$ -subproblem was defined by nodes  $u_i$  where  $u_i$  has depth  $\ell_i + j_i$ . Let  $\beta$  be the side length of  $\text{Top}(j, \lambda)$  along the  $Y$ -axis. As  $\beta$  is chosen such that  $\text{Top}(j, \lambda)$  contains  $\lambda$  points and the pointset well distributed, the volume of  $\lambda$ -top box is  $\Theta(\lambda/n)$ . This implies, it suffices to pick  $\beta = \Theta(\frac{\lambda}{n} \prod_{i=1}^{d-1} 2^{\ell_i+j_i})$ . Now remember that the  $Y$ -coordinate of the top boundary of the  $\lambda$ -top box is  $y$  and the  $Y$ -coordinate of its lower boundary is  $y - \beta$ .

Consider a sum  $s \in \mathcal{M}_j$ . Now consider the smallest box enclosing  $e(s)$ ; w.l.o.g., we use the notation  $e(s)$  to refer to this box. For  $t + 1 \leq i \leq d - 1$ , the  $i$ -th side length of  $e(s)$  is  $2^{-\ell_i-j_i-\zeta_i(s)}$  because  $s$  was placed at node  $w_i \in T_i$  which is below  $u_i$  and thus our extensions extends the  $i$ -dimension of the box to match that of  $w_i$ . However, for  $1 \leq i \leq t$ , the  $i$ -th side length of  $e(s)$  is  $2^{-\ell_i-j_i}$ . We have

$$\text{Vol}(e(s) \cap \text{Top}(j, \lambda)) \leq \beta \prod_{i=1}^t 2^{-\ell_i-j_i} \prod_{i=t+1}^{d-1} 2^{-\ell_i-j_i-\zeta_i(s)} = \Theta\left(\frac{\lambda}{n} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}\right). \tag{3}$$

Observe that we have assumed  $s$  covers at least  $C$  points inside  $\text{Top}(j, \lambda)$ . However, our point set is well-distributed which implies the number of points covered by  $s$  is at most  $\frac{n}{\varepsilon_d} \text{Vol}(e(s) \cap \text{Top}(j, \lambda))$  which by Eq. 3 is bounded by  $O(\frac{\lambda}{\varepsilon_d} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)})$ . We are picking the point  $f(\mathbf{g})$  randomly among the  $\lambda$  points inside the  $\text{Top}(j, \lambda)$  which implies the probability that  $f(\mathbf{g})$  gets covered is at most

$$O\left(\frac{1}{\varepsilon_d} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}\right). \tag{4}$$

Note that above inequality is only with respect to the *random choices of  $\mathbf{g}$*  and ignores the probability of  $s \in \mathbf{S}'_j$ . However, the only necessary condition for a sum  $s \in \mathcal{M}_j$  to be in  $\mathbf{S}'_j$

### 3:12 A New Lower Bound for Semigroup Orthogonal Range Searching

is that its  $Y$ -coordinate falls within the top and bottom boundaries of  $\text{Top}(j, \lambda)$  along the  $Y$ -axis. The probability of this event is at most  $\beta$  by construction. As this probability is independent of choice of  $\mathbf{p}$ , we have

$$\Pr[s \text{ covers } f(\mathbf{g}), s \in \mathbf{S}'_j] = O\left(\frac{\beta}{\varepsilon_d} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}\right). \quad (5)$$

Now we consider the definition of the potential function  $\Psi$  to realize that we have

$$\sum_{k_{t+1}=0}^{\infty} \cdots \sum_{k_{d-1}=0}^{\infty} \frac{\#_{k_{t+1}, \dots, k_{d-1}}}{2^{k_{t+1} + \dots + k_{d-1}}} = \Psi_{v_{t+1}, \dots, v_{d-1}} = \sum_{s \in \mathbf{S}'_j} \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}. \quad (6)$$

The left hand side is the definition of the potential function  $\Psi$  where as the right hand side counts exactly the same concept: a sum  $s$  placed at depth  $\ell_i + j_i + \zeta_i(s)$  of  $T_i$  and at a descendant of  $v_i$ , for  $t+1 \leq i \leq d-1$ , contributes exactly  $\prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}$  to the potential  $\Psi$ .

Remember that  $m(\mathbf{p})$  is the number of sums that cover a random point  $\mathbf{p}$  selected uniformly among the points inside  $\text{Top}(j, \lambda)$ . We have

$$\begin{aligned} \sum_{s \in \mathcal{M}_j} \Pr[s \text{ covers } f(\mathbf{g}), s \in \mathbf{S}'_j] &= \sum_{s \in \mathcal{M}_j} O\left(\frac{\beta \prod_{i=t+1}^{d-1} 2^{-\zeta_i(s)}}{\varepsilon_d}\right) = && \text{(from Eq. 5)} \\ O\left(\frac{\beta \Psi_{v_{t+1}, \dots, v_{d-1}}}{\varepsilon_d}\right) &= O\left(\frac{\beta}{\varepsilon_d} \cdot \frac{|S^+(\mathcal{D})| \prod_{i=1}^t j_i}{\varepsilon^2 h^{d-1} \prod_{i=1}^{d-1} 2^{j_i + \ell_i}}\right) = && \text{(from Eq. 6 and Eq. 2)} \\ O\left(\frac{\frac{\lambda}{n} \prod_{i=1}^{d-1} 2^{\ell_i + j_i}}{\varepsilon_d} \cdot \frac{|S^+(\mathcal{D})| \prod_{i=0}^t j_i}{\varepsilon^2 h^{d-1} \prod_{i=1}^{d-1} 2^{j_i + \ell_i}}\right) &= && \text{(from definition of } \beta) \\ O\left(\frac{\lambda}{n \varepsilon_d} \cdot \frac{|S^+(\mathcal{D})| \prod_{i=0}^t j_i}{\varepsilon^2 h^{d-1}}\right) &= O\left(\frac{\frac{\delta h^{d-1}}{j_1 j_2 \dots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})}}{n \varepsilon_d} \cdot \frac{|S^+(\mathcal{D})| \prod_{i=0}^t j_i}{\varepsilon^2 h^{d-1}}\right) = && \\ & && \text{(from the definition of } \lambda) \\ O\left(\frac{\delta}{\varepsilon_d \varepsilon^2}\right) &< \frac{1}{2^d C'} && \text{(from simplification and picking } \delta = O(\varepsilon_d \varepsilon^2 C'^{-1} 2^{-d}) \text{ small enough)} \end{aligned}$$

Observe that  $\Pr[s \text{ covers } f(\mathbf{g}), s \in \mathbf{S}'_j] = \frac{1}{\lambda} \sum_{g=1}^{\lambda} \Pr[s \text{ covers } f(g), s \in \mathbf{S}'_j]$ . Now our Main Lemma follows from plugging this in Eq. 1.

### 3.3 The Lower Bound Proof

Our proof strategy is to use Lemma 6 to show that the query algorithm is forced to use a lot of sums that only cover a constant number of points inside the query, leading to a large query time.

► **Theorem 12.** *Let  $P$  be a well-distributed point set containing  $\Theta(n)$  points in  $\mathbb{R}^d$ . Answering semigroup queries on  $P$  using  $S^+(n)$  storage and with  $Q(n)$  query bound requires that  $S^+(n) \cdot Q(n) = \Omega(n(\log n \log \log n)^{d-1})$ .*

We pick a random query according to the distribution defined in the previous subsection. By Lemma 6, every  $j$ -subproblem for  $1 \leq j_i \leq h/2$ , has a constant probability of being well-defined. Let  $\mathcal{W}$  be the set of all the well-defined subproblems. For a  $j$ -subproblem, let  $\lambda_j$  be the value  $\lambda$  as it is defined in Lemma 6. Observe that if a  $j$ -subproblem for  $j = (j_1, \dots, j_{d-1})$ , is well-defined, then  $\text{Top}(j, \lambda_j)$  contains  $\lambda_j = \frac{\delta h^{d-1}}{j_1 j_2 \dots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})}$  points. However, if  $j$ -subproblem

or  $\text{Top}(j, \lambda_j)$  is not well-defined, then we consider  $\text{Top}(j, \lambda_j)$  to contain 0 points. We define the top of the query,  $\text{Top}(q)$ , to be the set of points  $\cup_{j=(j_1, \dots, j_{d-1}), 1 \leq j_1, \dots, j_{d-1} \leq h/2} \text{Top}(j, \lambda_j)$ . As each  $j$ -subproblem and  $\text{Top}(j, \lambda_j)$ , for  $j_i \leq h/2$  has a constant probability of being well-defined, we have

$$\begin{aligned} \mathbb{E}[|\text{Top}(q)|] &= \sum_{j=(j_1, \dots, j_{d-1}), 1 \leq j_1, \dots, j_{d-1} \leq \frac{h}{2}} \mathbb{E}[\text{Top}(j, \lambda_j)] \\ &= \Theta(1) \sum_{j_1=1}^{h/2} \dots \sum_{j_{d-1}=1}^{h/2} \frac{\delta h^{d-1}}{j_1 j_2 \dots j_{d-1}} \cdot \frac{n}{S^+(\mathcal{A})} \\ &= \sum_{j_1=1}^{h/2} \dots \sum_{j_{d-2}=1}^{h/2} \frac{\delta \Theta(\log h) h^{d-1}}{j_1 j_2 \dots j_{d-2}} \cdot \frac{n}{S^+(\mathcal{A})} \\ &= \dots = \Theta\left(\frac{\delta \log^{d-1} h h^{d-1} n}{S^+(\mathcal{A})}\right). \end{aligned} \tag{7}$$

In the full paper [1], we show that we can find a subset  $\text{Top}(q)$  that contain at least a constant fraction its points, s.t., every sum can cover at most a constant number of points in this subset. As a result, the total number of sums required to cover the points in  $\text{Top}(q)$  is asymptotically the same as Eq. 7, our claimed lower bound. This would complete the proof.

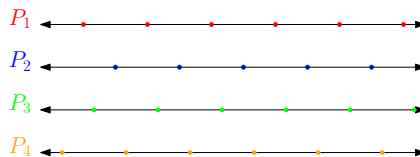
#### 4 The Upper Bounds

In the full version of our paper [1], we prove the following theorem that shows the analysis of our lower bound from the previous section is almost tight.

► **Theorem 13.** *For a set  $P$  of  $n$  points placed uniformly randomly inside the unit cube in  $\mathbb{R}^d$ , one can build a data structure that uses  $O(n)$  storage such that a  $(d+k)$ -sided query can be answered with the expected query bound of  $O(\log^{d-1} n (\log \log n)^k)$ , for  $1 \leq k \leq d-1$ .*

*If  $P$  is well-distributed, then the query bound can be made worst-case.*

Due to lack of space, the technical parts of the proof appear in the full version only. However, the main idea is to simulate the phenomenon we have captured in our lower bound: the idea that one can store sums such that the sums from different subproblems “help” each other. To do that, we define the notion of “collectively well-distributed” point sets. Intuitively, collectively well-distributed point sets is a collection of point sets  $\mathcal{P}$  where each element of  $\mathcal{P}$  is a well-distributed point set but importantly, certain unions of the point sets in  $\mathcal{P}$  are also well-distributed point sets. See Fig. 4 for an example. It turns out that we can turn any properly “collectively well-distributed” point set  $\mathcal{P}$  that contains  $O(n)$  points, into a data structure that uses  $O(n)$  space (i.e., we form a constant number of sums per point in  $\mathcal{P}$ ) and has the desired query time.



■ **Figure 4** The point sets  $P_1, P_2, P_3, P_4$  are well-distributed. For any continuous set of integers  $I \subset \{1, \dots, 4\}$ ,  $\cup_{i \in I} P_i$  is also well-distributed but  $P_1 \cup P_3$  might not be well-distributed.



## 5 Conclusions

In this paper we considered the semigroup range searching problem from a lower bound point of view. We improved the best previous lower bound trade-off offered by Chazelle by analysing a well-distributed point set for  $(2d - 1)$ -sided queries for an idempotent semigroup. Furthermore, we showed that our analysis is tight which leads us to suspect that we have found an (almost) optimal lower bound for idempotent semigroups as we believe it is unlikely that a more difficult point set exists. Thus, two prominent open problems emerge: (i) Can we improve the known data structures under the *extra* assumption that the semigroup is idempotent? (ii) Can we improve our lower bound under the *extra* assumption that the semigroup is not idempotent? Note that the effect of idempotence on other variants of range searching was studied at least once before [6].

---

## References

- 1 Peyman Afshani. A New Lower Bound for Semigroup Orthogonal Range Searching, 2019. [arXiv:1903.07967](https://arxiv.org/abs/1903.07967).
- 2 Peyman Afshani, Lars Arge, and Kasper Green Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Symposium on Computational Geometry (SoCG)*, pages 323–332, 2012. doi:10.1145/2261250.2261299.
- 3 Peyman Afshani and Anne Drimel. On the complexity of range searching among curves. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 898–917, 2017.
- 4 Pankaj K. Agarwal. Range searching. In J. E. Goodman, J. O’Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 2016.
- 5 N. Alon and B. Schieber. OPTIMAL PREPROCESSING FOR ANSWERING ON-LINE PRODUCT QUERIES. Technical Report 71/87, Tel-Aviv University, 1987.
- 6 Sunil Arya, Theocharis Malamatos, and David M. Mount. On the Importance of Idempotence. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 564–573, 2006.
- 7 Jon Louis Bentley. Decomposable searching problems. *Information Processing Letters (IPL)*, 8(5):244–251, 1979.
- 8 Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM (CACM)*, 23(4):214–229, 1980.
- 9 Bernard Chazelle. Lower bounds for orthogonal range searching: part II. The arithmetic model. *Journal of the ACM (JACM)*, 37(3):439–463, 1990.
- 10 Michael L. Fredman. The inherent complexity of dynamic data structures which accommodate range queries. In *Proc. 21st Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 191–199, October 1980.
- 11 Michael L. Fredman. A Lower Bound on the Complexity of Orthogonal Range Queries. *Journal of the ACM (JACM)*, 28(4):696–705, 1981.
- 12 Michael L. Fredman. Lower Bounds on the Complexity of Some Optimal Data Structures. *SIAM Journal of Computing*, 10(1):1–10, 1981.
- 13 Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences (JCSS)*, 18(2):110–127, 1979.
- 14 Andrew C. Yao. Space-time Tradeoff for Answering Range Queries (Extended Abstract). In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, STOC ’82, pages 128–136. ACM, 1982.



# Independent Range Sampling, Revisited Again

**Peyman Afshani**

Aarhus University, Denmark  
peyman@cs.au.dk

**Jeff M. Phillips**

University of Utah, Salt Lake City, USA  
jeffp@cs.utah.edu

---

## Abstract

We revisit the range sampling problem: the input is a set of points where each point is associated with a real-valued weight. The goal is to store them in a structure such that given a query range and an integer  $k$ , we can extract  $k$  independent random samples from the points inside the query range, where the probability of sampling a point is proportional to its weight.

This line of work was initiated in 2014 by Hu, Qiao, and Tao and it was later followed up by Afshani and Wei. The first line of work mostly studied unweighted but dynamic version of the problem in one dimension whereas the second result considered the static weighted problem in one dimension as well as the unweighted problem in 3D for halfspace queries.

We offer three main results and some interesting insights that were missed by the previous work: We show that it is possible to build efficient data structures for range sampling queries if we allow the query time to hold in expectation (the first result), or obtain efficient worst-case query bounds by allowing the sampling probability to be approximately proportional to the weight (the second result). The third result is a conditional lower bound that shows essentially one of the previous two concessions is needed. For instance, for the 3D range sampling queries, the first two results give efficient data structures with near-linear space and polylogarithmic query time whereas the lower bound shows with near-linear space the worst-case query time must be close to  $n^{2/3}$ , ignoring polylogarithmic factors. Up to our knowledge, this is the first such major gap between the expected and worst-case query time of a range searching problem.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Computational geometry

**Keywords and phrases** Range Searching, Data Structures, Sampling

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.4

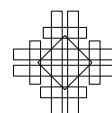
**Related Version** A full version of the paper is available at [1], <https://arxiv.org/abs/1903.08014>.

**Funding** *Peyman Afshani*: supported by DFF (Det Frie Forskningsråd) of Danish Council for Independent Research under grant ID DFF-7014-00404.

*Jeff M. Phillips*: supported by NSF CCF-1350888, CNS-1514520, CNS-1564287, IIS-1816149, and in particular ACI-1443046. Part of the work was completed while visiting the Simons Institute for Theory of Computing.

## 1 Introduction

In range searching, the goal is store a set  $P$  of points in a data structure such that given a query range, we can answer certain questions about the subset of points inside the query range. The difficulty of the range searching problem, thus depends primarily on the shape of the query as well as types of questions that the data structure is able to answer. These range searching questions have been studied extensively and we refer the reader to the survey by Agarwal and Erickson [3] for a deeper review of range searching problems.



## 4:2 Independent Range Sampling, Revisited Again

Let us for a moment fix a particular query shape. For example, assume we are given a set  $P \subset \mathbb{R}^2$  to preprocess and at the query time we will be given a query halfplane  $h$ . The simplest type of question is an *emptiness* query where we simply want to report to the user whether  $h \cap P$  is empty or not. Within the classical literature of range searching, the most general (and thus the most difficult) variant of range searching is semigroup range searching where each point in  $P$  is assigned a weight from a semigroup and at the query time the goal is to return the sum of the weights of the points of  $h \cap P$ . The restriction of the weights to be from a semigroup is to disallow subtraction. As a result, semigroup range searching data structures can answer a diverse set of questions. Other classical variants of range searching lie between the emptiness and the semigroup variant. In our example, the emptiness queries can be trivially solved with  $O(n)$  space and  $O(\log n)$  query time whereas the semigroup variant can only be solved with  $O(\sqrt{n})$  query time using  $O(n)$  space. Finally, the third important variant, range reporting, where the goal is to output all the points in  $P \cap h$ , is often close to the emptiness variant in terms of difficulty. E.g., halfplane queries can be answered in  $O(\log n + k)$  time where  $k$  is the size of the output, using  $O(n)$  space.

**Sampling queries.** Let  $P$  be a large set of points that we would like to preprocess for range searching queries. Consider a query range  $h$ . Classical range searching solutions can answer simple questions such as the list of points inside  $h$ , or the number of them. However, we immediately hit a barrier if we are interested in more complex questions, e.g., what if we want to know how a “typical” point in  $h$  looks like? Or if we are curious about the distribution of the data in  $h$ . In general, doing more complex data analysis requires that we extract the list of all the points inside  $h$  but this could be an expensive operation. For questions of this type, as well as many other similar questions, it is very useful to be able to extract a (relatively) small random sample from the subset of points inside  $h$ . In fact, range sampling queries were considered more than two decades ago within the context of database systems [7, 4, 5, 11, 10]. Indeed the entire field of sample complexity, which provides the basis for statistics and machine learning, argues how any statistical quantity can be understood from an iid sample of the data. Range sampling allows this literature to quickly be specified to data in a query range. However, many of these classical solutions fail to hold in the worst-case as they use R-trees or Quadrees whose performance depends on the distribution of the coordinates of the input points (which could be pretty bad). Some don’t even guarantee that the samples extracted in the future will be independent of the samples extracted in the past. For example, sometimes asking the same query twice would return the same samples.

**The previous results.** Given a set of  $n$  weights, one can sample one weight with proportional probability using the well-known “alias method” by A. J. Walker [12]. This method uses linear space and can sample a weight in worst-case constant time.

The rigorous study of *independent range sampling* was initiated by Hu *et al.* [8]. They emphasized the requirement that any random sample extracted from the data structure should be independent of all the other samples. They studied the one-dimensional version and for an unweighted point set and presented a linear-sized data structure that could extract  $k$  random samples in  $O(\log n + k)$  time and it could be updated in  $O(\log n)$  time as well. A few years later, Afshani and Wei [2] revisited the problem and solved the one-dimensional version of the problem for weighted points: they presented a linear-size data structure that could answer queries in  $O(\text{Pred}(n) + k)$  time where  $\text{Pred}(n)$  referred to the running time of a predecessor query (often  $O(\log n)$  but sometimes it could be faster, e.g., if the input is indexed by an array then the predecessor query can be answered trivially in  $O(1)$  time). They also studied the 3D halfspace queries but for unweighted points. Their main result was an optimal data structure of linear-size that could answer queries in  $O(\log n + k)$  time.

**Our results.** We provide general results for the independent range sampling problem on weighted input (wIRS), and design specific results for 3D halfspace queries. We show a strong link between the wIRS and the range max problem. Namely, we show that the range max problem is at least as hard as wIRS, and also we provide a general formulation to solve the wIRS problem using range max. For halfspace queries in 3D, our framework gives a structure that uses  $O(n \log n)$  space and has  $O(\log^2 n + k)$  query time. We improve the space complexity to  $O(n)$  when the ratio of the weights is  $n^{O(1)}$ . This solution uses rejection sampling, so it only provides an expected query time bound. To compensate, we provide another solution that has worst-case query time, but allows the points to be sampled within a  $(1 \pm \varepsilon)$  factor of their desired probability, and may ignore points that would be sampled with probability less than  $\gamma/n$  for  $\gamma < \varepsilon < 1$ . This structure requires  $O(n \log n)$  space, and has a worst-case query time of  $O(\log(n/\gamma)(\log n + 1/\varepsilon^3) + k)$ .

Finally, we show a conditional lower bound when we enforce worst-case query time and exact sampling probabilities, in what we call the separated algebraic decision tree (SAD) model. This model allows any decision tree structure that compares random bits to algebraic functions of a set of input weights. In this model, we show wIRS is as hard as the range sum problem, which is conjectured to be hard. In particular, if the best known solution to the range sum problem for halfspaces in 3D is optimal, then the wIRS problem would require  $\Omega(n^{2/3-o(1)})$  query time if it uses near-linear space. This provides the first such separation between expected  $O(\log^2 n + k)$  and worst-case  $\Omega(n^{2/3-o(1)})$  query time for a range searching problem that we are aware of.

## 2 A Randomized Data Structure

In this section, we show that if we allow for the query bound to hold in expectation, then the range sampling problem can be solved under some general circumstances. Intuitively, we show that we need two ingredients: one, a data structure for range maximum queries, and two, a data structure that can sample from a weighted set of points under the assumptions that the weights are within a constant factor of each other. Furthermore, with an easy observation, we can also show that the range sampling problem is at least as hard as range maximum problem. We consider the input as a general set system  $(X, \mathcal{R})$ . We assume the input is a set  $X$  of  $n$  data elements (e.g., points) and we consider queries to be elements of a set of ranges  $\mathcal{R}$  where each  $R \in \mathcal{R}$  is a subset of  $X$ . The set  $\mathcal{R}$  is often given implicitly, and for example, if  $X$  is a set of points in  $\mathbb{R}^3$ ,  $\mathcal{R}$  could be the set of all  $h \cap X$  where  $h$  is a halfspace in 3D. Note that our model of computation is the real RAM model.

► **Definition 1** (The Range Maximum Problem). *Let  $X$  be a data set, s.t., each element  $x \in X$  is assigned a real-value weight  $w(x)$ . The goal is to store  $X$  in a structure, s.t., given a range  $R \in \mathcal{R}$ , we can find the element in  $R$  with the maximum weight.*

Given a weighted set  $X$ , for any subset  $Y \subset X$ , we denote by  $w(Y)$  the sum  $\sum_{x \in Y} w(x)$ . First we observe that range sampling is at least as hard as the range maximum problem.

► **Lemma 2.** *Assume we can solve range sampling queries on input  $(X, \mathcal{R})$  and for any weight assignment  $w: X \rightarrow \mathbb{R}$  using  $S(|X|)$  space and  $Q(|X|)$  query time where the query only returns one sample. Then, given the set  $X$  and a weight function  $w': X \rightarrow \mathbb{R}$ , we can store  $X$  in a data structure of size  $S(|X|)$  such that given a query  $R$ , we can find the data element in  $R$  with the maximum weight in  $Q(|X|)$  time, with high probability.*

**Proof.** See the full paper for the proof [1]. ◀

## 4:4 Independent Range Sampling, Revisited Again

Next, we show that weighted range sampling can be obtained from a combination of a range maximum data structure and a particular form of weighted range sampling data structure for almost uniform weights.

► **Lemma 3.** *Let  $(X, \mathcal{R})$  be an input to the range sampling problem. Assume, we have a structure for the range maximum queries that uses  $O(S_m(|X|))$  space and with query time of  $O(Q_m(|X|))$ . Furthermore, assume for any subset  $X' \subset X$  we can build a structure  $\mathcal{D}_s(X')$  that uses  $O(S_s(|X'|))$  space and given a query  $R \in \mathcal{R}$ , it does the following: it can return  $w(Y)$  for a subset  $Y \subset X'$  with the property that  $R \cap X' \subset Y$ , and  $|Y| = O(|R \cap X'|)$  and furthermore, the structure can extract  $k$  random samples from  $Y$  in  $O(Q_s(|X'|) + k)$  time.*

*Then, we can answer range sampling queries using  $O(S_m(|X|) + S_s(|X|))$  space and with expected query time of  $O(Q_m(|X|) + Q_s(|X|) \log |X| + k)$ .*

**Proof.** Let  $n = |X|$ . We store  $X$  in a data structure for the range maximum queries. We partition  $X$  into subsets  $X_i \subset X$  in the following way. We place the element  $x_1$  with the largest weight in  $X_1$  and then we add to  $X_1$  any element whose weight is at least  $w(x_1)/2$  and then recurse on the remaining elements. Observe that for all  $x, x' \in X_i$  we have  $w(x)/w(x') \in (1/2, 2]$ . Thus, the weight function  $w$  is almost uniform on each  $X_i$ . We store  $X_i$  in a data structure  $\mathcal{D}_s(X_i)$ .

Next, we build a subset sum information over the total weight  $W(X_i)$  of the (disjoint) union of all subsets  $X_{i'}$  with  $i' \geq i$ , that is,  $W(X_i) = \sum_{j \geq i} w(X_j)$ . Consider a query  $R \in \mathcal{R}$ .

**Step 1: Use the Range Maximum Structure.** Issue a single range-max query on  $X$  for the query range  $R \in \mathcal{R}$ . Let  $x$  be the answer to the range-max query, assume  $x \in X_i$ .

**Step 2: top-level alias structure.** Having found  $i$ , we identify the smallest index  $i' \geq i$  such that the maximum weight  $w(x')$  for  $x' \in X_{i'}$  and the minimum weight  $w(x)$  for  $x \in X_i$  satisfy  $w(x)/w(x') > n^2$ . As the weights in the sets  $X_i$  decreases geometrically, we have  $i' - i = O(\log n)$ . Then, for each  $X_j$  with  $j \in [i, i')$ , we use the data structure  $\mathcal{D}_s(X_j)$  to identify the set  $Y_j$  such that  $Y_j$  contains the set  $R \cap X_j$ . This returns the values  $w(Y_j)$ , and the total running time of this step is  $O(Q_s(n) \log n)$ .

We build a top-level alias structure on  $i' - i + 1$  values: all the  $i' - i$  values  $w(Y_j)$ ,  $i \leq j < i'$ , as well as the value  $W(X_{i'})$ . Let  $T = W(X_{i'}) + \sum_{j=i}^{i'-1} w(Y_j)$ . This can be done in  $O(\log n)$  time as  $i' - i = O(\log n)$ .

**Step 3: Extracting samples.** To generate  $k$  random samples from  $R$ , we first sample a value using the top-level alias structure. This can result in two different cases:

**Case 1.** It returns the value  $W(X_{i'})$ . Let  $X_{i'+} = \cup_{j \geq i'} X_j$ . In this case, we sample an element from the set  $R \cap X_{i'+}$ , by building an alias structure on  $X_{i'+}$  in  $O(n)$  time. The probability of sampling an element  $x_j \in X_{i'+}$  is set exactly to  $\frac{w(x_j)}{W(X_{i'+})}$ . Note that these probabilities of  $x_j \in X_{i'+} \cap R$  do not add up to one which means the sampling might fail and we might not return any element. If this happens, we go back to the top-level alias structure and try again. Notice that  $R$  contains at least one element  $x_i$  from  $X_i$ , and that for any  $x \in X_j$ ,  $j \geq i'$  have  $w(x) \leq w(x_i)/n^2$  which implies  $W(X_{i'+}) \leq w(x_i)/n$ . Thus, this case can happen with probability at most  $1/n$ , meaning, even if we spend  $O(n)$  time to answer the query, the expected query time is  $O(1)$ .

**Case 2.** It returns a value  $w(Y_j)$ , for  $i \leq j < i'$ . We place  $Y_j$  into a list for now. At some later point (to be described) we will extract a sample  $z$  from  $Y_i$ . If  $z$  happens to be inside  $R$ , then we return  $z$ , otherwise, the sampling fails and we go back to the top-level structure.

We iterate until  $k$  queries have been pooled. Then, we issue them in  $O(\log n)$  batches to data structure  $\mathcal{D}_s(X_j)$ ,  $i \leq j < i'$ . Ignoring the failure events in case (2), processing a batch of  $k$  queries will take  $O(Q_s(n) \log n + k)$  time. Notice that each iteration of the above procedure will succeed with a constant probability: case (1) is very unlikely (happens with probability less than  $1/n$ ) and for case (2) observe that we have  $w(Y_i) = O(w(R \cap X_i))$  and thus each query will succeed with constant probability. As a result, in expectation we only issue a constant number of batches of size  $k$  to extract  $k$  random samples.

It remains to show that we sample each element with the correct probability. The probability of reaching case (1) is equal to  $\frac{W(X_{i'})}{T}$ . Thus, the probability of sampling an element  $x_j \in X_{i'+}$  is equal to  $\frac{W(X_{i'})}{T} \cdot \frac{w(x_j)}{W(X_{i'})} = \frac{w(x_j)}{T}$ . The same holds in case (2) and the probability of sampling an element  $x_j \in X_j$ ,  $i \leq j < i'$  is  $\frac{w(x_j)}{T}$ . Thus, conditioned on the event that the sampling succeeds, each element is sampled with the correct probability. ◀

### 3 3D Halfspace Sampling

#### 3.1 Preliminaries

In this section, we consider random sampling queries for 3D halfspaces. But we first need to review some preliminaries.

► **Lemma 4.** *Let  $P$  be a set of  $n$  points in  $\mathbb{R}^3$ . Let  $P = P_1 \cup \dots \cup P_t$  be a partition of  $P$  into  $t$  subsets. We can store  $P$  in a data structure of size  $O(n \log t)$  such that given a query halfspace  $h$ , we can find the smallest index  $i$  such that  $P_i \cap h \neq \emptyset$  in  $O(\log n \log t)$  time.*

**Proof.** See the full paper for the proof [1]. ◀

The following folklore result is a special case of the above lemma.

► **Corollary 5.** *Let  $P$  be a set of  $n$  points in  $\mathbb{R}^3$  where each point  $p \in P$  is assigned a real-valued weight  $w(p)$ . We can store  $P$  in a data structure of size  $O(n \log n)$  such that given a query halfspace  $h$ , we can find the point with maximum weight in  $O(\log^2 n)$  time.*

We also need the following preliminaries. Given a set  $H$  of  $n$  hyperplanes in  $\mathbb{R}^3$ , the *level* of a point  $p$  is the number hyperplanes that pass below  $p$ . The  $(\leq k)$ -*level* of  $H$  (resp.  $k$ -*level* of  $H$ ) is the closure of the subset of  $\mathbb{R}^3$  containing points with level at most  $k$  (resp. exactly  $k$ ). An *approximate  $k$ -level* of  $H$  is a surface composed of triangles (possibly infinite triangles) that lies above  $k$ -level of  $H$  but below  $(ck)$ -level of  $H$  for a fixed constant  $c$ . For a point  $q \in \mathbb{R}^3$ , we define the conflict list of  $q$  with respect to  $H$  as the subset of hyperplanes in  $H$  that pass below  $q$  and we denote this with  $\Delta(H, q)$ . Similarly, for a triangle  $\tau$  with vertices  $v_1, v_2$ , and  $v_3$ , we define  $\Delta(H, \tau) = \Delta(H, v_1) \cup \Delta(H, v_2) \cup \Delta(H, v_3)$ . One of the main tools that we will use is the existence of small approximate levels. This follows from the existence of shallow cuttings together with some geometric observations.

► **Lemma 6.** *For any set  $H$  of  $n$  hyperplanes in  $\mathbb{R}^3$ , and any parameter  $1 \leq k \leq n/2$ , there exists an approximate  $k$ -level which is a convex surface consisting of  $O(n/k)$  triangles.*

*Furthermore, we can construct a hierarchy of approximate  $k_i$ -levels  $\mathcal{L}_i$ , for  $k_i = 2^i$  and  $i = 0, \dots, \log n$ , together with the list  $\Delta(H, \tau)$  for every triangle  $\tau \in \mathcal{L}_i$  in  $O(n \log n)$  time and  $O(n)$  space. Given a query point  $q$ , we can find an index  $i$  in  $O(\log n)$  time such that there exists a triangle  $\tau \in \mathcal{L}_i$  that lies above  $q$  such that  $|\Delta(H, q)| = O(|\Delta(H, \tau)|)$ .*

**Proof.** See the full paper for the proof [1]. ◀

## 4:6 Independent Range Sampling, Revisited Again

We will also use the following results.

► **Theorem 7** (The Partition Theorem, [9]). *Given a set  $P$  of  $n$  points in 3D and an integer  $0 < r \leq n/2$ , there exists a partition of  $P$  into  $r$  subsets  $P_1, \dots, P_r$ , each of size  $\Theta(n/r)$ , where each  $P_i$  is enclosed by a tetrahedron  $T_i$ , s.t., any hyperplane crosses  $O(r^{2/3})$  tetrahedra.*

► **Lemma 8.** *Let  $T$  be tree of size  $n$  where each leaf  $v$  stores a real-valued non-negative weight  $w(v)$ . We can build a data structure s.t., given an internal node  $u \in T$  at the query time, we can independently sample a leaf with probability proportional to its weight in the subtree of  $u$ . The data structure uses  $O(n)$  space and it can answer queries in  $O(1)$  worst-case time.*

**Proof.** See the full paper for the proof [1]. ◀

Afshani and Wei [2] consider unweighted sampling for 3D halfspace queries. We next use the following technical result of theirs.

► **Lemma 9.** *Let  $H$  be a set of  $n$  hyperplanes in 3D. Let  $f(n) = (\log n)^{c \log \log n}$  where  $c$  is a large enough constant. We can build a tree  $T_{\text{global}}$  with  $n$  leafs where each hyperplane is stored in one leaf such that the following holds: Given a point  $q \in \mathbb{R}^3$  with level  $k$  where  $k \geq f(n)$ , we can find  $k' = O(k/\log^2 n)$  internal nodes  $u_1, \dots, u_{k'}$  in  $T_{\text{global}}$  such that  $\Delta(H, q) = T_{\text{global}}(u_1) \cup \dots \cup T_{\text{global}}(u_{k'})$  where  $T_{\text{global}}(u_i)$  is the set of hyperplanes stored in the subtree of  $u_i$ .*

Unfortunately, the above lemma is not stated explicitly by Afshani and Wei, however,  $T_{\text{global}}$  is the “Global Structure” that is described in [2] under the same notation.

### 3.2 A Solution with Expected Query Time

We now observe that we can use Lemma 3 to give a data structure for weighted halfspace range sampling queries in 3D. We first note that using Corollary 5 and by building a hierarchy of shallow cuttings, i.e., building approximate  $k_i$ -levels for  $k_i = 2^i$ ,  $i = 0, \dots, \log n$ , we can get a data structure with  $O(n \log n)$  space that can answer queries in  $O(\log^2 n + k)$  query time. Furthermore, as Lemma 2 shows, our problem is at least as hard as the halfspace range maximum problem which currently has no better solution than  $O(n \log n)$  space and  $O(\log^2 n)$  expected query time. Thus, it seems we cannot do better unless we can do better for range maximum queries, a problem that seems very difficult.

However, the reduction given by Lemma 2 is not completely satisfying since we need to create a set of weights that are exponentially distributed. As a result, it does not capture a more “natural” setting where the ratio between the largest and the smallest weight is bounded by a parameter  $U$  that is polynomial in  $n$ . Our improved solution is the following which shows when  $U = n^{O(1)}$  we can in fact reduce the space to linear.

► **Theorem 10.** *Let  $P$  be a set of  $n$  weighted points in  $\mathbb{R}^3$ , where the smallest weight is 1 and the largest weight is  $U$ . We can store  $P$  in a data structure of size  $O(n \min\{\log n, \log \log_n U\})$  such that given a query halfspace and an integer  $k$ , we can extract  $k$  independent random samples from the points inside  $h$  in  $O(\log^2 n + k)$  time.*

**Proof summary.** For the complete proof see the full paper [1]. The basic idea is to partition the input into sets where the ratio of weights is within  $n^2$ . Then, we use Lemmas 9 and 8. ◀

### 3.3 Worst-Case Time with Approximate Weights

All of the previous data structures sample items exactly proportional to their weight, but rely on rejection sampling. Hence, their query time is expected, and not worst case. With small probability these structures may repeatedly sample items which are not in  $h$ , and then need to reset and try again with no bound on the worst-case time. To achieve worst-case query time, we need some modifications. We allow for items to be sampled “almost” proportional to their weights, i.e., we introduce a notion of approximation. As we shall see in the next chapter, without some kind of approximation, our task is very likely impossible.

**Problem definition.** We consider an input set  $X$  of  $n$  points where  $x_i$  has weight  $w_i$  and we would like to store  $X$  in a data structure. At the query time, we are given a halfspace  $h$  and a value  $k$  and we would like to extract  $k$  random samples from  $X \cap h$ . Let  $w(h) = \sum_{x_i \in X \cap h} w_i$ , and set two parameters  $0 < \gamma < \varepsilon < 1$ . Ideally, we would like to sample each  $x_i$  with probability  $w_i/w(h)$ . Instead we sample  $x_i$  with probability  $\rho_i$ , if  $w_i/w(h) \geq \gamma/n$  then

$$(1 - \varepsilon) \frac{w_i}{w(h)} \leq \rho_i \leq (1 + \varepsilon) \frac{w_i}{w(h)}, \quad (1)$$

and if  $w_i/w(h) < \gamma/n$  then we must have  $\rho_i \leq (1 + \varepsilon) \frac{w_i}{w(h)}$ . That is, we sample all items within a  $(1 \pm \varepsilon)$  factor of their desired probability, except for items with very small weight, which could be ignored and not sampled. The smaller items are such that the sum of their desired probabilities is at most  $\gamma$ .

**Overview of modifications.** We start in the same framework as Section 3.2 and Lemma 3, i.e., we partition  $X$  into subsets  $X_\ell$  by weight. Then, we need to make the following modifications: (1) We can now ignore small enough weights; (2) We can no longer use rejection sampling to probe into each set  $X_\ell$ , rather we need to collect a bounded number (a function  $f(\varepsilon)$ ) of candidates from all  $X_\ell$  which is guaranteed to contain some point in the query  $h$ . (3) We can also no longer use rejection sampling within each  $X_\ell$  to get candidates, instead we build a stratified sample via the partition theorem.

Change (1) is trivial to implement. Remember that given query  $h$ , we first identify indices  $i$  and  $i'$  such that  $X_i$  contains the largest weight in  $h$  and the weights in  $X_{i'}$  are a factor  $n^2$  smaller. We now require weights in  $X_{i'}$  to be a factor  $n/\gamma$  smaller, and let  $i' = i + t$  where  $t = O(\log(n/\gamma))$ . We can now ignore all the remaining sets: the sets  $X_j$  with  $j \geq i + t$  will have weights so small that even if there are  $\Omega(n)$  points within, the sum of their weights will be at most  $\gamma$  times the largest weight. Since  $\gamma < \varepsilon$ , this implicitly increases all of the other weights in each  $X_\ell$  (for  $\ell \in [i, i + t)$ ) by at most a factor  $(1 + \varepsilon)$ .

We next describe how we can build a data structure on each  $X_\ell$  to generate  $f(1/\varepsilon)$  candidate points. Once we have these  $t \cdot f(1/\varepsilon)$  points, we can build two alias structures on them (they will come with weights proportional to the probability they should be selected), and select points until we find one in  $h$ . As a first pass, to generate  $k$  samples, we can repeat this  $k$  times, or bring  $kf(1/\varepsilon)$  samples from each  $X_\ell$ . We will return to this subproblem to improve the dependence on  $k$  and  $\varepsilon$  by short-circuiting a low-probability event and reloading these points dynamically.

#### 3.3.1 Generating Candidate Points

Here we will focus on sampling our set of candidates. We will do this for every set  $X_\ell$ ,  $i \leq \ell < i'$ . However, to simplify the presentation, we will assume that the input is a set  $X$  of  $n$  points such that the weights of the points in  $X$  are within factor 2 of each other. We will



## 4:8 Independent Range Sampling, Revisited Again

sample  $f(1/\varepsilon)$  candidate points from  $X$  (representing a subset  $X_\ell$ ) s.t., the set of candidates intersects with the query halfspace  $h$ . Each candidate will be sampled with a probability that is almost uniform, i.e., it fits within our framework captured by Eq. 1.

Let  $H$  be the set of hyperplanes dual to points in  $X$ . We maintain a hierarchical shallow-cutting of approximate levels (Lemma 6) on  $H$ . By Lemma 6, we get the following in the primal setting (on  $X$ ), given a query halfspace  $h$ : We can build  $O(|X|)$  subsets of  $X$  where the subsets have in total  $O(|X| \log |X|)$  points. Given a query halfspace  $h$ , in  $O(\log n)$  time, we can find a subset  $X'$  so that  $X \cap h \subset X'$  and  $|X'| = O(|X \cap h|)$ . We now augment this structure with the following information on each such subset  $X'$ , without increasing the space complexity. We maintain an  $r$ -partition  $(Z_1, \Delta_1), (Z_2, \Delta_2), \dots, (Z_{r'}, \Delta_{r'})$  on  $X'$ . (That is, so  $r' = \Theta(r)$ , each subset  $Z_j \subset \Delta_j$  and has size bound  $|X'|/r \leq |Z_j| \leq 2|X'|/r$ , and the boundary of any halfspace  $h$  intersects at most  $O(r^{2/3})$  cells  $\Delta_j$ .) For each  $Z_j$  we maintain an alias structure so in  $O(1)$  time we can generate a random  $s_j$  from the points within. It is given a weight  $W_j = \sum_{x \in Z_j} w(x)$ . In  $O(r)$  time we can generate a weighted set  $S = \{s_1, s_2, \dots, s_{r'}\}$ ; this will be the candidate set.

The sum of all weights of points within  $h$  is  $w(h) = \sum_{x \in X \cap h} w(x)$ , and so we would like to approximately sample each  $x \in X' \cap h$  with probability  $w(x)/w(h)$ .

► **Lemma 11.** *Let  $r = \Omega(1/\varepsilon^3)$  and consider a candidate set  $S$ , and sample one point proportional to their weights. For a point  $x \in X' \cap h$ , the probability  $\rho_x$  that it is selected satisfies*

$$(1 - \varepsilon) \frac{w(x)}{w(h)} \leq \rho_x \leq (1 + \varepsilon) \frac{w(x)}{w(h)}.$$

**Proof.** Of the  $r'$  cells in  $S$ , classify them in sets as *inside* if  $\Delta_j \in h$ , as *outside* if  $\Delta_j \cap h = \emptyset$ , and as *straddling* otherwise. We can ignore the outside sets. There are  $O(r)$  inside sets, and  $O(r^{2/3})$  straddling sets.

For point  $x \in S_j$  from an inside set, it ideally should be selected with probability  $\frac{w(x)}{W_j} \cdot \frac{W_j}{w(h)} = \frac{w(x)}{w(h)}$ . Indeed it is the representative of  $S_j$  with probability  $\frac{w(x)}{W_j}$  and is given weight proportional to  $W_j$  in the alias structure. We now examine two cases, that all representative points in the straddling sets are in  $h$ , and that none are; the probability  $x$  is selected will be between the probability of these two cases, and the desired probability it is selected will also be between these two cases. Let  $W_{in} = \sum_{S_j \text{ is inside}} W_j$  and  $W_{str} = \sum_{S_j \text{ is straddling}} W_j$ . The probability  $x$  is selected if it is the representative of  $S_j$  is then in the range  $[\frac{W_j}{W_{in} + W_{str}}, \frac{W_j}{W_{in}}]$ . The ratio of these probabilities is  $\frac{W_j}{W_{in}} \cdot \frac{W_{in} + W_{str}}{W_j} = \frac{W_{in} + W_{str}}{W_{in}} = 1 + \frac{O(r^{2/3})}{\Theta(r)} = 1 + O(r^{1/3})$ . Setting  $r = \Omega(1/\varepsilon^3)$  ensures that these probabilities are within a  $(1 + \varepsilon)$ -factor of each other, and on all points from an inside set, are chosen with approximately the correct probability.

For a point  $x$  in a straddling set  $S_j$ , it should be selected with probability  $\frac{w(x)}{W_j}$  and is selected with probability between  $L_x = \frac{w(x)}{W_j} \frac{W_j}{W_{in} + W_{str}}$  and  $U_x = \frac{w(x)}{W_j} \frac{W_j}{W_{in} + W_j}$ . As with points from an inside set, these are within a  $(1 + \varepsilon)$ -factor of each other if  $r = \Omega(1/\varepsilon^3)$ . And indeed since  $W_{in} \leq w(h) \leq W_{in} + W_{str}$  then  $L_x \leq \frac{w(x)}{w(h)} \leq U_x$ , and the desired probability of sampling straddling point  $x$  is in that range. ◀

### 3.3.2 Constructing the $k$ Samples

To select  $k$  random samples, the simplest way is to run this procedure  $k$  times sequentially, generating  $O(k/\varepsilon^3)$  candidate points; this is on top of  $O(\log n)$  to identify the proper subset  $X'$  from the shallow cutting, applied to all  $t = O(\log(n/\gamma))$  weight partitions  $X_\ell$ .



We can do better by first generating  $O(1/\varepsilon^3)$  candidate points per  $X_\ell$ , enough for a single random point in  $h$ . Now we place these candidates in two separate alias structures along with the candidate points from the other  $t$  structures. There are  $O(t/\varepsilon^3)$  candidate points placed in an *inside* alias structure, and  $O(t/\varepsilon^2)$  points placed in a *straddling* alias structure. Now to generate one point, we flip a coin proportional to the total weights in the two structures. If we go to the *inside* structure, we always draw a point in  $h$ , we are done. If we go to the straddling structure, we may or may not select a point in  $h$ . If we do, we are done; if not we flip another coin to decide to go to one of the two structures, and repeat.

It is easy to see this samples points with the correct probability, but it does not yet have a worst case time. We could use a dynamic aliasing structure [6] on the straddling set, so we sample those without replacement, and update the coin weight. However, this adds a  $O(\log(\frac{1}{\varepsilon} \log(n/\gamma)))$  factor to each of  $O(t/\varepsilon^2)$  steps which might be needed. A better solution is to only allow the coin to direct the sampler to the straddling sets at most once; if it goes there and fails, then it automatically redirects to the alias structure on the inside sets which must succeed. This distorts the sampling probabilities, but not by too much since in the rejection sampling scheme, the probability of going to the straddling set even once is  $O(\varepsilon)$ .

► **Lemma 12.** *For any candidate point  $s$  let  $\rho_s$  be the probability it should be sampled, and  $\rho'_s$  the probability it is sampled with the one-shot deterministic scheme. Then  $\rho_s \leq \rho'_s \leq (1 + \varepsilon)\rho_s$  if  $s$  is an inside point and  $(1 - \varepsilon)\rho_s \leq \rho'_s \leq \rho_s$  if  $s$  is from a straddling set.*

**Proof.** The probability that the coin directs to the inside set is  $\pi_{in} = \frac{W_{in}}{W_{in} + W_{str}} = \frac{1}{1 + W_{str}/W_{in}} = \frac{1}{1 + \Omega(\varepsilon)} = 1 - O(\varepsilon)$ . Let  $w_{str}$  be the probability of selecting a point inside  $h$ , given that the coin has directed to the straddling set; we only need that  $W_{str} \in [0, 1]$ .

For a candidate point in the inside set  $s$ , the probability it is selected in the rejection sampling scheme is  $\rho_s = \frac{w(s)}{W_{in}}(1 - O(\varepsilon))$ , and in the deterministic scheme is  $\rho'_s = \frac{w(s)}{W_{in}}(\pi_{in} + (1 - \pi_{in})(1 - w_{str}))$  which is in the range  $[\rho_s, \frac{w(s)}{W_{in}}]$ , and these have a ratio  $1 + O(\varepsilon)$ .

Similarly, for a candidate point in the straddling set  $s$ , the probability it is selected in the rejection sampling scheme is

$$\rho_s = \frac{w(s)}{W_{str}}(1 - \pi_{in})(1 + W_{str}(1 - \pi_{in})(1 + \dots)) = \frac{w(s)}{W_{str}}(1 - \pi_{in})(1 + O(\varepsilon))$$

and in the deterministic scheme is  $\rho'_s = \frac{w(s)}{W_{str}}(1 - \pi_{in})$ . Thus  $\rho'_s$  is in the range  $[\rho_s(1 - O(\varepsilon)), \rho_s]$ . Adjusting the constant coefficients in  $\varepsilon$  elsewhere in the algorithm completes the proof. ◀

Now to generate the next independent point (which we need  $k$  of), we do not need to re-query with  $h$  or rebuild the alias structures. In particular, each candidate point can have a pointer back to the alias structure within its partition cell, so it can replace its representative candidate point. Moreover, since the points have weight proportional to their cell in the partition  $W_j$ , these weights do not change on a replacement. And more importantly, the points we never inspected to see if they belong to  $h$  do not need to be replaced. This deterministic process only inspects at most 2 points, and these can be replaced in  $O(1)$  time. Hence extending to  $k$  samples, only increases the total runtime by an additive term  $O(k)$ .

**Final bound.** We now have the ingredients for our final bound. In general the argument follows that of Lemma 3 except for a few changes. First, we allow items with probability total less than  $\gamma n$  to be ignored. This replaces a  $\log n$  factor in query time to be replaced with  $t = \log(n/\gamma)$  term. Second, we require  $O(t \log n)$  time to identify the relevant subset  $X'$  in each of  $t$  shallow-cutting structures. Then we select  $O(1/\varepsilon^3)$  candidate points from each of  $t$  weight partitions, in  $O(t/\varepsilon^3)$  time. Then pulling  $k$  independent samples, and refilling the candidates takes  $O(k)$  time. This results in the following final bound:

## 4:10 Independent Range Sampling, Revisited Again

► **Theorem 13.** *Let  $X$  be a set of  $n$  weighted points in  $\mathbb{R}^3$ , where the smallest weight is 1 and the largest weight is  $U$ . Choose  $0 < \gamma < \varepsilon \leq 1/2$ . We can store  $X$  in a data structure of size  $O(n \min\{\log n, \log \log_n U\})$  such that for any integer  $k$ , we can extract  $k$  independent random samples from the points  $(\varepsilon, \gamma)$ -approximately proportional to their weights in worst case time  $O(\log(n/\gamma)(\log n + 1/\varepsilon^3) + k)$*

In particular, if  $\varepsilon, \gamma = \Omega(1)$ , then the worst case time is  $O(\log^2 n + k)$  matching the expected time algorithm to sample by the exact weights.

### 4 Lower Bound for Worst-Case Time with Exact Weights

In this section, we focus on proving a (conditional) lower bound for a data structure that can extract one random sample in  $Q(n)$  worst-case time using  $S(n)$  space. Our main result is that under a reasonably restricted model, the data structure must essentially solve an equivalent range searching problem in the “group model”. As a result, we get a conditional lower bound as this latter problem is conjectured to be difficult. As an example, our conditional lower bound suggests that halfspace range sampling queries would require that  $S(n)Q^3(n) = \Omega(n^3)$ , i.e., with near-linear space we can only expect to get close to  $O(n^{2/3})$  query time. This is in contrast to the case when we allow expected query time or approximate weights. As already shown, we can solve the same problem with  $O(n \log n)$  space and  $O(\log^2 n)$  query time which reveals a large polynomial gap between the expected and the worst-case variants of the problem. To our knowledge, this is the first time such a large gap appears in the range searching literature between the worst-case and expected query times.

**The Model of Computation.** We assume the input is a list  $X$  of  $n$  elements,  $x_1, \dots, x_n$ , where each element  $x_i$  is associated with a real-valued weight  $w(x_i)$ . We use the decision tree model of computation. We assume the data structure has three components: a preprocessing algorithm that builds the data structure, a data structure which is a set of  $S(n)$  stored real values, and finally a query algorithm that given a query  $q$  it returns an element sampled with the correct probability in  $Q(n)$  worst-case time. We allow no approximation: the query algorithm should return an element  $x$  with exactly  $w(x)/w(X)$  probability.

**The main bottleneck.** The challenge in obtaining our lower bound was understanding where the main computational bottleneck lied and formulating a plan of attack to exploit it. This turned out to be in the query algorithm. As a result, we place no restrictions on the storage, or the preprocessing part of the algorithm, an idea that initially sounds very counter intuitive. After giving the algorithm the input  $X$  together with the weight assignment  $w: X \rightarrow \mathbb{R}$ , the algorithm stores some  $S(n)$  values in its storage. Afterwards, we move to the query phase and this is where we would like to put reasonable limits. We give the query algorithm the query  $q$ . We allow the query algorithm access to a set of  $t$  real random numbers,  $R = \{r_1, \dots, r_t\}$ , generated uniformly in  $[0, 1]$ . We restrict the query algorithm to be a “binary decision tree”  $T$  but in a specialized format: each node  $v$  of  $T$  involves a comparison between some random number  $r_v \in R$  and a rational function  $f_v = g_v/G_v$  where  $g_v$  and  $G_v$  are  $n$ -variate polynomials of the input weights  $w_1, \dots, w_n$ . To be more precise, we assume the query algorithm can compute polynomials  $g_v$  and  $G_v$  (either using polynomials stored at the ancestors of  $v$  or using the values stored by the data structure). The query algorithm at node  $v$  computes the ratio  $g_v$  and  $G_v$  and compares it to the random value  $r_v \in \{r_1, \dots, r_t\}$ . If  $v$  is an internal node, then  $v$  will have two children  $u_1$  and  $u_2$  and the algorithm will proceed to  $u_1$  if  $f_v(w(x_1), \dots, w(x_n)) < r_v$  and to  $u_2$  if otherwise. If  $v$  is a

leaf node, then  $v$  will return (a fixed) element  $x_v \in X$ . Note that there is no restriction on the rational function  $f_v$ ; it could be of an arbitrary size or complexity. We call this the *separated Algebraic decision tree* (SAD) model since at each node, we have “separated” the random numbers from the rational function that involves the weights of the input elements; a more general model would be to allow a rational function of the weights  $w_i$  and the random numbers  $r_j$ . If we insist the polynomials  $g_v$  and  $G_v$  be linear (i.e., degree one), then we call the model *separated linear decision tree* model (SLD).

► **Theorem 14.** *Consider an algorithm for range sampling in the SAD model where the input is a list of  $n$  of elements,  $x_1, \dots, x_n$  together with a weight assignment  $w(x_i) \in \mathbb{R}$ , for each  $x_i$ . Assume that the query algorithm has a worst-case bound, i.e., the maximum depth of its decision tree  $T$  is bounded by a function  $d(n)$ . Then, for every query  $q$ , there exists a node  $v \in T$  such that  $G_v$  is divisible by the polynomial  $\sum_{p \in q} w(p)$ .*

**Proof.** Consider the query decision tree  $T$ . By our assumptions,  $T$  is a finite tree that involves some  $N(n)$  nodes and it has the maximum depth of  $d(n)$ . Each node  $v$  involves a comparison between some random number  $r_i$  and a rational function  $f_v(w(x_1), \dots, w(x_t))$ . To reduce clutter, we will simply write the rational function as  $f_v$ . Let  $\mathcal{P}$  be the set of all the rational functions stored at the nodes of  $T$ . Note that each unique rational function appears only once in  $\mathcal{P}$ . Consider the set  $\Delta_{\mathcal{P}} = \{f_1 - f_2 | f_1, f_2 \in \mathcal{P}\}$  which is the set of pairwise differences. As each unique rational function appears only once in  $\mathcal{P}$ , it follows that none of the rational functions in  $\Delta_{\mathcal{P}}$  is identical to zero. This in particular implies that we can find real values  $w_1, \dots, w_n$  such that none of the rational functions in  $\Delta_{\mathcal{P}}$  are zero on  $w_1, \dots, w_n$ . Thus, as these functions are continuous at the points  $(w_1, \dots, w_n)$ , it follows that we can find a real value  $\varepsilon > 0$  such that for every weight assignment  $w(x_i) \in [w_i, w_i + \varepsilon]$ , the rational functions in  $\Delta_{\mathcal{P}}$  have the same (none zero) sign. In the rest of the proof, we only focus on the weight assignment functions  $w$  with the property that  $w(x_i) \in [w_i, w_i + \varepsilon]$ .

Let  $\mathcal{U}$  be the unit cube in  $\mathbb{R}^t$ .  $\mathcal{U}$  denotes the total probability space that corresponds to the random variables  $r_1, \dots, r_t$ . Every point in  $\mathcal{U}$  corresponds to a possible value for the random variables  $r_1, \dots, r_t$ . Now consider one rational function  $f_v$  stored at a node  $v$  of  $T$ , and assume  $v$  involves a comparison between  $r_i$  and  $f_v$  and let  $u_1$  and  $u_2$  be the left and the right child of  $v$ . By our assumption, we follow the path to  $u_1$  if  $f_v \leq r_i$  but to  $u_2$  if otherwise. Observe that this is equivalent to partitioning  $\mathcal{U}$  into two regions by a hyperplane perpendicular to the  $i$ -th axis at point  $f_v$ . As a result, for every node  $v \in T$ , we can assign a rectangular subset of  $\mathcal{U}$  that denotes its *region* and it includes all the points  $(r_1, \dots, r_t)$  of  $\mathcal{U}$  such that we would reach node  $v$  if our random variables were sampled to be  $(r_1, \dots, r_t)$ .

The next observation is that we can assume the region of each node is defined by *fixed* rational functions. Consider the list of rational functions encountered on the path from  $v$  to the root of  $T$ . Assume among this list, the rational functions  $f_1, \dots, f_m$  are involved in comparisons with  $r_i$ . Clearly, the lower boundary of the region of  $v$  along the  $i$ -th dimension is  $\min\{f_1, \dots, f_m\}$  whereas its upper boundary is  $\max\{f_1, \dots, f_m\}$ . Now, observe that since we have assumed that each  $f_i - f_j$  has a fixed sign, it follows that these evaluate to a fixed rational function. Thus, let  $f_{i,v}$  (resp.  $F_{i,v}$ ) be the rational function that describes the lower (resp. upper) boundary of the region of  $v$  along the  $i$ -th dimension. Let  $f_{i,v} = a_{i,v}/b_{i,v}$  and  $F_{i,v} = A_{i,v}/B_{i,v}$  where  $a_{i,v}, b_{i,v}, A_{i,v}, B_{i,v}$  are some polynomials of  $w(x_1), \dots, w(x_n)$  stored in tree  $T$  (e.g.,  $b_{i,v}$  is equal to some  $G_u$  for some ancestor  $u$  of  $v$  and the same holds for  $B_{i,v}$ ).

The Lebesgue measure of the region of  $v$ ,  $\text{Vol}(v)$ , is thus defined by the rational function

$$\text{Vol}(v) = \prod_{i=1}^t (F_{i,v} - f_{i,v}) = \prod_{i=1}^t \left( \frac{A_{i,v}}{B_{i,v}} - \frac{a_{i,v}}{b_{i,v}} \right) = \prod_{i=1}^t \frac{A_{i,v}b_{i,v} - a_{i,v}B_{i,v}}{B_{i,v}b_{i,v}}$$

## 4:12 Independent Range Sampling, Revisited Again

$\text{Vol}(v)$  is the probability of reaching  $v$ . Consider a query  $q$  that contains  $k$  elements. W.l.o.g., let  $x_1, \dots, x_k$  be these  $k$  elements. Let  $v_1, \dots, v_\ell$  be the set of all the leaf nodes that return  $x_1$ . For  $x_1$  to have been sampled with correct probability we must have the following identity

$$\sum_{i=1}^{\ell} \text{Vol}(v_i) = \frac{w(x_1)}{w(x_1) + \dots + w(x_k)}.$$

Observe that since the polynomial  $w(x_1) + \dots + w(x_k)$  is irreducible, it follows that at least one of the polynomials  $b_i$  or  $B_i$  for some  $i$  has this polynomial as a factor. ◀

**Conditional Lower Bound.** Intuitively, our above theorem suggests that if we can perform range sampling in finite worst-case time, then we should also be able to find the total weight of the points in the query range – since the total weight  $\sum_{p \in q} w(p)$  is encoded in some rational function. This latter problem is conjectured to be difficult but obtaining provable good lower bounds remains elusive. This suggests that short of a breakthrough, we can only hope for a conditional lower bound. This is reinforced by this observation that an efficient data structure for range sum queries leads to an efficient data structure for range sampling.

► **Observation 15.** *Assume for any set of  $X$  of  $n$  elements each associated with a real-valued weight, we can build a data structure that uses  $S(n)$  space such that given a query range  $q$ , it can output the total weight of the elements in  $q$  in  $Q(n)$  time.*

*Then, we can extract  $k$  random samples from  $q$  by a data structure that uses  $O(S(n) \log n)$  space and has the query time of  $O(kQ(n) \log n)$ .*

**Proof.** Partition  $X$  into two equal-sized sets  $X_1$  and  $X_2$  and build the data structure for range sum on each. Then recurse on  $X_1$  and  $X_2$ . The total space complexity is  $O(S(n) \log n)$ . Given a query  $q$ , it suffices to show how to extract one sample. Using the range sum query  $q$  on  $X_1$  and  $X_2$ , we can know the exact value of  $w_1 = \sum_{x \in q \cap X_1} w(x)$  and  $w_2 = \sum_{x \in q \cap X_2} w(x)$ . Thus, we can recurse into  $X_1$  with probability  $w_1/(w_1 + w_2)$  and into  $X_2$  with  $w_2/(w_1 + w_2)$ . In  $O(\log n)$  recursion steps we find a random sample with query time  $O(Q(n) \log n)$ . ◀

This implies that in the SLD model, the range sampling problem in the worst-case is equivalent to the range sum problem, ignoring polylog factors. This has consequences for query ranges like halfspaces where the latter problem is conjectured to be hard.

► **Conjecture 16.** *For every integer  $n$ , there exists a set  $P$  of  $\Theta(n)$  points in  $\mathbb{R}^d$  with the following property: If for any function  $w: P \rightarrow \mathbb{R}$  given as input, we can build a data structure of size  $S(n)$  such that for any query halfspace  $h$ , it can return the value  $\sum_{p \in P \cap h} w(p)$  in  $Q(n)$  time, then, we must have  $S(n)Q^d(n) = \Omega(n^{d-o(1)})$ .*

► **Corollary 17.** *Assume Conjecture 1 holds for  $d = 3$ . Then, there exists an input set of  $n$  points in  $\mathbb{R}^3$  such that for any data structure that uses  $S(n)$  space and solves the range sampling problem where the query algorithm is a decision tree  $T$  with worst-case query time  $Q(n)$ , we must have  $S(n)Q^3(n) = \Omega(n^{3-o(1)})$ . Thus if space  $S(n)$  is near-linear, query time  $Q(n) = \Omega(n^{2/3-o(1)})$ .*

---

### References

- 1 Peyman Afshani and Jeff M. Phillips. Independent Range Sampling, Revisited Again, 2019. [arXiv:1903.08014](#).
- 2 Peyman Afshani and Zhewei Wei. Independent Range Sampling, Revisited. In *European Symposium on Algorithms*, pages 3:1–3:14, 2017.

- 3 Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. *Advances in Discrete and Computational Geometry*, pages 1–56, 1999.
- 4 Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42. ACM, 2013.
- 5 Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. Random sampling for histogram construction: How much is enough? In *ACM SIGMOD Record*, pages 436–447. ACM, 1998.
- 6 T Hagerup, K Mehlhorn, and JI Munro. Optimal algorithms for generating discrete random variables with changing distributions. *Lecture Notes in Computer Science*, 700:253–264, 1993.
- 7 Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. *ACM SIGMOD Record*, 26(2):171–182, 1997.
- 8 Xiaocheng Hu, Miao Qiao, and Yufei Tao. Independent range sampling. In *ACM Symposium on Principles of Database Systems*, pages 246–255, 2014.
- 9 Jiří Matoušek. Efficient partition trees. *Discrete & Computational Geometry*, 8(3):315–334, 1992.
- 10 Frank Olken. *Random sampling from databases*. PhD thesis, University of California at Berkeley, 1993.
- 11 Frank Olken and Doron Rotem. Random sampling from databases: a survey. *Statistics and Computing*, 5(1):25–42, 1995.
- 12 Alastair J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.



# An Efficient Algorithm for Generalized Polynomial Partitioning and Its Applications

**Pankaj K. Agarwal**


Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129 USA  
pankaj@cs.duke.edu

**Boris Aronov** 

Department of Computer Science and Engineering, Tandon School of Engineering,  
New York University, Brooklyn, NY 11201, USA  
boris.aronov@nyu.edu

**Esther Ezra**

Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel  
School of Math, Georgia Institute of Technology, Atlanta, Georgia 30332, USA  
eezra3@math.gatech.edu

**Joshua Zahl** 

Department of Mathematics, University of British Columbia, Vancouver, BC V6T 1Z2, Canada  
jzahl@math.ubc.ca

---

## Abstract

In 2015, Guth proved that if  $\mathcal{S}$  is a collection of  $n$   $g$ -dimensional semi-algebraic sets in  $\mathbb{R}^d$  and if  $D \geq 1$  is an integer, then there is a  $d$ -variate polynomial  $P$  of degree at most  $D$  so that each connected component of  $\mathbb{R}^d \setminus Z(P)$  intersects  $O(n/D^{d-g})$  sets from  $\mathcal{S}$ . Such a polynomial is called a *generalized partitioning polynomial*. We present a randomized algorithm that computes such polynomials efficiently – the expected running time of our algorithm is linear in  $|\mathcal{S}|$ . Our approach exploits the technique of *quantifier elimination* combined with that of  $\varepsilon$ -samples.

We present four applications of our result. The first is a data structure for answering point-enclosure queries among a family of semi-algebraic sets in  $\mathbb{R}^d$  in  $O(\log n)$  time, with storage complexity and expected preprocessing time of  $O(n^{d+\varepsilon})$ . The second is a data structure for answering range search queries with semi-algebraic ranges in  $O(\log n)$  time, with  $O(n^{t+\varepsilon})$  storage and expected preprocessing time, where  $t > 0$  is an integer that depends on  $d$  and the description complexity of the ranges. The third is a data structure for answering vertical ray-shooting queries among semi-algebraic sets in  $\mathbb{R}^d$  in  $O(\log^2 n)$  time, with  $O(n^{d+\varepsilon})$  storage and expected preprocessing time. The fourth is an efficient algorithm for cutting algebraic planar curves into pseudo-segments.

**2012 ACM Subject Classification** Mathematics of computing → Combinatorial algorithms; Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** Polynomial partitioning, quantifier elimination, semi-algebraic range spaces,  $\varepsilon$ -samples

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.5

**Related Version** <https://arxiv.org/abs/1812.10269>

**Funding** *Pankaj K. Agarwal*: P. Agarwal was supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by an ARO grant W911NF-15-1-0408, and by BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

*Boris Aronov*: B. Aronov was supported by NSF grants CCF-12-18791 and CCF-15-40656, and by grant 2014/170 from the US-Israel Binational Science Foundation.

*Esther Ezra*: E. Ezra was supported by NSF CAREER under grant CCF:AF 1553354 and by Grant 824/17 from the Israel Science Foundation.

*Joshua Zahl*: J. Zahl was supported by an NSERC Discovery grant.



© Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl;  
licensed under Creative Commons License CC-BY

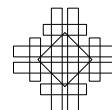
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 5; pp. 5:1–5:14

Leibniz International Proceedings in Informatics



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





## 1 Introduction

In 2015, Guth [11] proved that if  $\mathcal{S}$  is a collection of  $n$   $g$ -dimensional semi-algebraic sets<sup>1</sup> in  $\mathbb{R}^d$  and if  $D \geq 1$  is an integer, then there is a  $d$ -variate polynomial  $P$  of degree at most  $D$  so that each connected component of  $\mathbb{R}^d \setminus Z(P)$  intersects  $O(n/D^{d-g})$  sets from  $\mathcal{S}$ <sup>2</sup>, where the implicit constant in the  $O(\cdot)$  notation depends on  $d$  and on the degree and number of polynomials required to define each semi-algebraic set. We refer to such a polynomial  $P$  as a *generalized partitioning polynomial*. Guth’s proof established the existence of a generalized partitioning polynomial, but it did not offer an efficient algorithm to compute such a polynomial for a given collection of semi-algebraic sets. In this paper we study the problem of computing a generalized partitioning polynomial efficiently and present a few applications of such an algorithm.

**Related work.** In 2010, Guth and Katz [12] resolved the Erdős distinct distances problem in the plane. A major ingredient in their proof was a partitioning theorem for points in  $\mathbb{R}^d$ . Specifically, they proved that given a set of  $n$  points in  $\mathbb{R}^d$  and an integer  $D \geq 1$ , there is a  $d$ -variate *partitioning polynomial*  $P$  of degree at most  $D$  so that each connected component of  $\mathbb{R}^d \setminus Z(P)$  contains  $O(n/D^d)$  points from the set. Their polynomial partitioning theorem led to a flurry of new results in combinatorial and incidence geometry, harmonic analysis, and theoretical computer science.

The Guth-Katz result established the existence of a partitioning polynomial, but it did not provide an efficient algorithm to compute such a polynomial. In [2], Agarwal, Matoušek, and Sharir developed an efficient algorithm to compute partitioning polynomials, matching the degree bound obtained in [12] up to a constant factor. They used this algorithm to construct a linear-size data structure that can answer semialgebraic range queries amid a set of  $n$  points in  $\mathbb{R}^d$  in time  $O(n^{1-1/d} \text{polylog}(n))$ , which is near optimal.

A major open question in geometric searching is whether a semialgebraic range query can be answered in  $O(\log n)$  time using a data structure of size roughly  $n^d$ ; such a data structure is known only in very special cases, e.g., when query ranges are simplices [3]. If  $t$  is the number of (real valued) parameters needed to represent query ranges, then the best known data structure for semialgebraic range searching uses  $O(n^{t+\varepsilon})$  space for  $t \leq 4$  and  $O(n^{2t-4+\varepsilon})$  space for  $t > 4$  [3]. As we show below, an efficient algorithm for computing a generalized partitioning polynomial leads to a semialgebraic range searching data structure with  $O(\log n)$  query time and  $O(n^{t+\varepsilon})$  space.

In [4], the last three authors developed an efficient algorithm for constructing a partition of  $\mathbb{R}^3$  adapted to a set of space curves. This partition is not given by a partitioning polynomial, but it shares many of the same properties. For other settings, however, no effective method is known for computing a partitioning polynomial.

**Our results.** Our main result is an efficient algorithm for computing a generalized partitioning polynomial for a family of semi-algebraic sets (Theorem 11): Given a set  $\mathcal{S}$  of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$  for some constant  $b > 0$  (see Section 2 for the definition of the complexity of a semi-algebraic set), our algorithm computes a generalized partitioning polynomial of given degree  $D$  in expected time  $O(ne^{\text{Poly}(D)})$ .

<sup>1</sup> Roughly speaking, a semi-algebraic set in  $\mathbb{R}^d$  is the set of points in  $\mathbb{R}^d$  that satisfy a Boolean formula over a set of polynomial inequalities. See [8, Chapter 2] for formal definitions of a semialgebraic set and its dimension.

<sup>2</sup> Guth stated his result for the special case where the semi-algebraic sets are real algebraic varieties, but his proof in fact holds in the more general setting of semi-algebraic sets.



In Section 4 we present four applications of our algorithm:

- (i) Let  $\mathcal{S}$  be a family of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$  for some constant  $b > 0$ . Each set in  $\mathcal{S}$  is assigned a weight that belongs to a semigroup. We present a data structure of size  $O(n^{d+\varepsilon})$ , for any constant  $\varepsilon > 0$ , that can compute, in  $O(\log n)$  time, the cumulative weight of the sets in  $\mathcal{S}$  containing a query point. We refer to the latter as a *point-enclosure* query. The data structure can be constructed in  $O(n^{d+\varepsilon})$  randomized expected time. This is a significant improvement over the best known data structure by Koltun [14], for  $d > 4$ , that uses  $O(n^{2d-4+\varepsilon})$  space.
- (ii) Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ , each of which is assigned a weight in a semigroup, and let  $\mathcal{R}$  be a (possibly infinite) family of semi-algebraic sets in  $\mathbb{R}^d$ . Suppose that there exists a positive integer  $t$  and an injection  $f: \mathcal{R} \rightarrow \mathbb{R}^t$  so that for each  $p \in P$ , the set  $f(\{R \in \mathcal{R} \mid p \in R\})$  is a semi-algebraic set in  $\mathbb{R}^t$  of complexity at most  $b$ . We can construct in  $O(n^{t+\varepsilon})$  randomized expected time a data structure of size  $O(n^{t+\varepsilon})$ , for any constant  $\varepsilon > 0$ , that can compute in  $O(\log n)$  time the cumulative weight of  $P \cap \gamma$  for a query range  $\gamma \in \mathcal{R}$ .
- (iii) Given a family  $\mathcal{S}$  of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , we present a data structure of size  $O(n^{d+\varepsilon})$ , for any constant  $\varepsilon > 0$ , that can answer vertical ray shooting queries in  $O(\log^2 n)$  time. The data structure can be constructed in  $O(n^{d+\varepsilon})$  randomized expected time.
- (iv) Finally, we follow the technique of Sharir and Zahl [15] to cut  $n$  algebraic planar curves into a collection of  $O(n^{3/2+\varepsilon})$  *pseudo-segments* (that is, a collection of Jordan arcs, each pair of which intersects at most once), where the constant of proportionality depends on the degree of the curves. By exploiting Theorem 11, we show that this collection can be constructed in comparable time bound. We comment that applying the algorithm in the previous work of Aronov *et al.* [4] results in the same asymptotic bound on the number of pseudo-segments, however, its running time is quadratic in the number of curves. Thus our algorithm yields a considerable improvement in the running time.

Throughout this paper we assume that  $d, b$ , and  $\varepsilon$  are constants. Whenever big-O notation is used, the implicit constant may depend on  $d, b$ , and  $\varepsilon$ .

## 2 Preliminaries

In what follows, the *complexity* of a semi-algebraic set  $S$  in  $\mathbb{R}^d$  is the minimum value  $b$  so that  $S$  can be represented as the set of points  $x \in \mathbb{R}^d$  satisfying a Boolean formula with at most  $b$  atoms of the form  $P(x) < 0$  or  $P(x) = 0$ , with each  $P$  being a  $d$ -variate polynomial of degree at most  $b$ .

Our analysis makes extensive use of concepts and results from real algebraic geometry and random sampling. We review them below.

### 2.1 Polynomials, partitioning, and quantifier elimination

**Sign conditions.** Let  $\mathbb{R}[x_1, \dots, x_d]$  denote the set of all  $d$ -variate polynomials with real coefficients. For  $P_1, \dots, P_\ell \in \mathbb{R}[x_1, \dots, x_d]$ , a *sign condition* on  $(P_1, \dots, P_\ell)$  is an element of  $\{-1, 0, 1\}^\ell$ . A *strict sign condition* on  $(P_1, \dots, P_\ell)$  is an element of  $\{-1, 1\}^\ell$ . A sign condition  $(\nu_1, \dots, \nu_\ell) \in \{-1, 0, 1\}^\ell$  is *realizable* if the set

$$\{x \in \mathbb{R}^d \mid \text{sign}(P_j(x)) = \nu_j \text{ for each } j = 1, \dots, \ell\} \quad (1)$$

## 5:4 An Efficient Algorithm for Generalized Polynomial Partitioning

is non-empty. A realizable strict sign condition is defined analogously. The set (1) is called the *realization* of the sign condition. The set of *realizations of sign conditions* (resp., *realizations of strict sign conditions*) corresponding to the tuple  $(P_1, \dots, P_\ell)$  is the collection of all non-empty sets of the above form. These sets are pairwise disjoint and partition  $\mathbb{R}^d$ , by definition.

While a tuple of  $\ell$  polynomials has  $3^\ell$  sign conditions and  $2^\ell$  strict sign conditions, Milnor and Thom (see, e.g., [6, 10]) showed that any  $\ell$  polynomials in  $\mathbb{R}[x_1, \dots, x_d]$  of degree at most  $s$  (with  $2 \leq d \leq \ell$ ) have at most  $(50s\ell/d)^d$  realizable sign conditions.

**Polynomials and partitioning.** The set of polynomials in  $\mathbb{R}[x_1, \dots, x_d]$  of degree at most  $b$  is a real vector space of dimension  $\binom{b+d}{d}$ ; we identify this vector space with  $\mathbb{R}^{\binom{b+d}{d}}$ . For a point  $q \in \mathbb{R}^{\binom{b+d}{d}}$ , let  $P_q \in \mathbb{R}[x_1, \dots, x_d]$  be the corresponding polynomial of degree at most  $b$ .

► **Remark 1.** Consider the polynomial  $Q(q, x) \in \mathbb{R}[q_1, \dots, q_{\binom{b+d}{d}}, x_1, \dots, x_d]$  given by  $Q(q, x) := P_q(x)$ . Since we can write  $Q(q, x) = \sum_{i=1}^{\binom{b+d}{d}} q_i H_i(x)$ , where  $H_i$  is a monomial of degree at most  $b$ ,  $Q$  has degree  $b+1$ .

For each positive integer  $j$ , let  $D_j$  be the smallest positive integer so that  $\binom{D_j+d}{d} > 2^{j-1}$ ; we have  $D_j = O(2^{j/d})$ . Let  $k$  be a positive integer. For each  $j = 1, \dots, k$ , pick a  $2^{j-1}$ -dimensional subspace  $\mathbb{V}_j$  of  $\mathbb{R}^{\binom{D_j+d}{d}}$ . These subspaces  $\mathbb{V}_j$  will be fixed hereafter. Define the product space

$$\mathbb{Y}_k := \prod_{j=1}^k \mathbb{V}_j. \quad (2)$$

We identify each point  $y = (y_1, \dots, y_k) \in \mathbb{Y}_k$ , where  $y_j \in \mathbb{V}_j$ , with a  $k$ -tuple of polynomials  $\mathbf{P}_y = (P_{y_1}, \dots, P_{y_k})$ . For each  $j = 1, \dots, k$ ,  $\deg(P_j) = O(2^{j/d})$  and thus  $\deg(\prod_{j=1}^k P_j) = O(2^{k/d})$ .

Let  $\mathbf{P}_y \in \mathbb{Y}_k$ , let  $\mathcal{S}$  be a collection of semi-algebraic sets in  $\mathbb{R}^d$ , and let  $\alpha \geq 1$ . We say that  $\mathbf{P}_y$  is a  $(k, \alpha)$ -*partitioning tuple* for  $\mathcal{S}$  if  $\mathbf{P}_y$  has exactly  $2^k$  realizable strict sign conditions and the realization of each of them intersects at most  $|\mathcal{S}|/\alpha$  sets from  $\mathcal{S}$ .<sup>3</sup> Guth [11] proved that, for an appropriate choice of  $\alpha$ , a  $(k, \alpha)$ -partitioning tuple is guaranteed to exist:

► **Proposition 2** (Generalized Polynomial Partitioning [11]). *Let  $\mathcal{S}$  be a family of semi-algebraic sets in  $\mathbb{R}^d$ , each of dimension at most  $g$  and complexity at most  $b$ . For each  $k \geq 1$ , there exists a  $(k, \alpha)$ -partitioning tuple for  $\mathcal{S}$ , with  $\alpha = \Omega_{b,d}(2^{k(1-g/d)})$ .*

We also recall Theorem 2.16 from [7]:

► **Proposition 3.** *Let  $\mathcal{P}$  be a set of  $s$  polynomials in  $\mathbb{R}[x_1, \dots, x_d]$  of degree at most  $t$ . Then there is an algorithm that computes a set of points meeting every connected component of every realizable sign condition on  $\mathcal{P}$  in time  $O(s^d t^{O(d)})$ . There is also an algorithm providing the list of signs of all the polynomials of  $\mathcal{P}$  at each of these points in time  $O(s^{d+1} t^{O(d)})$ .*

<sup>3</sup> As in [11], we work with a  $k$ -tuple of polynomials instead of a single polynomial so that we can bound the number of sets intersected by the realization of a sign condition rather than by a connected component of a realization.

**Singly exponential quantifier elimination.** Let  $h$  and  $\ell$  be non-negative integers and let  $\mathcal{P} = \{P_1, \dots, P_s\} \subset \mathbb{R}[x_1, \dots, x_h, y_1, \dots, y_\ell]$ . Let  $\Phi(y)$  be a first-order formula given by

$$(\exists x_1, \dots, x_h)F(P_1(x, y), \dots, P_s(x, y)), \tag{3}$$

where  $y = (y_1, \dots, y_\ell)$  is a block of  $\ell$  free variables;  $x$  is a block of  $h$  variables, and  $F(P_1, \dots, P_s)$  is a quantifier-free Boolean formula with atomic predicates of the form  $\text{sign}(P_i(x_1, \dots, x_h, y)) = \sigma$ , with  $\sigma \in \{-1, 0, 1\}$ .

The Tarski-Seidenberg theorem states that the set of points  $y \in \mathbb{R}^\ell$  satisfying the formula  $\Phi(y)$  is semi-algebraic. The next proposition is a quantitative version of this result that bounds the number and degree of the polynomial equalities and inequalities needed to describe the set of points satisfying  $\Phi(y)$ . This proposition is known as a “singly exponential quantifier elimination,” and its more general form (where  $\Phi(y)$  may contain a mix of  $\forall$  and  $\exists$  quantifiers) can be found in [7, Theorem 2.27].

► **Proposition 4.** *Let  $\mathcal{P}$  be a set of at most  $s$  polynomials, each of degree at most  $t$  in  $h + \ell$  real variables. Given a formula  $\Phi(y)$  of the form (3), there exists an equivalent quantifier-free formula*

$$\Psi(y) = \bigvee_{i=1}^I \bigwedge_{j=1}^{J_i} \left( \bigvee_{n=1}^{N_{i,j}} \text{sign}(P_{ijn}(y)) = \sigma_{ijn} \right), \tag{4}$$

where  $P_{ijn}$  are polynomials in the variables  $y$ ,  $\sigma_{ijn} \in \{-1, 0, 1\}$ ,

$$\begin{aligned} I &\leq s^{(h+1)(\ell+1)} t^{O(h \cdot \ell)}, \\ J_i &\leq s^{(h+1)} t^{O(h)}, \\ N_{ij} &\leq t^{O(h)}, \end{aligned} \tag{5}$$

and the degrees of the polynomials  $P_{ijn}(y)$  are bounded by  $t^{O(h)}$ . Moreover, there is an algorithm to compute  $\Psi(Y)$  in time  $s^{(h+1)(\ell+1)} t^{O(h \cdot \ell)}$ .

## 2.2 Range spaces, VC dimension, and $\varepsilon$ -samples

We first recall several standard definitions and results from [13, Chapter 5]. A *range space* is a pair  $\Sigma = (X, \mathcal{R})$ , where  $X$  is a set and  $\mathcal{R}$  is a collection of subsets of  $X$ . Let  $(X, \mathcal{R})$  be a range space and let  $A \subset X$  be a set. We define the *restriction* of  $\Sigma$  to  $A$ , denoted by  $\Sigma_A$  to be  $(A, \mathcal{R}_A)$ , where  $\mathcal{R}_A := \{R \cap A \mid R \in \mathcal{R}\}$ . If  $A$  is finite, then  $|\mathcal{R}_A| \leq 2^{|A|}$ . If equality holds, then we say  $A$  is *shattered*. We define the *shatter function* by  $\pi_{\mathcal{R}}(z) := \max_{|A|=z} |\mathcal{R}_A|$ . The *VC dimension* of  $\Sigma$  is the largest cardinality of a set shattered by  $\mathcal{R}$ . If arbitrarily large finite subsets can be shattered, we say that the VC dimension of  $\Sigma$  is infinite.

Let  $\Sigma$  be a range space,  $A$  a finite subset of  $X$ , and  $0 \leq \varepsilon \leq 1$ . A set  $B \subset A$  is an  $\varepsilon$ -*sample* (also known as  $\varepsilon$ -*approximation*) of  $\Sigma_A$  if

$$\left| \frac{|A \cap R|}{|A|} - \frac{|B \cap R|}{|B|} \right| \leq \varepsilon \quad \forall R \in \mathcal{R}.$$

The following classical theorem of Vapnik and Chervonenkis [16] guarantees that, if the VC-dimension of  $\Sigma$  is finite, then for each positive  $\varepsilon > 0$ , a sufficiently large random sample of  $A$  is likely to be an  $\varepsilon$ -sample.<sup>4</sup>

<sup>4</sup> The stated bound is not the strongest possible (see, e.g., [13, Chapter 7] for an improved bound), but is sufficient for our purposes.

► **Proposition 5** ( $\varepsilon$ -Sample Theorem). *Let  $\Sigma = (X, \mathcal{R})$  be a range space of VC dimension at most  $d$  and let  $A \subset X$  be finite. Let  $0 < \varepsilon, \delta < 1$ . Then a random subset  $B \subset A$  of cardinality  $\frac{8d}{\varepsilon^2} \log \frac{1}{\varepsilon\delta}$  is an  $\varepsilon$ -sample for  $\Sigma_A$  with probability at least  $1 - \delta$ .*

► **Proposition 6** ([10, 13]). *Let  $\Sigma = (X, \mathcal{R})$  be a range space whose shatter function  $\pi_{\mathcal{R}}(z)$  satisfies the bound  $\pi_{\mathcal{R}}(z) \leq Cz^\rho$ , for all positive integers  $z$ , where  $\rho > 0$  is a real parameter. Then  $\Sigma$  has VC dimension at most  $4\rho \log(C\rho)$ .*

The following theorem can be proven by an argument closely following that in the proof of Corollary 2.3 from [10]; see [1] for details.

► **Theorem 7.** *Let  $Z \subset \mathbb{R}^d \times \mathbb{R}^\ell$  be a semi-algebraic set of complexity  $b$ . For each  $y \in \mathbb{R}^\ell$ , define  $R_y = \{x \in \mathbb{R}^d \mid (x, y) \in Z\}$ . Then the range space  $(\mathbb{R}^d, \{R_y \mid y \in \mathbb{R}^\ell\})$  has VC dimension at most  $200\ell^2 \log b$ .*

### 3 Computing Generalized Polynomial Partition

In this section we obtain the main result of the paper: given a collection  $\mathcal{S}$  of semi-algebraic sets in  $\mathbb{R}^d$ , each of dimension at most  $g$  and complexity at most  $b$ , a  $(k, \Omega_{b,d}(D^{d-g}))$ -partitioning tuple for  $\mathcal{S}$  can be computed efficiently. We obtain this result in several steps. Given a semialgebraic set  $S$ , a sign condition  $\sigma \in \{-1, +1\}^k$  and a real value  $b > 0$ , we first show that the set of  $k$ -tuples of degree- $b$  polynomials whose realization of  $\sigma$  intersects  $S$  is a semialgebraic set. This in turn implies that, if  $S_1, \dots, S_n$  are semi-algebraic sets and if  $m \leq n$ , then the set of  $k$ -tuples of degree- $b$  polynomials whose realization intersects at most  $m$  of the sets  $S_1, \dots, S_n$  is semi-algebraic. We use a quantifier-elimination algorithm to find a desired  $k$ -tuple. Unfortunately, the running time of the algorithm is exponential in  $n$ . We reduce the running time of the algorithm by using a random sampling technique – we show that it suffices to compute a partitioning tuple with respect to a small-size random subset of  $\mathcal{S}$ .

#### 3.1 The parameter space of semi-algebraic sets

Fix positive integers  $b, d, g$ , and  $k$ , and let  $D = 2^{k/d}$ . Hereafter we assume that  $D = \Omega(2^b)$ , which can be enforced by choosing  $k$  sufficiently large.

As above, let  $\mathcal{S}$  be a family of semi-algebraic sets in  $\mathbb{R}^d$ , each of dimension at most  $g$  and complexity at most  $b$ . Let  $G: \{0, 1\}^b \rightarrow \{0, 1\}$  be a Boolean function. Let  $\mathbb{X} = (\mathbb{R}^{\binom{b+d}{d}})^b$ . We identify a point  $x = (q_1, \dots, q_b) \in \mathbb{X}$  with the semi-algebraic set

$$Z_{x,G} = \{v \in \mathbb{R}^d \mid G(P_{q_1}(v) \geq 0, \dots, P_{q_b}(v) \geq 0) = 1\} \subset \mathbb{R}^d.$$

Observe that each semi-algebraic set in  $\mathcal{S}$  is of the form  $Z_{x,G}$  for some choice of  $x \in \mathbb{X}$  and a Boolean function  $G$ . Let  $\mathbb{Y} = \mathbb{Y}_k$ . For each  $y \in \mathbb{Y}$ , define  $S_y := \{u \in \mathbb{R}^d \mid P_1(u) > 0, \dots, P_k(u) > 0\}$ , where  $(P_1, \dots, P_k)$  is the tuple associated with  $y$ . Define

$$W_G := \{(x, y) \in \mathbb{X} \times \mathbb{Y} \mid Z_{x,G} \cap S_y \neq \emptyset\}.$$

► **Theorem 8.** *The set  $W_G$  is semi-algebraic; it is defined by  $O(e^{\text{Poly}(D)})$  polynomials, each of degree  $D^{O(d)}$ .*

**Proof.** Define  $V = \{(x, y, v) \in \mathbb{X} \times \mathbb{Y} \times \mathbb{R}^d \mid v \in Z_{x,G} \cap S_y\}$ . The condition  $v \in Z_{x,G}$  is a Boolean condition on  $b$  polynomial inequalities. By Remark 1, each of these polynomials has degree at most  $b+1$ . Similarly, the condition  $v \in S_y$  consists of  $k$  polynomial inequalities, each

of degree at most  $D+1$ . This means that there exists a set of polynomials  $\Omega = \{Q_1, \dots, Q_{b+k}\}$  of degree  $b + D + 1$  in the variables  $x, y, v$ , and a Boolean function  $F(z_1, \dots, z_{b+k})$  so that

$$V = \{(x, y, v) \in \mathbb{X} \times \mathbb{Y} \times \mathbb{R}^d \mid F(Q_1(x, y, v) \geq 0, \dots, Q_{b+k}(z, y, v) \geq 0) = 1\}.$$

With the above definitions

$$W_G = \{(x, y) \mid \exists(v_1, \dots, v_d) : F(Q_1(x, y, v), \dots, Q_{b+k}(x, y, v)) = 1\}.$$

We now apply Proposition 4. We have a set  $\Omega$  of  $s = b + k$  polynomials, each of degree at most  $t = b + D + 1$ . The variables  $h$  and  $\ell$  from the hypothesis of Proposition 4 are set to  $h = d$  and  $\ell = O\left(\binom{b+d}{d}^b + D^d\right) = \text{Poly}(D)$ ; recall that  $D$  is sufficiently larger than  $b$ , and thus  $\ell$  is a suitably chosen polynomial function of  $D$ . With these assignments, Proposition 4 says that  $W_G$  can be expressed as a quantifier-free formula of the form

$$\bigvee_{i=1}^I \bigwedge_{j=1}^{J_i} \left( \bigvee_{n=1}^{N_{i,j}} \text{sign}(P_{ijn}(x, y)) = \sigma_{ijn} \right), \tag{6}$$

where  $P_{ijn}$  are polynomials in the variables  $(x, y)$ ,  $\sigma_{ijn} \in \{-1, 0, 1\}$ ,

$$\begin{aligned} I &\leq (b+k)^{\text{Poly}(D)}(b+D)^{\text{Poly}(D)} = O(e^{\text{Poly}(D)}), \\ J_i &\leq (b+k)^{d+1}(b+D)^{O(d)} = O(D^{O(d)}), \\ N_{ij} &\leq (b+D)^{O(d)} = O(D^{O(d)}), \end{aligned} \tag{7}$$

where the degrees of the polynomials  $P_{ijn}(y)$  are bounded by  $(b+D)^{O(d)} = D^{O(d)}$ .

Summarizing, the quantifier-free formula (6) for  $W_G$  is a Boolean combination of  $O(e^{\text{Poly}(D)})$  polynomial inequalities, each of degree  $D^{O(d)}$ , as claimed. ◀

### 3.2 A singly-exponential algorithm

In this section, we discuss how to compute a  $(k, \alpha)$ -partitioning tuple (for an appropriate value of  $\alpha$ ) for a small number  $m$  of semi-algebraic sets.

▶ **Theorem 9.** *Let  $\mathcal{S}$  be a family of  $m$  semi-algebraic sets in  $\mathbb{R}^d$ , each of dimension at most  $g$  and complexity at most  $b$ . Let  $1 \leq k \leq \log m$  and let  $D = 2^{k/d}$ . Then a  $(k, \Omega_{b,d}(D^{d-g}))$ -partitioning tuple for  $\mathcal{S}$  can be computed in  $O(e^{\text{Poly}(m)})$  time.*

**Proof.** Set  $\mathbb{Y} = \mathbb{Y}_k$ . As above, we identify points in  $\mathbb{Y}$  with  $k$  tuples  $(P_1, \dots, P_k)$  of polynomials. The argument in Theorem 8, as well as the fact that the class of semi-algebraic sets is closed under the operation of taking a projection, show that for each  $S \in \mathcal{S}$  and each  $\sigma \in \{-1, 1\}^k$ ,

$$I_{S,\sigma} := \{y \in \mathbb{Y} \mid S \cap \{\sigma_1 P_1 > 0, \sigma_2 P_2 > 0, \dots, \sigma_k P_k > 0\} \neq \emptyset\}$$

is a semi-algebraic set in  $\mathbb{Y}$  that can be expressed as a Boolean combination of  $O(e^{\text{Poly}(D)})$  polynomials, each of degree  $D^{O(d)}$ . Moreover, it can be computed in time  $O(e^{\text{Poly}(D)})$  (see once again Proposition 4).

Let  $C_{b,d}$  be a constant to be specified later (the constant will depend only on  $b$  and  $d$ ) and let  $N = C_{b,d}|\mathcal{S}|D^{g-d} + 1$ ; observe that  $N = O(m)$ . For each  $\sigma \in \{-1, 1\}^k$  and for each set  $S' \subset \mathcal{S}$  of cardinality  $|S'| \geq N$ , the set  $\{y \in \mathbb{Y} \mid y \in I_{S,\sigma} \text{ for every } S \in S'\}$  is a semi-algebraic

## 5:8 An Efficient Algorithm for Generalized Polynomial Partitioning

set in  $\mathbb{Y}$  that can be expressed as a Boolean combination of  $O(N'e^{\text{Poly}(D)}) = O(me^{\text{Poly}(D)})$  polynomials, each of degree  $D^{O(d)}$ , where  $N' = |S'|$ . Therefore

$$\mathcal{K} := \bigcup_{\substack{S' \subset S \\ |S'| \geq N}} \{y \in \mathbb{Y} \mid y \in I_{S,\sigma} \text{ for every } S \in S'\} \quad (8)$$

is a semi-algebraic set in  $\mathbb{Y}$  that can be expressed as a Boolean combination of

$$\sum_{\substack{S' \subset S \\ |S'| \geq N}} O\left(\binom{m}{|S'|} me^{\text{Poly}(D)}\right) = O(e^{m+\text{Poly}(D)}) = O(e^{\text{Poly}(m)})$$

polynomials, each of degree  $D^{O(d)}$ . This and the fact that the class of semi-algebraic sets is closed under the operation of taking complement imply that

$$\text{Good}(\sigma) := \mathbb{Y} \setminus \mathcal{K} = \{y \in \mathbb{Y} \mid y \in I_{S,\sigma} \text{ for at most } C_{b,d}|\mathcal{S}|D^{g-d} \text{ sets } S \in \mathcal{S}\}$$

is a semi-algebraic set in  $\mathbb{Y}$  that can be expressed as a Boolean combination of  $O(e^{\text{Poly}(m)})$  polynomials, each of degree  $D^{O(d)}$ . This means that the set

$$\bigcap_{\sigma \in \{-1,1\}^k} \text{Good}(\sigma) \quad (9)$$

is a semi-algebraic set in  $\mathbb{Y}$  that can be expressed as a Boolean combination of  $O(e^{\text{Poly}(m)})$  polynomials, each of degree  $D^{O(d)}$ . Recall that by assumption  $1 \leq k \leq \log m$  and  $D = 2^{k/d}$ . It thus follows that the degree is bounded by  $\text{Poly}(m)$ . Similarly, the dimension of the space  $\mathbb{Y}$  is bounded by  $\text{Poly}(m)$  as well.

Proposition 2 guarantees that if  $C_{b,d}$  is selected sufficiently large, then the set (9) is non-empty. By Proposition 3, it is possible to locate a point in this set in  $O(e^{\text{Poly}(m)})$  time, concluding the proof of the theorem.  $\blacktriangleleft$

### 3.3 Speeding up the algorithm using $\varepsilon$ -sampling

In this section we first state the following lemma, whose proof is omitted (see the full version of this paper [1] for the details):

► **Lemma 10.** *For every choice of positive integers  $b$  and  $d$ , there is a constant  $C = C(b, d)$  so that the following holds. Let  $C_0$  be a positive integer. Let  $\mathcal{S}$  be a finite collection of semi-algebraic sets in  $\mathbb{R}^d$ , each of dimension at most  $g$  and complexity at most  $b$ . Let  $k$  be a positive integer and let  $D = 2^{k/d}$ . Let  $B \subset \mathcal{S}$  be a randomly chosen subset of  $\mathcal{S}$  of cardinality at least  $CD^C$  and let  $(P_1, \dots, P_k)$  be a  $(k, \frac{D^{d-g}}{C_0})$ -partitioning tuple for  $B$ . Then with probability at least  $1/2$ , each of the  $D^d$  realizable sign conditions of  $(P_1, \dots, P_k)$  intersects  $O(|\mathcal{S}|C_0D^{g-d})$  elements from  $\mathcal{S}$ .*

Note that Lemma 10 states that it is sufficient to consider a random subset  $B$  of size polynomial in  $D$  in order to obtain an appropriate partitioning tuple for the entire collection  $\mathcal{S}$ , with reasonable probability.

We next proceed as follows. We select a random sample of  $\mathcal{S}$  of cardinality  $CD^C$  and use Theorem 9 to compute the corresponding partitioning tuple  $(P_1, \dots, P_k)$ . This takes  $O(e^{\text{Poly}(D)})$  time. By Lemma 10, this tuple will be a  $(k, \Omega_{b,d}(D^{d-g}))$ -partitioning tuple for  $\mathcal{S}$  with probability at least  $1/2$ . We can verify whether the partitioning tuple works in

$O(|\mathcal{S}| \text{Poly}(D))$  time. If the tuple does not produce the appropriate partition, we discard it and try again; the expected number of trials is at most 2. The verification step is done as follows. For each semi-algebraic set  $S \in \mathcal{S}$  we compute the subset of sign conditions of  $(P_1, \dots, P_k)$ , with which it has a non-empty intersection. To this end, we restrict each of the polynomials  $P_1, \dots, P_k$  to  $S$  and apply Proposition 3 on this restricted collection, thereby obtaining a set of points meeting each connected component of each of the realizable sign conditions, as well as the corresponding list of signs of the restricted polynomials for each of these points. This is done in  $O(D^{O(d)})$  time for a single semi-algebraic set  $S \in \mathcal{S}$ , and overall  $O(|\mathcal{S}|D^{O(d)})$  time, over all sets. We refer the reader to [5] for further details concerning the complexity of the restriction of  $P_1, \dots, P_k$  to  $S$ . We have thus shown:

► **Theorem 11.** *Let  $\mathcal{S}$  be a finite collection of semi-algebraic sets in  $\mathbb{R}^d$ , each of which has dimension at most  $g$  and complexity at most  $b$ . Let  $k \geq 1$  and let  $D = 2^{k/d}$ . Then a  $(k, \Omega_{b,d}(D^{d-g}))$ -partitioning tuple for  $\mathcal{S}$  can be computed in  $O(|\mathcal{S}| \text{Poly}(D) + e^{\text{Poly}(D)})$  expected time by a randomized algorithm.*

## 4 Applications

In this section we describe a few applications of Theorem 11, namely, point-enclosure queries amid semi-algebraic sets, semi-algebraic range searching with logarithmic query time, vertical ray shooting amid semi-algebraic sets, and cutting algebraic curves into pseudo-segments.

### 4.1 Point-enclosure queries

Let  $\mathcal{S}$  be a set of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$ . Each set  $S$  is assigned a weight  $w(S)$ . We assume that the weights belong to a semigroup, i.e., subtractions are not allowed, and that the semigroup operation can be performed in constant time. We wish to preprocess  $\mathcal{S}$  into a data structure so that the cumulative weight of the sets in  $\mathcal{S}$  that contain a query point can be computed in  $O(\log n)$  time; we refer to this query as *point-enclosure* query. Note that if the weight of each set is 1 and the semi-group operation is Boolean  $\vee$ , then the point-enclosure query becomes a *union-membership query*: determine whether the query point lies in  $\bigcup \mathcal{S}$ .

We follow a standard hierarchical partitioning scheme of space, e.g., as in [9, 3], but use Theorem 11 at each stage. Using this hierarchical partition, we construct a tree data structure  $\mathcal{T}$  of depth  $O(\log n)$ , and a query is answered by following a path in  $\mathcal{T}$ . More precisely, we fix sufficiently large positive constants  $D = D(b, d)$  and  $n_0 = n_0(D)$ . If  $n \leq n_0$ ,  $\mathcal{T}$  consists of a single node that stores  $\mathcal{S}$  itself. So assume that  $n > n_0$ . Using Theorem 11, we construct a tuple  $\mathcal{P} = (P_1, \dots, P_k)$  of  $d$ -variate polynomials of degree at most  $D$ , which have  $2^k = O(D^d)$  realizable sign conditions, each of which with a realization that meets the boundaries of at most  $O(|\mathcal{S}|/D)$  sets of  $\mathcal{S}$ . For each realizable sign condition  $\sigma$ , let  $\mathcal{S}_\sigma \subseteq \mathcal{S}$  be the family of sets whose boundaries meet the realization of  $\sigma$ , and let  $\mathcal{S}_\sigma^* \subseteq \mathcal{S}$  be the family of sets that contain the realization of  $\sigma$ .

We compute  $\mathcal{S}_\sigma$ ,  $\mathcal{S}_\sigma^*$ , and  $W_\sigma = w(\mathcal{S}_\sigma^*)$ , as follows: We first apply Proposition 3 to  $\mathcal{P}$  to compute, in  $O(D^{O(d)})$  time, a representative point  $\xi_\sigma$  in the realization of every realizable sign condition  $\sigma$ . We fix a set  $S \in \mathcal{S}$ , mark every realizable sign condition  $\sigma$  that meets  $\partial S$ , and add  $S$  to the set  $\mathcal{S}_\sigma$ . This step is similar to the one described in the proof of Theorem 11, that is, we restrict each of the polynomials  $P_1, \dots, P_k$  to the algebraic varieties representing the boundary of  $S$  and apply Proposition 3 to this restricted collection. Each remaining sign condition  $\sigma$  is either contained in  $S$  or disjoint from it, which can be determined by testing whether its representing point  $\xi_\sigma$  is contained in  $S$ . If the answer is yes, we add the weight  $w(S)$  to  $W_\sigma$ . This task can be completed in overall  $O(nD^{O(d)})$  time over all sets of  $\mathcal{S}$ .



We create the root  $v$  of  $\mathcal{T}$  and store the tuple  $\mathcal{P}$  at  $v$ . We then create a child  $z_\sigma$  for each realizable sign condition  $\sigma$  and store  $\sigma$  and  $W_\sigma$  at  $z_\sigma$ . We recursively construct the data structure for each  $\mathcal{S}_\sigma$  and attach it to  $z_\sigma$  as its subtree. Since each node of  $\mathcal{T}$  has degree at most  $O(D^d)$  and the size of the subproblem reduces by a factor of  $D$  at each level of the recursion, a standard analysis shows that the total size of the data structure is  $O(n^{d+\varepsilon})$ , where  $\varepsilon > 0$  is a constant that can be made arbitrarily small by choosing  $D$  and  $n_0$  to be sufficiently large. Similarly, the expected preprocessing time is also  $O(n^{d+\varepsilon})$ .

Given a query point  $q \in \mathbb{R}^d$ , we compute the cumulative weight of the sets containing  $q$  by traversing a path in the tree in a top-down manner: We start from the root and maintain a partial weight  $W$ , which is initially set to 0. At each node  $v$ , we find the sign condition  $\sigma$  of the polynomial tuple at  $v$  whose realization contains  $q$ , add  $W_\sigma$  to  $W$ , and recursively query the child  $z_\sigma$  of  $v$ . The total query time is  $O(\log n)$ , where the constant of proportionality depends on  $D$  (and thus on  $\varepsilon$ ). Putting everything together, we obtain the following:

► **Theorem 12.** *Let  $\mathcal{S}$  be a set of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$  for some constant  $b > 0$ , and let  $w(S)$  be the weight of each set  $S \in \mathcal{S}$  that belongs to a semigroup. Assuming that the semigroup operation can be performed in constant time,  $\mathcal{S}$  can be preprocessed in  $O(n^{d+\varepsilon})$  randomized expected time into a data structure of size  $O(n^{d+\varepsilon})$ , for any constant  $\varepsilon > 0$ , so that the cumulative weight of the sets that contain a query point can be computed in  $O(\log n)$  time.*

## 4.2 Range searching

Next, we consider range searching with semi-algebraic sets: Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ . Each point  $p \in P$  is assigned a weight  $w(p)$  that belongs to a semigroup. Again we assume that the semigroup operation takes constant time. We wish to preprocess  $P$  so that, for a query range  $\gamma$ , represented as a semi-algebraic set in  $\mathbb{R}^d$ , the cumulative weight of  $\gamma \cap P$  can be computed in  $O(\log n)$  time. Here we assume that the query ranges (semi-algebraic sets) are parameterized as described in Section 3.1. That is, we have a fixed  $b$ -variate Boolean function  $G$ . A query range is represented as a point  $x \in \mathbb{X} = \mathbb{R}^t$ , for some  $t \leq \binom{b+d}{d}^b$ , and the underlying semi-algebraic set is  $Z_{x,G}$ . We refer to  $t$  as the *dimension of the query space*, and to the range searching problem in which all query ranges are of the form  $Z_{x,G}$  as  $(G, t)$ -semi-algebraic range searching.

For a point  $p \in \mathbb{R}^d$ , let  $S_p \subseteq \mathbb{X}$  denote the set of semi-algebraic sets  $Z_{x,G}$  that contain  $p$ , i.e.,  $S_p = \{x \in \mathbb{X} \mid p \in Z_{x,G}\}$ . It can be checked that  $S_p$  is a semi-algebraic set whose complexity depends only on  $b, d$ , and  $G$ . Let  $\mathcal{S} = \{S_p \mid p \in P\}$ . For a query range  $Z_{x,G}$ , we now wish to compute the cumulative weight of the sets in  $\mathcal{S}$  that contain  $x$ . This can be done using Theorem 12. Putting everything together, we obtain the following:

► **Theorem 13.** *Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ , let  $w(p)$  be the weight of  $p \in P$  that belongs to a semigroup, and let  $G$  be a fixed  $b$ -variate Boolean function for some constant  $b > 0$ . Let  $t \leq \binom{b+d}{d}^b$  be the dimension of the query space. Assuming that the semigroup operation can be performed in constant time,  $P$  can be preprocessed in  $O(n^{t+\varepsilon})$  randomized expected time into a data structure of size  $O(n^{t+\varepsilon})$ , for any constant  $\varepsilon > 0$ , so that a  $(G, t)$ -semi-algebraic range query can be answered in  $O(\log n)$  time.*

► **Remark.** If  $G$  is the conjunction of a set of  $b$  polynomial inequalities, then the size of the data structure can be significantly improved by using a multi-level data structure, with a slight increase in the query time to  $O(\log^b n)$ ; see, e.g., [3]. Roughly speaking, the value of  $t$  will now be the dimension of the parametric space of each polynomial defining the query semi-algebraic set, rather than the dimension of the parametric space of the entire semi-algebraic set (which is the conjunction of  $b$  such polynomials).



### 4.3 Vertical ray shooting

We next present an efficient data structure for answering *vertical ray-shooting* queries: Preprocess a collection  $\mathcal{S}$  of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$ , into a data structure so that the first set of  $\mathcal{S}$  hit by  $\rho_q$ , the ray emanating in the  $(+x_d)$ -direction from a query point  $q$ , can be reported quickly. If there is more than one such set, the query procedure returns one of them, arbitrarily.

**The overall data structure.** Our data structure for vertical ray shooting is similar to the one described in Section 4.1, except that we store an auxiliary data structure at each node of the tree to determine which of its children the query procedure should visit recursively.

Again we fix two constants  $D$  and  $n_0$ . If  $n \leq n_0$ , the tree data structure  $\mathcal{T}$  consists of a single node that stores  $\mathcal{S}$ . So assume that  $n > n_0$ . We compute a partitioning polynomial tuple  $\mathcal{P} = (P_1, \dots, P_k)$  for  $\mathcal{S}$  of degree  $D$ . For each realizable sign condition  $\sigma$ , we compute the set  $\mathcal{S}_\sigma \subseteq \mathcal{S}$  whose boundaries meet the realization of  $\sigma$ . We create the root node  $v$  of  $\mathcal{T}$  and create a child  $z_\sigma$  for each realizable sign condition  $\sigma$ . We store two auxiliary data structures  $\text{DS}_1(v)$  and  $\text{DS}_2(v)$  at  $v$ , described below, each of which can be constructed in  $O(n^{d+\varepsilon})$  randomized expected time and requires  $O(n^{d+\varepsilon})$  space. Given a query point  $q \in \mathbb{R}^d$ ,  $\text{DS}_1(v), \text{DS}_2(v)$  together determine, in  $O(\log n)$  time, the sign condition  $\sigma$  whose realization contains the first intersection point of  $\rho_q$  with a set of  $\mathcal{S}$ . We recursively construct the data structure for each subset  $\mathcal{S}_\sigma$  and attach it to  $z_\sigma$  as its subtree.

A standard analysis of multi-level data structures (see e.g. [3]) shows that the total size of  $\mathcal{T}$  is  $O(n^{d+\varepsilon})$ , for any constant  $\varepsilon > 0$ , and that it can be constructed in  $O(n^{d+\varepsilon})$  randomized expected time.

For a query point  $q \in \mathbb{R}^d$ , the first set hit by  $\rho_q$  can be computed by traversing a root-to-leaf path in  $\mathcal{T}$ . Suppose we are at a node  $v$ . If  $v$  is a leaf, then we naively check all sets in  $\mathcal{S}_v$  to find the first among them hit by  $\rho_q$ . Otherwise, we use the auxiliary data structures  $\text{DS}_1(v)$  and  $\text{DS}_2(v)$  to determine in  $O(\log n)$  time the sign condition  $\sigma$  whose realization contains the first intersection point of  $\rho_q$  and a set of  $\mathcal{S}$ . We recursively visit the child  $z_\sigma$  of  $v$ . Since the depth of  $\mathcal{T}$  is  $O(\log n)$ , the total query time is  $O(\log^2 n)$ .

This completes the description of the overall algorithm. What remains is to describe the auxiliary data structures  $\text{DS}_1, \text{DS}_2$ .

**Auxiliary data structures.** Recall that the auxiliary data structures are used to determine the sign condition of  $\mathcal{P}$  whose realization contains the first intersection point of a vertical ray with a set of  $\mathcal{S}$ . We first refine the realizations of sign conditions of  $\mathcal{P}$  into “cylindrical” cells, as follows. Let  $f = \prod_{i=1}^k P_i$ ; by construction, the degree of  $f$  is  $O(D)$ . By Warren [17, Theorem 2], the number of connected components of  $\mathbb{R}^d \setminus Z(f)$  is at most  $O(D)^d$ ; from now on we refer to these components as *cells*.<sup>5</sup> We refine the cells of  $\mathbb{R}^d \setminus Z(f)$  using the so-called *first-stage CAD (cylindrical algebraic decomposition)*; see, e.g., [6, Chapter 5] for a detailed overview of standard CAD. That is, this is a simplified version of CAD, presented in [2].

Roughly speaking, the first-stage CAD for  $f$  is obtained by constructing a collection  $\mathcal{G}$  of polynomials in the variables  $x_1, \dots, x_{d-1}$ , whose zero sets contain the projection onto  $\mathbb{R}^{d-1}$  of the set of points in  $Z(f)$  of vertical tangency, self-intersection of zeros sets (roots with multiplicity), or a singularity of some other kind. Having constructed  $\mathcal{G}$ , the first-stage CAD

<sup>5</sup> This notion is somewhat different than the notion of realizable sign conditions, where one can have several connected components representing the same sign condition.

is obtained as the arrangement  $\mathcal{A}(\{f\} \cup \mathcal{G})$  in  $\mathbb{R}^d$ , where the polynomials in  $\mathcal{G}$  are now regarded as  $d$ -variate polynomials, that is, we lift them in the  $x_d$ -direction; the geometric interpretation of the lifting operation is to erect a “vertical wall” in  $\mathbb{R}^d$  over each zero set within  $\mathbb{R}^{d-1}$  of a  $(d-1)$ -variate polynomial from  $\mathcal{G}$ , and the first-stage CAD is the arrangement of these vertical walls plus  $Z(f)$ . It follows by construction that the cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  are vertical stacks of “cylindrical” cells. In more detail, for each cell  $\tau$  of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ , there is unique cell  $\varphi$  of the  $(d-1)$ -dimensional arrangement  $\mathcal{A}(\mathcal{G})$  in  $\mathbb{R}^{d-1}$  such that one of the following two cases occur: (i)  $\tau = \{(x, \xi(x)) \mid x \in \varphi\}$ , where  $\xi: \varphi \rightarrow \mathbb{R}$  is a continuous semi-algebraic function (i.e.,  $\tau$  is the graph of  $\xi$  over  $\varphi$ ); or (ii)  $\tau = \{(x, t) \mid x \in \varphi, t \in (\xi_1(x), \xi_2(x))\}$ , where  $\xi_i, i = 1, 2$  is a continuous semi-algebraic function on  $\varphi$ , the constant function  $\varphi \rightarrow \{-\infty\}$ , or the constant function  $\varphi \rightarrow \{+\infty\}$ , and  $\xi_1(x) < \xi_2(x)$  for all  $x \in \varphi$  (i.e.,  $\tau$  is a cylindrical cell over  $\varphi$  bounded from below (resp. above) by the graph of  $\xi_1$  (resp.  $\xi_2$ )). As stated in [2], the total number of cells in  $\mathcal{A}(\{f\} \cup \mathcal{G})$  is  $D^{O(d)}$ , and each of them is a semi-algebraic set defined by  $D^{O(d^4)}$  polynomials of degree  $D^{O(d^3)}$  (this is, in fact, an application of Proposition 3).

For a cell  $\varphi$  of the  $(d-1)$ -dimensional arrangement  $\mathcal{A}(\mathcal{G})$ , let  $V(\varphi)$  be the stack of cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  over  $\varphi$ , i.e., the set of cells that project to  $\varphi$ .

We note that the sign condition of  $\mathcal{P}$  is the same for all points in a cell of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ , i.e., each cell lies in the realization of a single sign condition of  $\mathcal{P}$ . It thus suffices to find the cell of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  that contains the first intersection point of a vertical ray with a set of  $\mathcal{S}$  in order to find the sign condition of  $\mathcal{P}$  whose realization contains such a point. We construct  $\text{DS}_1, \text{DS}_2$  on the cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  to quickly determine the desired cell.

**The structure  $\text{DS}_1$ .** Fix a cell  $\tau$  of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ .  $\text{DS}_1$  is used to determine whether a query ray  $\rho_q$  whose source point lies in  $\tau$  intersects any set of  $\mathcal{S}$  inside  $\tau$ .

For each input set  $S \in \mathcal{S}$  that intersects  $\tau$ , let  $\uparrow S$  be the set of points  $x$  in  $\mathbb{R}^d$  such that the vertical ray  $\rho_x$  intersects  $S \cap \tau$ , i.e.,  $\uparrow(S)$  is the union of the rays in the  $(-x_d)$ -direction emanating from the points of  $S \cap \tau$ .  $\uparrow S$  is a semi-algebraic set whose complexity depends only on  $b, d$ , and  $D$ . Let  $\uparrow \mathcal{S}_\tau = \{\uparrow S \mid S \in \mathcal{S}, S \cap \tau \neq \emptyset\}$ .  $\uparrow \mathcal{S}_\tau$  can be computed in  $O(n)$  time. Using Theorem 12, we process  $\uparrow \mathcal{S}$  into a data structure  $\text{DS}_1(\tau)$  of size  $O(n^{d+\varepsilon})$  so that for a query point  $q \in \mathbb{R}^d$ , we can determine in  $O(\log n)$  time whether  $q \in \bigcup \uparrow \mathcal{S}$ , i.e., whether  $\rho_q$  intersects any set of  $\mathcal{S}$  within  $\tau$ . We construct  $\text{DS}_1(\tau)$  for all cells  $\tau$  of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ . The total size of the data structure, summed over all cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ , is  $O(n^{d+\varepsilon})$ , and it can be constructed in  $O(n^{d+\varepsilon})$  randomized expected time.

**The structure  $\text{DS}_2$ .** Fix a cell  $\tau$  of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ .  $\text{DS}_2$  is used to determine whether a line parallel to the  $x_d$ -axis intersects any set of  $\mathcal{S}$  inside  $\tau$ .

For each input set  $S \in \mathcal{S}$  that intersects  $\tau$ , let  $\downarrow S_\tau$  be the projection of  $S \cap \tau$  onto the hyperplane  $x_d = 0$ . For a point  $q \in \mathbb{R}^d$ , the vertical line (parallel to the  $x_d$ -axis) through  $q$  intersects  $S$  inside  $\tau$  if and only if  $\downarrow q \in \downarrow S_\tau$  (where  $\downarrow q$  is the projection of  $q$  onto the hyperplane  $x_d = 0$ ).  $\downarrow S_\tau$  is a  $(d-1)$ -dimensional semi-algebraic set whose complexity depends only on  $b$  and  $D$ . Let  $\downarrow \mathcal{S}_\tau = \{\downarrow S_\tau \mid S \in \mathcal{S}, S \cap \tau \neq \emptyset\}$ .  $\downarrow \mathcal{S}_\tau$  can be constructed in  $O(n)$  time. Using Theorem 12, we process  $\downarrow \mathcal{S}_\tau$  into a data structure  $\text{DS}_2(\tau)$  of size  $O(n^{d-1+\varepsilon})$  so that, for a query point  $q \in \mathbb{R}^d$ , we can determine in  $O(\log n)$  time whether  $\downarrow q \in \bigcup \downarrow \mathcal{S}_\tau$ , i.e., whether the vertical line through  $q$  intersects any set of  $\mathcal{S}$  inside  $\tau$ . We construct  $\text{DS}_2(\tau)$  for all cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ . The total size of the data structure, summed over all cells of  $\mathcal{A}(\{f\} \cup \mathcal{G})$ , is  $O(n^{d-1+\varepsilon})$ , and it can be constructed in  $O(n^{d-1+\varepsilon})$  randomized expected time.

**Answering a query.** Given a query point  $q \in \mathbb{R}^d$ , we determine the cell of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  that contains the first intersection point of  $\rho_q$  with a set of  $\mathcal{S}$  as follows. First, we determine the cell  $\tau$  of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  that contains the query point  $q$ . Using  $\text{DS}_1(\tau)$ , we determine in  $O(\log n)$

time whether  $\rho_q$  intersects  $\mathcal{S}$  inside  $\tau$ . If the answer is yes, then  $\tau$  is the desired cell. So assume that the answer is no. Let  $\varphi$  be the cell of the  $(d-1)$ -dimensional arrangement  $\mathcal{A}(\mathcal{G})$  such that  $\downarrow q \in \varphi$ . Let  $V(\varphi) = \langle \tau_1, \dots, \tau_r \rangle$  be the stack of cells over  $\varphi$ , and let  $\tau = \tau_i$  for some  $i \leq k$ . We visit the cells of  $V(\tau)$  one by one in order, starting from  $\tau_{i+1}$  until we find a cell  $\tau_j$  such that  $\downarrow q \in \bigcup \downarrow \mathcal{S}_{\tau_j}$ . Since  $q$  lies below  $\tau_j$ ,  $\rho_q$  intersects  $\mathcal{S}$  inside  $\tau_j$  if and only if  $\downarrow q \in \bigcup \downarrow (\mathcal{S}_{\tau_j})$ . If there is no such cell, we conclude that  $\rho_q$  does not intersect  $\mathcal{S}$ . Otherwise  $\tau_j$  is the cell of  $\mathcal{A}(\{f\} \cup \mathcal{G})$  that contains the first intersection point of  $\rho_q$  with a set of  $\mathcal{S}$ .

Putting everything together we obtain the following.

► **Theorem 14.** *Let  $\mathcal{S}$  be a collection of  $n$  semi-algebraic sets in  $\mathbb{R}^d$ , each of complexity at most  $b$  for some constant  $b > 1$ .  $\mathcal{S}$  can be preprocessed, in  $O(n^{d+\varepsilon})$  randomized expected time, into a data structure of size  $O(n^{d+\varepsilon})$ , for any constant  $\varepsilon > 0$ , so that a vertical ray-shooting query can be answered in  $O(\log^2 n)$  time.*

#### 4.4 Cutting algebraic curves into pseudo-segments

Sharir and Zahl [15] presented a technique for cutting algebraic plane curves into pseudo-segments, by lifting curves into three dimensions. More precisely, they prove that  $n$  non-overlapping algebraic curves of bounded degree  $d$  can be cut into  $O(n^{3/2} \log^{O(1)} n)$  Jordan arcs so that each pair of arcs intersect in at most one point. Their procedure exploits Proposition 2 for algebraic curves in three dimensions; see [15, Theorem 1.1]. Theorem 11 can be used to prove a slightly weaker constructive and efficient variant of the above result, for a sketch of the proof see the full version of this paper [1].

► **Theorem 15.** *Let  $\mathcal{C}$  be a set of  $n$  algebraic plane curves, each of degree at most  $d$ , with no two sharing a common component. Then  $\mathcal{C}$  can be cut into  $O(n^{3/2+C_d/\log \log n})$  Jordan arcs, where  $C_d$  is a constant that depends on  $d$ , so that each pair of arcs intersect in at most one point. This cutting can be computed in randomized expected time  $O(n^{3/2+C_d/\log \log n})$ .*

By replacing Theorem 1.1 in [15] with our Theorem 15, we obtain effective and efficient version of all of the subsequent results in [15].

---

#### References

- 1 Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl. An Efficient Algorithm for Generalized Polynomial Partitioning and Its Applications. *CoRR*, abs/1812.10269, 2018. [arXiv:1812.10269](#).
- 2 Pankaj K. Agarwal, Jivri'i Matoušek, and Micha Sharir. On Range Searching with Semialgebraic Sets. II. *SIAM J. Comput.*, 42(6):2039–2062, 2013. [doi:10.1137/120890855](#).
- 3 P.K. Agarwal. Simplex range searching and its variants: A review. *A Journey Through Discrete Mathematics*, pages 1–30, 2017.
- 4 Boris Aronov, Esther Ezra, and Joshua Zahl. Constructive Polynomial Partitioning for Algebraic Curves in  $\mathbb{R}^3$  with Applications. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2636–2648, 2019. [doi:10.1137/1.9781611975482.163](#).
- 5 Sal Barone and Saugata Basu. Refined Bounds on the Number of Connected Components of Sign Conditions on a Variety. *Discrete & Computational Geometry*, 47(3):577–597, 2012. [doi:10.1007/s00454-011-9391-3](#).
- 6 S. Basu, R. Pollack, and M.F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10. Springer Verlag, Berlin-Heidelberg, 2006.
- 7 Saugata Basu. Algorithms in Real Algebraic Geometry: A Survey. *CoRR*, abs/1409.1534, 2014. [arXiv:1409.1534](#).

- 8 J. Bochnak, M. Coste, and M.F. Roy. *Géométrie algébrique réelle (Second edition in english: Real Algebraic Geometry)*, volume 12(36). Springer Verlag, Berlin-Heidelberg, 1998.
- 9 Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, and Micha Sharir. A Singly Exponential Stratification Scheme for Real Semi-Algebraic Varieties and its Applications. *Theor. Comput. Sci.*, 84(1):77–105, 1991. doi:10.1016/0304-3975(91)90261-Y.
- 10 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A semi-algebraic version of Zarankiewicz’s problem. *J. Eur. Math. Soc.*, 19(6):1785–1810, 2017.
- 11 L. Guth. Polynomial partitioning for a set of varieties. *Math. Proc. Camb. Philos. Soc.*, 159(3):459–469, 2015.
- 12 L Guth and N. Katz. On the Erdos distinct distance problem in the plane. *Ann. of Math.*, 181:155–190, 2015.
- 13 S. Har-Peled. *Geometric Approximation Algorithms*, volume 173. AMS Press, Boston, MA, 2011.
- 14 Vladlen Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51(5):699–730, 2004. doi:10.1145/1017460.1017461.
- 15 Micha Sharir and Joshua Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Comb. Theory, Ser. A*, 150:1–35, 2017. doi:10.1016/j.jcta.2017.02.006.
- 16 V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.
- 17 H.E. Warren. Lower bound for approximation by nonlinear manifolds. *Trans. Amer. Math. Soc.*, 133:167–178, 1968.

# Efficient Algorithms for Geometric Partial Matching

**Pankaj K. Agarwal**

Duke University, Durham, USA  
pankaj@cs.duke.edu

**Hsien-Chih Chang**

Duke University, Durham, USA  
hsienchih.chang@duke.edu

**Allen Xiao**

Duke University, Durham, USA  
axiao@cs.duke.edu

---

## Abstract

Let  $A$  and  $B$  be two point sets in the plane of sizes  $r$  and  $n$  respectively (assume  $r \leq n$ ), and let  $k$  be a parameter. A matching between  $A$  and  $B$  is a family of pairs in  $A \times B$  so that any point of  $A \cup B$  appears in at most one pair. Given two positive integers  $p$  and  $q$ , we define the cost of matching  $M$  to be  $c(M) = \sum_{(a,b) \in M} \|a - b\|_p^q$  where  $\|\cdot\|_p$  is the  $L_p$ -norm. The geometric partial matching problem asks to find the minimum-cost size- $k$  matching between  $A$  and  $B$ .

We present efficient algorithms for geometric partial matching problem that work for any powers of  $L_p$ -norm matching objective: An exact algorithm that runs in  $O((n + k^2) \text{polylog } n)$  time, and a  $(1 + \varepsilon)$ -approximation algorithm that runs in  $O((n + k\sqrt{k}) \text{polylog } n \cdot \log \varepsilon^{-1})$  time. Both algorithms are based on the primal-dual flow augmentation scheme; the main improvements involve using dynamic data structures to achieve efficient flow augmentations. With similar techniques, we give an exact algorithm for the planar transportation problem running in  $O(\min\{n^2, rn^{3/2}\} \text{polylog } n)$  time.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** partial matching, transportation, optimal transport, minimum-cost flow, bichromatic closest pair

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.6

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1903.09358>.

**Funding** Work on this paper was supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by an ARO grant W911NF-15-1-0408, and by BSF Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

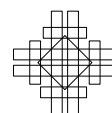
**Acknowledgements** We thank Haim Kaplan for discussion and suggestion to use Goldberg *et al.* [9] algorithm. We thank Debmalya Panigrahi for helpful discussions.

## 1 Introduction

Given two point sets  $A$  and  $B$  in the plane, we consider the problem of finding the minimum-cost partial matching between  $A$  and  $B$ . Formally, suppose  $A$  has size  $r$  and  $B$  has size  $n$  where  $r \leq n$ . Let  $G(A, B)$  be the undirected complete bipartite graph between  $A$  and  $B$ , and let the cost of edge  $(a, b)$  be  $c(a, b) = \|a - b\|_p^q$ , for some positive integers  $p$  and  $q$ . A *matching*  $M$  in  $G(A, B)$  is a set of edges sharing no endpoints. The *size* of  $M$  is the number of edges in  $M$ . The cost of matching  $M$ , denoted  $c(M)$ , is defined to be the sum of costs of edges in  $M$ . For a parameter  $k$ , the problem of finding the minimum-cost size- $k$  matching in



© Pankaj K. Agarwal, Hsien-Chih Chang, and Allen Xiao;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 6; pp. 6:1–6:14  
Leibniz International Proceedings in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



$G(A, B)$  is called the *geometric partial matching problem*. We call the corresponding problem in general bipartite graphs (with arbitrary edge costs) the *partial matching problem*.<sup>1</sup>

We also consider the following generalization of bipartite matching. Let  $\phi : A \cup B \rightarrow \mathbb{Z}$  be an integral *supply-demand function* with positive value on points of  $A$  and negative value on points of  $B$ , satisfying  $\sum_{a \in A} \phi(a) = -\sum_{b \in B} \phi(b)$ . Let  $U := \max_{p \in A \cup B} |\phi(p)|$ . A *transportation map* is a function  $\tau : A \times B \rightarrow \mathbb{R}_{\geq 0}$  such that  $\sum_{b \in B} \tau(a, b) = \phi(a)$  for all  $a \in A$  and  $\sum_{a \in A} \tau(a, b) = -\phi(b)$  for all  $b \in B$ . We define the cost of  $\tau$  to be

$$c(\tau) := \sum_{(a,b) \in A \times B} c(a, b) \cdot \tau(a, b).$$

The *transportation problem* asks to compute a transportation map of minimum cost.

**Related work.** Maximum-size bipartite matching is a classical problem in graph algorithms. Upper bounds include the  $O(m\sqrt{n})$  time algorithm by Hopcroft and Karp [10] and the  $O(m \min\{\sqrt{m}, n^{2/3}\})$  time algorithm by Even and Tarjan [7], where  $n$  is the number of nodes and  $m$  is the number of edges. The first improvement in over thirty years was made by Mądry [15], which uses an interior-point algorithm, runs in  $O(m^{10/7} \text{polylog } n)$  time.

The Hungarian algorithm [13] computes a minimum-cost maximum matching in a bipartite graph in roughly  $O(mn)$  time. Faster algorithms have been developed, such as the  $O(m\sqrt{n} \log(nC))$  time algorithms by Gabow and Tarjan [8] and the improved  $O(m\sqrt{n} \log C)$  time algorithm by Duan *et al.* [6] assuming the edge costs are integral; here  $C$  is the maximum cost of an edge. Ramshaw and Tarjan [17] showed that the Hungarian algorithm can be extended to compute a minimum-cost partial matching of size  $k$  in  $O(km + k^2 \log r)$  time, where  $r$  is the size of the smaller side of the bipartite graph. They also proposed a cost-scaling algorithm for partial matching that runs in time  $O(m\sqrt{k} \log(kC))$ , again assuming that costs are integral. By reduction to unit-capacity min-cost flow, Goldberg *et al.* [9] developed a cost-scaling algorithm for partial matching with an identical running time  $O(m\sqrt{k} \log(kC))$ , again only for integral edge costs.

In geometric settings, the Hungarian algorithm can be implemented to compute an optimal perfect matching between  $A$  and  $B$  (assuming equal size) in time  $O(n^2 \text{polylog } n)$  [11] (see also [21, 1]). This algorithm computes an optimal size- $k$  matching in time  $O(kn \text{polylog } n)$ . Faster approximation algorithms have been developed for computing perfect matchings in geometric settings [21, 22, 4, 19]. Recall that the cost of the edges are the  $q$ th power of their  $L_p$ -distances. When  $q = 1$ , the best algorithm to date by Sharathkumar and Agarwal [18] computes  $(1 + \varepsilon)$ -approximation to the value of optimal perfect matching in  $O(n \text{polylog } n \cdot \text{poly } \varepsilon^{-1})$  expected time with high probability. Their algorithm can also compute a  $(1 + \varepsilon)$ -approximate partial matching within the same time bound. For  $q > 1$ , the best known approximation algorithm to compute a perfect matching runs in  $O(n^{3/2} \text{polylog } n \log(1/\varepsilon))$  time [19]; it is not obvious how to extend this algorithm to the partial matching setting.

The transportation problem can also be formulated as an instance of the minimum-cost flow problem. The strongly polynomial uncapacitated min-cost flow algorithm by Orlin [16] solves the transportation problem in  $O((m + n \log n)n \log n)$  time. Lee and Sidford [14] give a weakly polynomial algorithm that runs in  $O(m\sqrt{n} \text{polylog}(n, U))$  time, where  $U$  is the maximum amount of node supply-demand. Agarwal *et al.* [2, 3] showed that Orlin's algorithm can be implemented to solve 2D transportation in time  $O(n^2 \text{polylog } n)$ . In case

<sup>1</sup> Partial matching is also called *imperfect matching* or *imperfect assignment* [17, 9].

of  $O(1)$ -dimension Euclidean space, by adapting the Lee-Sidford algorithm, they developed a  $(1 + \varepsilon)$ -approximation algorithm that runs in  $O(n^{3/2} \text{poly } \varepsilon^{-1} \text{polylog}(n, U))$  time. They also gave a Monte-Carlo algorithm that computes an  $O(\log^2(1/\varepsilon))$ -approximate solution in  $O(n^{1+\varepsilon})$  time with high probability. Recently, Khesin, Niklov, and Paramonov [12] obtained a  $(1 + \varepsilon)$ -approximation in low-dimensional Euclidean space that runs in randomized  $O(n \text{poly } \varepsilon^{-1} \text{polylog}(n, U))$  time.

**Our results.** There are three main results in this paper. First in Section 2 we present an efficient algorithm for computing an optimal partial matching in the plane.

► **Theorem 1.** *Given two point sets  $A$  and  $B$  in the plane each of size at most  $n$  and an integer  $k \leq n$ , a minimum-cost matching of size  $k$  between  $A$  and  $B$  can be computed in  $O((n + k^2) \text{polylog } n)$  time.*

We use *bichromatic closest pair (BCP)* data structures to implement the Hungarian algorithm efficiently, similar to Agarwal *et al.* [1] and Kaplan *et al.* [11]. But unlike their algorithms which take  $\Omega(n)$  time to find an augmenting path, we show that after  $O(n \text{polylog } n)$  time preprocessing, an augmenting path can be found in  $O(k \text{polylog } n)$  time. The key is to recycle (rather than rebuild) our data structures from one augmentation to the next. We refer to this idea as the *rewinding mechanism*.

Next in Sections 3, we obtain a  $(1 + \varepsilon)$ -approximation algorithm for the geometric partial matching problem in the plane by providing an efficient implementation of the unit-capacity min-cost flow algorithm by Goldberg *et al.* [9].

► **Theorem 2.** *Given two point sets  $A$  and  $B$  in  $\mathbb{R}^2$  each of size at most  $n$ , an integer  $k \leq n$ , and a parameter  $\varepsilon > 0$ , a  $(1 + \varepsilon)$ -approximate min-cost matching of size  $k$  between  $A$  and  $B$  can be computed in  $O((n + k\sqrt{k}) \text{polylog } n \cdot \log \varepsilon^{-1})$  time.*

The main challenge here is how to deal with the *dead nodes*, which neither have excess/deficit nor have flow passing through them, but still contribute to the size of the graph. We show that the number of *alive nodes* is only  $O(k)$ , and then represent the dead nodes implicitly so that the Hungarian search and computation of a blocking flow can be implemented in  $O(k \text{polylog } n)$  time.

Finally in Section 4 we present a faster algorithm for the transportation problem in  $\mathbb{R}^2$  when the two point sets are unbalanced.

► **Theorem 3.** *Given two point sets  $A$  and  $B$  in  $\mathbb{R}^2$  of sizes  $r$  and  $n$  respectively with  $r \leq n$ , along with supply-demand function  $\phi : A \cup B \rightarrow \mathbb{Z}$ , an optimal transportation map between  $A$  and  $B$  can be computed in  $O(\min\{n^2, rn^{3/2}\} \text{polylog } n)$  time.*

Our result improves over the  $O(n^2 \text{polylog } n)$  time algorithm by Agarwal *et al.* [3] for  $r = o(\sqrt{n})$ . Similar to their algorithm, we also use the strongly polynomial uncapacitated minimum-cost flow algorithm by Orlin [16], but additional ideas are needed for efficient implementation. Unlike in the case of matchings, the support of the transportation problem may have size  $\Omega(n)$  even when  $r$  is a constant; so naively we can no longer spend time proportional to the size of support of the transportation map. However, with careful implementation we ensure that the support is acyclic, and one can find an augmenting path in  $O(r\sqrt{n} \text{polylog } n)$  time with proper data structures, assuming  $r \leq \sqrt{n}$ .



## 2 Minimum-cost partial matchings using Hungarian algorithm

In this section, we solve the geometric partial matching problem and prove Theorem 1 by implementing the Hungarian algorithm for partial matching in  $O((n + k^2) \text{polylog } n)$  time.

A node  $v$  is *matched* by matching  $M$  if  $v$  is the endpoint of some edge in  $M$ ; otherwise  $v$  is *unmatched*. Given a matching  $M$ , an *augmenting path*  $\Pi = (a_1, b_1, \dots, a_\ell, b_\ell)$  is an odd-length path with unmatched endpoints ( $a_1$  and  $b_\ell$ ) that alternates between edges outside and inside of  $M$ . The symmetric difference  $M \oplus \Pi$  creates a new matching of size  $|M| + 1$ , called the *augmentation* of  $M$  by  $\Pi$ . The dual to the standard linear program for partial matching has one dual variable for each node  $v$ , called the *potential*  $\pi(v)$  of  $v$ . Given potential  $\pi$ , we can define the *reduced cost* of the edges to be  $c_\pi(v, w) := c(v, w) - \pi(v) + \pi(w)$ . Potential  $\pi$  is *feasible* on edge  $(v, w)$  if  $c_\pi(v, w)$  is nonnegative. Potential  $\pi$  is *feasible* if  $\pi$  is feasible on every edge in  $G$ . We say that an edge  $(v, w)$  is *admissible* under potential  $\pi$  if  $c_\pi(v, w) = 0$ .

**Fast implementation of Hungarian search.** The Hungarian algorithm is initialized with  $M \leftarrow \emptyset$  and  $\pi \leftarrow 0$ . Each iteration of the Hungarian algorithm augments  $M$  by an admissible augmenting path  $\Pi$ , discovered using a procedure called the *Hungarian search*. The algorithm terminates after  $k$  augmentations, exactly when  $|M| = k$ ; Ramshaw and Tarjan [17] showed that  $M$  is guaranteed to be an optimal partial matching.

The Hungarian search grows a set of *reachable nodes*  $X$  from all unmatched  $v \in A$  using augmenting paths of admissible edges. Initially,  $X$  is the set of unmatched nodes in  $A$ . Let the *frontier* of  $X$  be the edges in  $(A \cap X) \times (B \setminus X)$ .  $X$  is grown by *relaxing* an edge  $(a, b)$  in the frontier: Add  $b$  into  $X$ , modify potential to make  $(a, b)$  admissible, preserve  $c_\pi$  on other edges within  $X$ , and keep  $\pi$  feasible on edges outside of  $X$ . Specifically, the algorithm relaxes the min-reduced-cost frontier edge  $(a, b)$ , and then raises  $\pi(v)$  by  $c_\pi(a, b)$  for all  $v \in X$ . If  $b$  is already matched, then we also relax the matching edge  $(a', b)$  and add  $a'$  into  $X$ . The search finishes when  $b$  is unmatched, and an admissible augmenting path now can be recovered.

In the geometric setting, we find the min-reduced-cost frontier edge using a dynamic *bichromatic closest pair* (BCP) data structure, similar to [3, 21]. Given two point sets  $P$  and  $Q$  in the plane and a weight function  $\omega : P \cup Q \rightarrow \mathbb{R}$ , the BCP is two points  $a \in P$  and  $b \in Q$  minimizing the additively weighted distance  $c(a, b) - \omega(a) + \omega(b)$ . Thus, a minimum reduced-cost frontier edge is precisely the BCP of point sets  $P = A \cap X$  and  $Q = B \setminus X$ , with  $\omega = \pi$ . Note that the “state” of this BCP is parameterized by  $X$  and  $\pi$ .

The dynamic BCP data structure by Kaplan *et al.* [11] supports point insertions and deletions in  $O(\text{polylog } n)$  time and answers queries in  $O(\log^2 n)$  time for our setting. Each relaxation in the Hungarian search requires one query, one deletion, and at most one insertion (aside from the potential updates). As  $|M| \leq k$  throughout, there are at most  $2k$  relaxations in each Hungarian search, and the BCP can be used to implement each Hungarian search in  $O(k \text{polylog } n)$  time.

**Rewinding mechanism.** We observe that exactly one node of  $A$  is newly matched after an augmentation. Thus (modulo potential changes), we can obtain the initial state of the BCP for the  $(i + 1)$ -th Hungarian search from the  $i$ -th one with a single BCP deletion.

If we remember the sequence of points added to  $X$  in the  $i$ -th Hungarian search, then at the start of the  $(i + 1)$ -th Hungarian search we can *rewind* this sequence by applying the opposite insert/delete operation to each BCP update in reverse order to obtain the initial state of the  $i$ -th BCP. With one additional BCP deletion, we have the initial state of the  $(i + 1)$ -th BCP. The number of insertions/deletions is bounded by the number of relaxations



per Hungarian search which is  $O(k)$ . Therefore we can recover, in  $O(k \text{ polylog } n)$  time, the initial BCP data structure for each Hungarian search beyond the first. We refer to this procedure as the *rewinding mechanism*.

**Potential updates.** We modify a trick from Vaidya [21] to batch potential updates. Potential is tracked with a *stored value*  $\gamma(v)$ , while the *true value* of  $\pi(v)$  may have changed since  $\gamma(v)$  was last recorded. This is done by aggregating potential changes into a variable  $\delta$ , which is initially 0 at the very beginning of the algorithm. Whenever we would raise the potential of all nodes in  $X$ , we raise  $\delta$  by that amount instead. We maintain the following invariant:  $\pi(v) = \gamma(v)$  for  $v \notin X$ , and  $\pi(v) = \gamma(v) + \delta$  for  $v \in X$ .

At the beginning of the algorithm,  $X$  is empty and stored values are equal to true values. When  $a \in A$  is added to  $X$ , we update its stored value to  $\pi(a) - \delta$  for the current value of  $\delta$ , and use that stored value as its BCP weight. Since the BCP weights are uniformly offset from  $\pi(v)$  by  $\delta$ , the pair reported by the BCP is still minimum. When  $b \in B$  is added to  $X$ , we update its stored value to  $\pi(b) - \delta$  (although it won't be added to a BCP set). When a node is removed from  $X$  (e.g. by augmentation or rewinding), we update the stored potential  $\gamma(v) \leftarrow \pi(v) + \delta$ , again for the current value of  $\delta$ . Unlike Vaidya [21] and for the sake of rewinding, we do not reset  $\delta$  across Hungarian searches.

There are  $O(k)$  relaxations and thus  $O(k)$  updates to  $\delta$  per Hungarian search.  $O(k)$  stored values are updated per rewinding, so the time spent on potential updates per Hungarian search is  $O(k)$ . Putting everything together, our implementation of the Hungarian algorithm runs in  $O((n + k^2) \text{ polylog } n)$  time. This proves Theorem 1.

### 3 Approximating min-cost partial matching through cost-scaling

In this section we describe an approximation algorithm for computing a min-cost partial matching. We reduce the problem to computing a min-cost circulation in a flow network (Section 3.1). We adapt the cost-scaling algorithm by Goldberg *et al.* [9] for computing min-cost flow of a unit-capacity network (Section 3.2). Finally, we show how their algorithm can be implemented in  $O((n + k^{3/2}) \text{ polylog}(n) \log(1/\varepsilon))$  time in our setting (Section 3.3).

#### 3.1 From matching to circulation

Given a bipartite graph  $G$  with node sets  $A$  and  $B$ , we construct a flow network  $N = (V, \vec{E})$  in a standard way [17] so that a min-cost matching in  $G$  corresponds to a min-cost integral circulation in  $N$ .

**Flow network.** Each node in  $G$  becomes a node in  $N$  and each edge  $(a, b)$  in  $G$  becomes an arc  $a \rightarrow b$  in  $N$ ; we refer to these nodes and arcs as *bipartite nodes* and *bipartite arcs*. We also include a *source* node  $s$  and *sink* node  $t$  in  $N$ . For each  $a \in A$ , we add a *left dummy arc*  $s \rightarrow a$  and for each  $b \in B$  a *right dummy arc*  $b \rightarrow t$ . The cost  $c(v \rightarrow w)$  is equal to  $c(v, w)$  if  $v \rightarrow w$  is a bipartite arc and 0 if  $v \rightarrow w$  is a dummy arc. All arcs in  $N$  have unit capacity.

Let  $\phi : V \rightarrow \mathbb{Z}$  be an integral supply/demand function on nodes of  $N$ . The positive values of  $\phi(v)$  are referred to as *supply*, and the negative values of  $\phi(v)$  as *demand*. A *pseudoflow*  $f : \vec{E} \rightarrow [0, 1]$  is a function on arcs of  $N$ . The *support* of  $f$  in  $N$ , denoted as  $\text{supp}(f)$ , is the set of arcs with positive flow. Given a pseudoflow  $f$ , the *imbalance* of a node is

$$\phi_f(v) := \phi(v) + \sum_{w \rightarrow v \in \vec{E}} f(w \rightarrow v) - \sum_{v \rightarrow w \in \vec{E}} f(v \rightarrow w).$$

We call positive imbalance *excess* and negative imbalance *deficit*. A node is *balanced* if it has zero imbalance. If all nodes are balanced, the pseudoflow is a *circulation*. The *cost* of a pseudoflow is defined to be

$$c(f) := \sum_{v \rightarrow w \in \text{supp}(f)} c(v \rightarrow w) \cdot f(v \rightarrow w).$$

The *minimum-cost flow problem* (MCF) asks to find a circulation of minimum cost.

If we set  $\phi(s) = k$ ,  $\phi(t) = k$ , and  $\phi(v) = 0$  for all  $v \in A \cup B$ , then an integral circulation  $f$  corresponds to a partial matching  $M$  of size  $k$  and vice versa. Moreover,  $c(M) = c(f)$ . Hence, the problem of computing a min-cost matching of size  $k$  in  $G$  transforms to computing an integral circulation in  $N$ . The following lemma will be useful for our algorithm.

► **Lemma 4.** *Let  $N$  be the network constructed from the bipartite graph  $G$  above.*

- (i) *For any integral circulation  $g$  in  $N$ , the size of  $\text{supp}(g)$  is at most  $3k$ .*
- (ii) *For any integral pseudoflow  $f$  in  $N$  with  $O(k)$  excess, the size of  $\text{supp}(f)$  is  $O(k)$ .*

### 3.2 A cost-scaling algorithm

Before describing the algorithm, we need to introduce a few more concepts.

**Residual network and admissibility.** If  $f$  is an integral pseudoflow on  $N$  (that is,  $f(v \rightarrow w) \in \{0, 1\}$  for every arc in  $\vec{E}$ ), then each arc  $v \rightarrow w$  in  $N$  is either *idle* with  $f(v \rightarrow w) = 0$  or *saturated* with  $f(v \rightarrow w) = 1$ .

Given a pseudoflow  $f$ , the *residual network*  $N_f = (V, \vec{E}_f)$  is defined as follows. For each idle arc  $v \rightarrow w$  in  $\vec{E}$ , we add a *forward* residual arc  $v \rightarrow w$  in  $N_f$ . For each saturated arc  $v \rightarrow w$  in  $\vec{E}$ , we add a *backward* residual arc  $w \rightarrow v$  in  $N_f$ . The set of residual arcs in  $N_f$  is therefore

$$\vec{E}_f := \{v \rightarrow w \mid f(v \rightarrow w) = 0\} \cup \{w \rightarrow v \mid f(v \rightarrow w) = 1\}.$$

The cost of a forward residual arc  $v \rightarrow w$  is  $c(v \rightarrow w)$ , while the cost of a backward residual arc  $w \rightarrow v$  is  $-c(v \rightarrow w)$ . Each arc in  $N_f$  also has unit capacity. By Lemma 4,  $N_f$  has  $O(k)$  backward arcs if  $f$  has  $O(k)$  excess.

A *residual pseudoflow*  $g$  in  $N_f$  can be used to change  $f$  into a different pseudoflow on  $N$  by *augmentation*. For simplicity, we only describe augmentation for the case where  $f$  and  $g$  are integral. Specifically, augmenting  $f$  by  $g$  produces a pseudoflow  $f'$  in  $N$  where

$$f'(v \rightarrow w) = \begin{cases} 0 & w \rightarrow v \in \vec{E}_f \text{ and } g(w \rightarrow v) = 1, \\ 1 & v \rightarrow w \in \vec{E}_f \text{ and } g(v \rightarrow w) = 1, \\ f(v \rightarrow w) & \text{otherwise.} \end{cases}$$

Using LP duality for min-cost flow, we assign *potential*  $\pi(v)$  to each node  $v$  in  $N$ . The *reduced cost* of an arc  $v \rightarrow w$  in  $N$  with respect to  $\pi$  is defined as

$$c_\pi(v \rightarrow w) := c(v \rightarrow w) - \pi(v) + \pi(w).$$

Similarly we define the reduced cost of arcs in  $N_f$ : the reduced cost of a forward residual arc  $v \rightarrow w$  in  $N_f$  is  $c_\pi(v \rightarrow w)$ , and the reduced cost of a backward residual arc  $w \rightarrow v$  in  $N_f$  is  $-c_\pi(v \rightarrow w)$ . Abusing the notation, we also use  $c_\pi$  to denote the reduced cost of arcs in  $N_f$ .

The *dual feasibility constraint* asks that  $c_\pi(v \rightarrow w) \geq 0$  holds for every arc  $v \rightarrow w$  in  $\vec{E}$ ; potential  $\pi$  that satisfy this constraint is said to be *feasible*. Suppose we relax the dual feasibility constraint to allow some small violation in the value of  $c_\pi(v \rightarrow w)$ . We say that a

pair of pseudoflow  $f$  and potential  $\pi$  is  $\theta$ -optimal [20, 5] if  $c_\pi(v \rightarrow w) \geq -\theta$  for every residual arc  $v \rightarrow w$  in  $\vec{E}_f$ . Pseudoflow  $f$  is  $\theta$ -optimal if it is  $\theta$ -optimal with respect to some potential  $\pi$ ; potential  $\pi$  is  $\theta$ -optimal if it is  $\theta$ -optimal with respect to some pseudoflow  $f$ . Given a pseudoflow  $f$  and potential  $\pi$ , a residual arc  $v \rightarrow w$  in  $\vec{E}_f$  is *admissible* if  $c_\pi(v \rightarrow w) \leq 0$ . We say that a pseudoflow  $g$  in  $N_f$  is *admissible* if  $g(v \rightarrow w) > 0$  only on admissible arcs  $v \rightarrow w$ , and  $g(v \rightarrow w) = 0$  otherwise.<sup>2</sup> We will use the following well-known property of  $\theta$ -optimality.

► **Lemma 5.** *Let  $f$  be an  $\theta$ -optimal pseudoflow in  $N$  and let  $g$  be an admissible pseudoflow in  $N_f$ . Then  $f$  augmented by  $g$  is also  $\theta$ -optimal in  $N$ .*

Using Lemma 4, the following lemma can be proved about  $\theta$ -optimality:

► **Lemma 6.** *Let  $f$  be a  $\theta$ -optimal integer circulation in  $N$ , and  $f^*$  be an optimal integer circulation for  $N$ . Then,  $c(f) \leq c(f^*) + 6k\theta$ .*

**Estimating the value of  $c(f^*)$ .** We now describe a procedure for estimating  $c(f^*)$  within a polynomial factor, which is used to initialize the cost-scaling algorithm.

Let  $T$  be a minimum spanning tree of  $A \cup B$  under the cost function  $c$ . Let  $e_1, e_2, \dots, e_{n-1}$  be the edges of  $T$  sorted in nondecreasing order of length. Let  $T_i$  be the forest consisting of the nodes of  $A \cup B$  and edges  $e_1, \dots, e_i$ . We call a matching  $M$  *intra-cluster* if both endpoints of each edge in  $M$  lie in the same connected component of  $T_i$ . The following lemma will be used by our cost-scaling algorithm:

► **Lemma 7.** *Let  $A$  and  $B$  be two point sets in the plane. Define  $i^*$  to be the smallest index  $i$  such that there is an intra-cluster matching of size  $k$  in  $T_{i^*}$ . Set  $\bar{\theta} := n^q \cdot c(e_{i^*})$ . Then*

- (i) *The value of  $i^*$  can be computed in  $O(n \log n)$  time.*
- (ii)  *$c(e_{i^*}) \leq c(f^*) \leq \bar{\theta}$ .*
- (iii) *There is a  $\theta$ -optimal circulation in the network  $N$  with respect to the all-zero potential, assuming  $\phi(s) = k$ ,  $\phi(t) = -k$ , and  $\phi(v) = 0$  for all  $v \in A \cup B$ .*

As a consequence of Lemmas 7(ii) and 6, we have:

► **Corollary 8.** *The cost of a  $\underline{\theta}$ -optimal integral circulation in  $N$  is at most  $(1 + \varepsilon)c(f^*)$ , where  $\underline{\theta} := \frac{\varepsilon}{6k} \cdot c(e_{i^*})$ .*

**Overview of the algorithm.** We are now ready to describe our algorithm, which closely follows Goldberg *et al.* [9]. The algorithm works in rounds called *scales*. Within each scale, we fix a *cost scaling parameter*  $\theta$  and maintain potential  $\pi$  with the following property:

(\*) There exists a  $2\theta$ -optimal integral circulation in  $N$  with respect to  $\pi$ .

For the initial scale, we set  $\theta \leftarrow \bar{\theta}$  and  $\pi \leftarrow 0$ . By Lemma 7(iii), property (\*) is satisfied initially. Each scale of the algorithm consists of two stages. In the *scale initialization* stage, SCALE-INIT computes a  $\theta$ -optimal pseudoflow  $f$ . In the *refinement* stage, REFINES converts  $f$  into a  $\theta$ -optimal (integral) circulation  $g$ . In both stages,  $\pi$  is updated as necessary. If  $\theta \leq \underline{\theta}$ , we return  $g$ . Otherwise, we set  $\theta \leftarrow \theta/2$  and start the next scale. Note that property (\*) is satisfied in the beginning of each scale.

By Corollary 8, when the algorithm terminates, it returns an integral circulation  $\tilde{f}$  in  $N$  of cost at most  $(1 + \varepsilon)c(f^*)$ , which corresponds to a  $(1 + \varepsilon)$ -approximate min-cost matching of size  $k$  in  $G$ . The algorithm terminates in  $\log_2(\bar{\theta}/\underline{\theta}) = O(\log(n/\varepsilon))$  scales.

<sup>2</sup> The same admissibility/feasibility definitions will be used later in Section 4. However, the algorithm in Section 4 maintains a 0-optimal  $f$  and therefore admissible residual arcs always have  $c_\pi(v \rightarrow w) = 0$ .

**Scale initialization.** In the first scale, we compute a  $\bar{\theta}$ -optimal pseudoflow by simply setting  $f(v \rightarrow w) \leftarrow 0$  for all arcs in  $\vec{E}$ . For subsequent scales, we initialize  $f$  to the  $2\theta$ -optimal circulation of the previous scale. First, we raise the potential of all nodes in  $A$  by  $\theta$ , all nodes in  $B$  by  $2\theta$ , and  $t$  by  $3\theta$ . The potential of  $s$  is unchanged. Such potential change increases the reduced cost of all forward arcs to at least  $-\theta$ .

Next, for each backward arc  $w \rightarrow v$  in  $N_f$  with  $c_\pi(w \rightarrow v) < -\theta$ , we set  $f(v \rightarrow w) \leftarrow 0$  (that is, make arc  $v \rightarrow w$  idle), which replaces the backward arc  $w \rightarrow v$  in  $N_f$  with forward arc  $v \rightarrow w$  of positive reduced cost. After this step, the resulting pseudoflow must be  $\theta$ -optimal as all arcs of  $N_f$  have reduced cost at least  $-\theta$ .

The desaturation of each backward arc creates one unit of excess. Since there are at most  $3k$  backward arcs, the pseudoflow has at most  $3k$  excess after SCALE-INIT. There are  $O(n)$  potential updates and  $O(k)$  arcs to desaturate, so the time required for SCALE-INIT is  $O(n)$ .

**Refinement.** The procedure REFINE converts a  $\theta$ -optimal pseudoflow with  $O(k)$  excess into a  $\theta$ -optimal circulation, using a primal-dual augmentation algorithm. A path in  $N_f$  is an *augmenting path* if it begins at an excess node and ends at a deficit node. We call an admissible pseudoflow  $g$  in  $N_f$  an *admissible blocking flow* if  $g$  saturates at least one arc in every admissible augmenting path in  $N_g$ . In other words, there is no admissible excess-deficit path in the residual network after augmentation by  $g$ . Each iteration of REFINE finds an admissible blocking flow to be added to the current pseudoflow in two steps:

1. *Hungarian search*: a Dijkstra-like search that begins at the set of excess nodes and raises potential until there is an excess-deficit path of admissible arcs in  $N_f$ .
2. *Augmentation*: construct an admissible blocking flow by performing depth-first search on the set of admissible arcs of  $N_f$ . It suffices to repeatedly extract admissible augmenting paths until no more admissible excess-deficit paths remain.

The algorithm repeats these steps until the total excess becomes zero. The following lemma bounds the number of iterations in the REFINE procedure at each scale.

► **Lemma 9.** *Let  $\theta$  be the scaling parameter and  $\pi_0$  the potential function at the beginning of a scale, such that there exists an integral  $2\theta$ -optimal circulation with respect to  $\pi_0$ . Let  $f$  be a  $\theta$ -optimal pseudoflow with excess  $O(k)$ . Then REFINE terminates within  $O(\sqrt{k})$  iterations.*

**Proof.** We sketch the proof, which is adapted from Goldberg *et al.* [9]. Let  $f_0$  be the assumed  $2\theta$ -optimal integral circulation with respect to  $\pi_0$ , and let  $\pi$  be the potential maintained during REFINE. Let  $d(v) := (\pi(v) - \pi_0(v))/\theta$ , that is, the increase in potential at  $v$  in units of  $\theta$ . We divide the iterations of REFINE into two phases: before and after every (remaining) excess node has  $d(v) \geq \sqrt{k}$ . Each Hungarian search raises excess potential by at least  $\theta$ , since we use blocking flows. Thus, the first phase lasts at most  $\sqrt{k}$  iterations.

At the start of the second phase, consider the set of arcs  $E^+ := \{v \rightarrow w \in \vec{E} \mid f(v \rightarrow w) < f_0(v \rightarrow w)\}$ . One can argue that the remaining excess with respect to  $f$  is bounded above by the size of any cut separating the excess and deficit nodes [9, Lemma 4]. The proof examines cuts  $Y_i := \{v \mid d(v) > i\}$  for  $0 \leq i \leq \sqrt{k}$ . By  $\theta$ -optimality of  $f$  and  $2\theta$ -optimality of  $f_0$ , one can show that each arc in  $E^+$  crosses at most 3 cuts. Furthermore, the size of  $E^+$  is  $O(k)$ , bounded by the support size of  $f$  and  $f_0$ . Averaging, there is a cut among  $Y_i$ s of size  $O(k/\sqrt{k})$ , so the total excess remaining is  $O(\sqrt{k})$ . Each iteration of REFINE eliminates at least one unit of excess, so the number of second phase iterations is also at most  $O(\sqrt{k})$ . ◀

In the next subsection we show that after  $O(n \text{ polylog } n)$  time preprocessing, an iteration of REFINE can be performed in  $O(k \text{ polylog } n)$  time (Lemma 11). By Lemma 9 and the fact the algorithm terminates in  $O(\log(n/\varepsilon))$  scales, the overall running time of the algorithm is  $O((n + k^{3/2}) \text{ polylog } n \log(1/\varepsilon))$ , as claimed in Theorem 2.

### 3.3 Fast implementation of refinement stage

We now describe a fast implementation of REFINE. The Hungarian search and augmentation steps are similar: each traversing through the residual network using admissible arcs starting from the excess nodes. Due to lack of space, we only describe the former.

At a high level, let  $X$  be the subset of nodes visited by the Hungarian search so far. Initially  $X$  is the set of excess nodes. At each step, the algorithm finds a minimum-reduced-cost arc  $v \rightarrow w$  in  $N_f$  from  $X$  to  $V \setminus X$ . If  $v \rightarrow w$  is not admissible, the potential of all nodes in  $X$  is increased by  $\lceil c_\pi(v \rightarrow w)/\theta \rceil$  to make  $v \rightarrow w$  admissible. If  $w$  is a deficit node, the search terminates. Otherwise,  $w$  is added to  $X$  and the search continues.

Implementing the Hungarian search efficiently is more difficult than in Section 2 because (a) excess nodes may show up in  $A$  as well as in  $B$ , (b) a balanced node may become imbalanced later in the scales, and (c) the potential of excess nodes may be non-uniform. We therefore need a more complex data structure.

We call a node  $v$  of  $N$  *dead* if  $\phi_f(v) = 0$  and no arc of  $\text{supp}(f)$  is incident to  $v$ ; otherwise  $v$  is *alive*. Note that  $s$  and  $t$  are always alive. Let  $A^*$  denote the set of alive nodes in  $A$ ; define  $B^*$  similarly. There are only  $O(k)$  alive nodes, as each can be charged to its adjoining  $\text{supp}(f)$  arcs or its imbalance. We treat alive and dead nodes separately to implement the Hungarian search efficiently. By definition, dead nodes only adjoin forward arcs in  $N_f$ . Thus, the in-degree (resp. out-degree) of a node in  $A \setminus A^*$  (resp.  $B \setminus B^*$ ) is 1, and any path passing through a dead node has a subpath of the form  $s \rightarrow v \rightarrow b$  for some  $b \in B$  or  $a \rightarrow v \rightarrow t$  for some  $a \in A$ . Consequently, a path in  $N_f$  may have at most two consecutive dead nodes, and in the case of two consecutive dead nodes there is a subpath of the form  $s \rightarrow v \rightarrow w \rightarrow t$  where  $v \in A \setminus A^*$  and  $w \in B \setminus B^*$ . We call such paths, from an alive node to an alive node with only dead interior nodes, *alive paths*. Let the reduced cost  $c_\pi(\Pi)$  of an alive path  $\Pi$  be the sum of  $c_\pi$  over its arcs. We say  $\Pi$  is *weakly admissible* if  $c_\pi(\Pi) \leq 0$ .

We find the min-reduced-cost alive path of lengths 1, 2, and 3 leaving  $X$ , then relax the cheapest among them (raise potential of  $X$  by  $\lceil c_\pi(\Pi)/\theta \rceil$  and add every node of  $\Pi$  into  $X$ ). Essentially, relaxing alive paths “skips over” dead nodes. Since reduced costs telescope on paths, weak admissibility of an alive path depends only on the potential of its alive endpoints. Thus, we can query the minimum alive path using a partial assignment of  $\pi$  on only the alive nodes, leaving  $\pi$  over the dead nodes untracked. We now describe a data structure for each path length. Note that our “time budget” per Hungarian search is  $O(k \text{ polylog } n)$ .

**Finding length-1 paths.** This data structure finds a min-reduced-cost arc from an alive node of  $X$  to an alive node of  $V \setminus X$ . There are  $O(k)$  backward arcs, so the minimum among backward arcs can be maintained explicitly in a priority queue and retrieved in  $O(1)$  time.

There are three types of forward arcs:  $s \rightarrow a$  for some  $a \in A^*$ ,  $b \rightarrow t$  for some  $b \in B^*$ , and bipartite arc  $a \rightarrow b$  with two alive endpoints. Arcs of the first (resp. second) type can be found by maintaining  $A^* \setminus X$  (resp.  $B^* \cap X$ ) in a priority queue, but should only be queried if  $s \in X$  (resp.  $t \notin X$ ). The cheapest arc of the third type can be maintained using a dynamic BCP data structure between  $A^* \cap X$  and  $B^* \setminus X$ , with reduced cost as the weighted pair distance. Such a data structure can be implemented so that insertions/deletions can be performed in  $O(\text{polylog } k)$  time [11].

**Finding length-2 paths.** We describe how to find a cheapest path of the form  $s \rightarrow v \rightarrow b$  where  $v$  is dead and  $b \in B^*$ . A cheapest path  $a \rightarrow v \rightarrow t$  can be found similarly. Similar to length-1 paths, we only query paths starting at  $s$  if  $s \in X$  and paths ending at  $t$  if  $t \notin X$ .

## 6:10 Efficient Algorithms for Geometric Partial Matching

Note that  $c_\pi(s \rightarrow v \rightarrow b) = c(v, b) + \pi(b) - \pi(s)$ . Since  $\pi(s)$  is common in all such paths, it suffices to find a pair  $(v, w)$  between  $A \setminus A^*$  and  $B^* \setminus X$  minimizing  $c(v, w) + \pi(w)$ . This is done by maintaining a dynamic BCP data structure between  $A \setminus A^*$  and  $B^* \setminus X$  with the cost of a pair  $(v, w)$  being  $c(v, w) + \pi(w)$ . We may require an update operation for each alive node added to  $X$  during the Hungarian search, of which there are  $O(k)$ , so the time spent during a search is  $O(k \text{ polylog } n)$ .

Since the size of  $A \setminus A^*$  is at least  $r - k$ , we cannot construct this BCP from scratch at the beginning of each iteration. To resolve this, we use the idea of rewinding from Section 2, with a slight twist. There are now *two* ways that the initial BCP may change across consecutive Hungarian searches: (1) the initial set  $X$  may change as nodes lose excess through augmentation, and (2) the set of alive/dead nodes in  $A$  may change. The first is identical to the situation in Section 2; the number of excess depletions is  $O(k)$  over the course of REFINE. For the second, the alive/dead status of a node can change only if the blocking flow found passes through the node. By Lemma 10 below, there are  $O(k)$  such changes per Hungarian search, which can be done in  $O(k \text{ polylog } n)$  time.

**Finding length-3 paths.** We now describe how to find the cheapest path of the form  $s \rightarrow v \rightarrow w \rightarrow t$  where  $v \in A \setminus A^*$  and  $w \in B \setminus B^*$ . Note that  $c_\pi(s \rightarrow v \rightarrow w \rightarrow t) = c(v \rightarrow w) - \pi(s) + \pi(t)$ . A pair  $(v, w)$  between  $A \setminus A^*$  and  $B \setminus B^*$  minimizing  $c(v, w)$  can be found by maintaining a dynamic BCP data structure similar to the case of length-2 paths.

This BCP data structure has no dependency on  $X$  – the only update required comes from membership changes to  $A^*$  or  $B^*$  after an augmentation. Applying Lemma 10 again, there are  $O(k)$  alive/dead updates caused by an augmentation, so the time for these updates per Hungarian search is  $O(k \text{ polylog } n)$ .

**Updating potential.** Potential updates for alive nodes can be handled in a batched fashion as in Section 2. The three data structures above have no dependency on the dead node potential; we leave them untracked as described before. The Hungarian search remains intact since alive nodes are visited in the same order as when using arc-by-arc relaxations. However, we need values of  $\pi$  on all nodes at the end of a scale (for the next SCALE-INIT) and for individual dead nodes whenever they become alive (after augmentation).

We can reconstruct a “valid” potential in these situations. To recover potential for  $v \in A \setminus A^*$  we set  $\pi(v) \leftarrow \pi(s)$ , and for  $v \in B \setminus B^*$  we set  $\pi(v) \leftarrow \pi(t)$ . Straightforward calculation shows that such potential (1) preserves  $\theta$ -optimality, and (2) makes  $\Pi$  (arc-wise) admissible for any weakly admissible alive path  $\Pi$ . Hence, a blocking flow composed of weakly admissible alive paths is admissible under the recovered potential.

The following lemma is crucial to the analysis of running time for the Hungarian search, bounding both the number of relaxations and potential update/recovery operations.

► **Lemma 10.** *Both Hungarian search and augmentation stages explore  $O(k)$  nodes, and the blocking flow found in augmentation stage is incident to  $O(k)$  nodes.*

Augmentation can also be implemented in  $O(k \text{ polylog } n)$  time, after  $O(n \text{ polylog } n)$  time preprocessing, using similar data structures. We thus obtain the following:

► **Lemma 11.** *After  $O(n \text{ polylog } n)$  time preprocessing, each iteration of REFINE can be implemented in  $O(k \text{ polylog } n)$  time.*

## 4 Transportation algorithm

Given two point sets  $A$  and  $B$  in  $\mathbb{R}^2$  of sizes  $r$  and  $n$  respectively and a supply-demand function  $\phi : A \cup B \rightarrow \mathbb{Z}$  as defined in the introduction, we present an  $O(rn^{3/2} \text{polylog } n)$  time algorithm for computing an optimal transport map between  $A$  and  $B$ . By applying this algorithm in the case of  $r \leq \sqrt{n}$  and the one by Agarwal *et al.* [3] when  $r > \sqrt{n}$ , we prove Theorem 3. We use a standard reduction to the uncapacitated min-cost flow problem and use Orlin's algorithm [16] as well as some of the ideas from Agarwal *et al.* [3] for efficient implementation under the geometric settings. We first present an overview of the algorithm and then describe its fast implementation that achieves the desired running time.

### 4.1 Overview of the algorithm

Orlin's algorithm follows an excess-scaling paradigm and the primal-dual framework. It maintains a *scale parameter*  $\Delta$ , a flow function  $f$ , and potential  $\pi$  on the nodes. Initially  $\Delta$  is equal to the total supply,  $f = 0$ , and  $\pi = 0$ . We fix a constant parameter  $\alpha \in (0.5, 1)$ . A node  $v$  is called *active* if the magnitude of imbalance of  $v$  is at least  $\alpha\Delta$ . At each step, using the Hungarian search, the algorithm finds an admissible excess-to-deficit path between active nodes in the residual network and pushes a flow of amount  $\Delta$  along this path.<sup>3</sup> Repeat the process until either active excess or deficit nodes are gone; when this happens,  $\Delta$  is halved. The sequence of augmentations with a fixed value of  $\Delta$  is called an *excess scale*.

The algorithm also performs two preprocessing steps at the beginning of each excess scale. First, if  $f(v \rightarrow w) \geq 3n\Delta$ ,  $v \rightarrow w$  is contracted to a single node  $z$  with  $\phi(z) = \phi(v) + \phi(w)$ .<sup>4</sup> Second, if there are no active excess nodes and  $f(v \rightarrow w) = 0$  for every arc  $v \rightarrow w$ , then  $\Delta$  is aggressively lowered to  $\max_v \phi(v)$ .

When the algorithm terminates, an optimal circulation in the contracted network is found. We use the algorithm described in Agarwal *et al.* [3] to recover an optimal circulation for the original network in  $O(n \text{polylog } n)$  time. Orlin showed that the algorithm terminates within  $O(n \log n)$  scales and performs a total of  $O(n \log n)$  augmentations. In the next subsection, we describe an algorithm that, after  $O(n \text{polylog } n)$  time preprocessing, finds an admissible excess-to-deficit path in  $O(r\sqrt{n} \text{polylog } n)$  amortized time. Summing this cost over all augmentations, we obtain the desired running time.

### 4.2 An efficient implementation

Recall in the previous sections that we could bound the running time of the Hungarian search by the size of  $\text{supp}(f)$ . Here, the number of active imbalanced nodes at any scale is  $O(r)$ , and the length of an augmenting path is also  $O(r)$ . Therefore one might hope to find an augmenting path in  $O(r \text{polylog } n)$  time, by adapting the algorithms described in Sections 2 and 3. The challenge is that  $\text{supp}(f)$  may have  $\Omega(n)$  size, therefore an algorithm which runs in time proportional to the support size is no longer sufficient. Still, we manage to implement Hungarian search in time  $O(r\sqrt{n} \text{polylog } n)$ , by exploiting a few properties of  $\text{supp}(f)$  as described below.

We note that each arc of  $\text{supp}(f)$  is admissible with reduced cost 0, so we prioritize the relaxation of support arcs as soon as they arrive in  $X \times (V \setminus X)$ , over the non-support arcs. This strategy ensures the following crucial property.

<sup>3</sup> Note that this augmentation may convert an excess node into a deficit node.

<sup>4</sup> Intuitively,  $f(v \rightarrow w)$  is so high that future scales cannot deplete the flow on  $v \rightarrow w$ .

► **Lemma 12.** *If the support arcs  $\text{supp}(f)$  are relaxed as soon as possible,  $\text{supp}(f)$  is acyclic.*

Next, similar to Section 3, we call node  $u$  *alive* if (a)  $u$  is an active imbalanced node or (b) if  $u$  is incident to an arc of  $\text{supp}(f)$ ;  $u$  is *dead* otherwise. Unlike in Section 3, once a node becomes alive it cannot be dead again. Furthermore, a dead node may become alive only at the beginning of a scale (after the value of  $\Delta$  is reduced). Also, an augmenting path cannot pass through a dead node. Therefore, we can ignore all dead nodes during Hungarian search, and update the set of alive/dead nodes at the beginning of a scale.

Let  $B^* \subseteq B^\circledast$  be the set of nodes that are either (a) active imbalanced nodes or (b) incident to *exactly one* arc of  $\text{supp}(f)$ . Lemma 12 implies that  $B^\circledast \setminus B^*$  has size  $O(r)$ . We can therefore find the min-reduced-cost arc between  $X \cap A^\circledast$  and  $B^\circledast \setminus (B^* \cup X)$  using a BCP data structure as in Section 2, along with lazy potential updates and the rewinding mechanism. The total time spent by Hungarian search on the nodes of  $B^\circledast \setminus B^*$  will be  $O(r \text{ polylog } n)$ . We subsequently focus on handling  $B^*$ .

**Handling  $B^*$ .** We now describe how we query a min-reduced-cost arc between  $X \cap A^\circledast$  and  $B^* \setminus X$ . Each node  $b \in B^*$  is incident to exactly one arc in  $\text{supp}(f)$ . We partition these nodes into clusters depending on their unique neighbor in  $N_f$ . That is, for a node  $a \in A^\circledast$ , let  $B_a^* := \{b \in B^* \mid a \rightarrow b \in \text{supp}(f)\}$ . We refer to  $B_a^*$  as the *star* of  $a$ .

The crucial observation is that  $a$  is the only node in  $N_f$  reachable from each  $b \in B_a^*$ , so once the Hungarian search reaches a node of  $B_a^*$  and thus  $a$  (recall we prioritize relaxing support arcs), the Hungarian search need not visit any other nodes of  $B_a^*$ , as they will only lead to  $a$ . Hence, as soon as one node of  $B_a^*$  is reached, all other nodes of  $B_a^*$  can be discarded from further consideration. Using this observation, we handle  $B^*$  as follows.

We classify each  $a \in A^\circledast$  as *light* or *heavy*: heavy if  $|B_a^*| \geq \sqrt{n}$ , and light if  $|B_a^*| \leq 2\sqrt{n}$ . Note that if  $\sqrt{n} \leq |B_a^*| \leq 2\sqrt{n}$  then  $a$  may be classified as light or heavy. We allow this flexibility to implement reclassification in a lazy manner. Namely, a light node is reclassified as heavy once  $|B_a^*| > 2\sqrt{n}$ , and a heavy node is reclassified as light once  $|B_a^*| < \sqrt{n}$ . This scheme ensures that the star of  $a$  has gone through at least  $\sqrt{n}$  updates between two successive reclassifications, and these updates will pay for the time spent in updating the data structure when  $a$  is re-classified.

For each heavy node  $a \in A^\circledast \setminus X$ , we maintain a BCP data structure between  $B_a^*$  and  $X \cap A^\circledast$ . Next, for all light nodes in  $A^\circledast \setminus X$ , we collect their stars into a single set  $B_{<}^* := \bigcup_{a \text{ light}} B_a^*$ . We maintain one single BCP data structure between  $B_{<}^*$  and  $A^\circledast \cap X$ . Thus, at most  $r$  different BCP data structures are maintained for stars.

Using these data structures, we can compute and relax a min-reduced-cost arc  $v \rightarrow w$  between  $A^\circledast \cap X$  and  $B^* \setminus X$ . If  $w$  lies in some star  $B_a^*$ , then we also add  $a$  into  $X$ . If  $a$  is light, then we delete  $B_a^*$  from  $B_{<}^*$  and update the BCP data structure of  $B_{<}^*$ . If  $a$  is heavy, then we stop querying the BCP data structure of  $B_a^*$  for the remainder of the search. Finally, since  $a$  becomes part of  $X$ ,  $a$  is added to all  $O(r)$  BCP data structures. Recall that  $r \leq \sqrt{n}$  by assumption. Adding arc  $v \rightarrow w$  thus involves performing  $O(\sqrt{n})$  insertion/deletion operations in various BCP data structures, thereby taking  $O(\sqrt{n} \text{ polylog } n)$  time.

**Putting it together.** While proof is omitted, the following lemma bounds the running time of the Hungarian search.

► **Lemma 13.** *Assuming all BCP data structures are initialized correctly, the Hungarian search terminates within  $O(r)$  steps, and takes  $O(r\sqrt{n} \text{ polylog } n)$  time.*



Once an augmenting path is found and the augmentation is performed, the set of imbalanced nodes and the support arcs change. We thus need to update the sets  $B^*$ ,  $B_a^*$ s, and  $B_{<}^*$ . This can be accomplished in  $O(r \text{ polylog } n)$  amortized time. When we begin a new Hungarian search, we use the rewinding mechanism to set various BCP data structures in the right initial state. Finally, when we move from one scale to another, we also update the sets  $A^*$  and  $B^*$ . Omitting all the details, we conclude the following.

► **Lemma 14.** *Each Hungarian search can be performed in  $O(r\sqrt{n} \text{ polylog } n)$  time.*

Since there are  $O(n \log n)$  augmentations and the flow in the original network can be recovered from that in the contracted network in  $O(n \text{ polylog } n)$  time [3], the total running time of the algorithm is  $O(rn^{3/2} \text{ polylog } n)$ , as claimed in Theorem 3.

---

## References

- 1 Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical Decomposition of Shallow Levels in 3-Dimensional Arrangements and Its Applications. *SIAM J. Comput.*, 29(3):912–953, 1999. doi:10.1137/S0097539795295936.
- 2 Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster Algorithms for the Geometric Transportation Problem. In *Proc. 33rd Int. Sympos. Comput. Geom. (SoCG)*, pages 7:1–7:16, 2017. doi:10.4230/LIPIcs.SoCG.2017.7.
- 3 Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster Algorithms for the Geometric Transportation Problem. Preprint, 2019. arXiv:1903.08263.
- 4 Pankaj K. Agarwal and Kasturi R. Varadarajan. A near-linear constant-factor approximation for Euclidean bipartite matching? In *Proc. 20th Annu. Sympos. Comput. Geom. (SoCG)*, pages 247–252, 2004. doi:10.1145/997817.997856.
- 5 D. Bertsekas and D. El Baz. Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems. *SIAM J. Control and Opt.*, 25(1):74–85, 1987. doi:10.1137/0325006.
- 6 Ran Duan, Seth Pettie, and Hsin-Hao Su. Scaling Algorithms for Weighted Matching in General Graphs. *ACM Trans. Algorithms*, 14(1):8:1–8:35, 2018. doi:10.1145/3155301.
- 7 Shimon Even and Robert E. Tarjan. Network Flow and Testing Graph Connectivity. *SIAM J. Comput.*, 4(4):507–518, 1975. doi:10.1137/0204043.
- 8 Harold N. Gabow and Robert E. Tarjan. Faster Scaling Algorithms for Network Problems. *SIAM J. Comput.*, 18(5):1013–1036, 1989. doi:10.1137/0218069.
- 9 Andrew V. Goldberg, Sagi Hed, Haim Kaplan, and Robert E. Tarjan. Minimum-Cost Flows in Unit-Capacity Networks. *Theoret. Comput. Sci.*, 61(4):987–1010, 2017. doi:10.1007/s00224-017-9776-7.
- 10 John E. Hopcroft and Richard M. Karp. An  $n^{5/2}$  Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. doi:10.1137/0202019.
- 11 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic Planar Voronoi Diagrams for General Distance Functions and their Algorithmic Applications. In *Proc. 28th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 2495–2504, 2017. doi:10.1137/1.9781611974782.165.
- 12 Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov. Preconditioning for the Geometric Transportation Problem. Preprint, 2019. arXiv:1902.08384.
- 13 Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.
- 14 Yin Tat Lee and Aaron Sidford. Path Finding Methods for Linear Programming: Solving Linear Programs in  $\tilde{O}(\sqrt{\text{rank}})$  Iterations and Faster Algorithms for Maximum Flow. In *55th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 424–433, 2014. doi:10.1109/FOCS.2014.52.

- 15 Aleksander Madry. Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back. In *54th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 253–262, 2013. doi:10.1109/FOCS.2013.35.
- 16 James B. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. *Operations Research*, 41(2):338–350, 1993. doi:10.1287/opre.41.2.338.
- 17 Lyle Ramshaw and Robert E. Tarjan. A Weight-Scaling Algorithm for Min-Cost Imperfect Matchings in Bipartite Graphs. In *Proc. 53rd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 581–590, 2012. doi:10.1109/FOCS.2012.9.
- 18 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time  $\epsilon$ -approximation algorithm for geometric bipartite matching. In *Proc. 44th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 385–394, 2012. doi:10.1145/2213977.2214014.
- 19 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proc. 23rd Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 306–317, 2012. URL: <https://dl.acm.org/citation.cfm?id=2095116.2095145>.
- 20 Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–256, 1985. doi:10.1007/BF02579369.
- 21 Pravin M. Vaidya. Geometry Helps in Matching. *SIAM J. Comput.*, 18(6):1201–1225, 1989. doi:10.1137/0218080.
- 22 Kasturi R. Varadarajan. A Divide-and-Conquer Algorithm for Min-Cost Perfect Matching in the Plane. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 320–331, 1998. doi:10.1109/SFCS.1998.743466.

# Connecting the Dots (with Minimum Crossings)

**Akanksha Agrawal**

Ben-Gurion University, Beer-Sheva, Israel  
agrawal@post.bgu.ac.il

**Grzegorz Guśpiel**

Theoretical Computer Science Department, Faculty of Mathematics and Computer Science,  
Jagiellonian University, Kraków, Poland  
guspiel@tcs.uj.edu.pl

**Jayakrishnan Madathil**

The Institute of Mathematical Sciences, HBNI, Chennai, India  
jayakrishnanm@imsc.res.in

**Saket Saurabh**

The Institute of Mathematical Sciences, HBNI, Chennai, India  
University of Bergen, Bergen, Norway  
saket@imsc.res.in

**Meirav Zehavi**

Ben-Gurion University, Beer-Sheva, Israel  
meiravze@bgu.ac.il

---

## Abstract

We study a prototype CROSSING MINIMIZATION problem, defined as follows. Let  $\mathcal{F}$  be an infinite family of (possibly vertex-labeled) graphs. Then, given a set  $P$  of (possibly labeled)  $n$  points in the Euclidean plane, a collection  $L \subseteq \text{Lines}(P) = \{\ell : \ell \text{ is a line segment with both endpoints in } P\}$ , and a non-negative integer  $k$ , decide if there is a subcollection  $L' \subseteq L$  such that the graph  $G = (P, L')$  is isomorphic to a graph in  $\mathcal{F}$  and  $L'$  has at most  $k$  crossings. By  $G = (P, L')$ , we refer to the graph on vertex set  $P$ , where two vertices are adjacent if and only if there is a line segment that connects them in  $L'$ . Intuitively, in CROSSING MINIMIZATION, we have a set of locations of interest, and we want to build/draw/exhibit connections between them (where  $L$  indicates where it is feasible to have these connections) so that we obtain a structure in  $\mathcal{F}$ . Natural choices for  $\mathcal{F}$  are the collections of perfect matchings, Hamiltonian paths, and graphs that contain an  $(s, t)$ -path (a path whose endpoints are labeled). While the objective of seeking a solution with few crossings is of interest from a theoretical point of view, it is also well motivated by a wide range of practical considerations. For example, links/roads (such as highways) may be cheaper to build and faster to traverse, and signals/moving objects would collide/interrupt each other less often. Further, graphs with fewer crossings are preferred for graphic user interfaces.

As a starting point for a systematic study, we consider a special case of CROSSING MINIMIZATION. Already for this case, we obtain NP-hardness and W[1]-hardness results, and ETH-based lower bounds. Specifically, suppose that the input also contains a collection  $D$  of  $d$  non-crossing line segments such that each point in  $P$  belongs to exactly one line in  $D$ , and  $L$  does not contain line segments between points on the same line in  $D$ . Clearly, CROSSING MINIMIZATION is the case where  $d = n - 1$  – then,  $P$  is in general position. The case of  $d = 2$  is of interest not only because it is the most restricted non-trivial case, but also since it corresponds to a class of graphs that has been well studied – specifically, it is CROSSING MINIMIZATION where  $G = (P, L)$  is a (bipartite) graph with a so called *two-layer drawing*. For  $d = 2$ , we consider three basic choices of  $\mathcal{F}$ . For perfect matchings, we show (i) NP-hardness with an ETH-based lower bound, (ii) solvability in subexponential parameterized time, and (iii) existence of an  $\mathcal{O}(k^2)$ -vertex kernel. Second, for Hamiltonian paths, we show (i) solvability in subexponential parameterized time, and (ii) existence of an  $\mathcal{O}(k^2)$ -vertex kernel. Lastly, for graphs that contain an  $(s, t)$ -path, we show (i) NP-hardness and W[1]-hardness, and (ii) membership in XP.



© Akanksha Agrawal, Grzegorz Guśpiel, Jayakrishnan Madathil, Saket Saurabh, and Meirav Zehavi;

licensed under Creative Commons License CC-BY

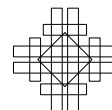
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 7; pp. 7:1–7:17

Leibniz International Proceedings in Informatics



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



**2012 ACM Subject Classification** Theory of computation → Fixed parameter tractability

**Keywords and phrases** crossing minimization, parameterized complexity, FPT algorithm, polynomial kernel,  $W[1]$ -hardness

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.7

**Related Version** A full version of the paper is available at <https://akanksha-agrawal.weebly.com/uploads/1/2/2/2/122276497/crossings.pdf>.

**Funding** *Akanksha Agrawal*: The work was carried out when the author was employed at Hungarian Academy of Sciences, and was supported by ERC Consolidator Grant SYSTEMATICGRAPH (No. 46 725978).

*Grzegorz Guśpiel*: Partially supported by the MNiSW grant DI2013 000443.

*Saket Saurabh*: Supported by ERC Consolidator Grant LOPPRE (No. 819416).

*Meirav Zehavi*: Supported by ISF grant no. 1176/18.

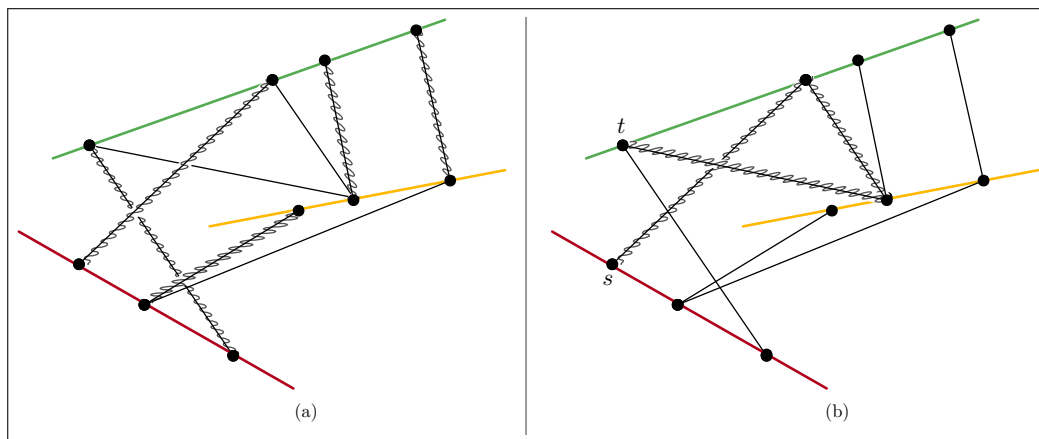
**Acknowledgements** We thank Grzegorz Gutowski and Paweł Rzażewski for many valuable comments regarding the NP-hardness proof for CM-PM.

## 1 Introduction

Let  $\mathcal{F}$  be an infinite family of (possibly vertex-labeled) graphs. Suppose that given a graph  $F$ , the membership of  $F$  in  $\mathcal{F}$  is testable in time polynomial in the size of  $F$ . For the family  $\mathcal{F}$ , we define a prototype CROSSING MINIMIZATION problem as follows (see Fig. 1). Given a set  $P$  of (possibly labeled)  $n$  points in the two-dimensional Euclidean plane, a collection  $L \subseteq \text{Lines}(P) = \{\ell : \ell \text{ is a line segment with both endpoints in } P\}$ , and a non-negative integer  $k$ , decide if there exists a subcollection  $L' \subseteq L$  such that the graph  $G = (P, L')$  is isomorphic<sup>1</sup> to a graph in  $\mathcal{F}$  and  $L'$  has at most  $k$  crossings. The notation  $G = (P, L')$  refers to the graph on vertex set  $P$ , where two vertices are adjacent if and only if there is a line segment that connects them in  $L'$ . Moreover, the number of crossings of  $L'$  is the number of pairs of line segments in  $L'$  that intersect each other at a point other than their possible common endpoint. The CROSSING MINIMIZATION problem is a general model for a wide range of scenarios where we have a set of points of interest that correspond to geographical areas or fixed objects such as cities, manufacturing machinery or immobile equipment, attractions and mailboxes, and we want to build, draw or exhibit connections between them (where  $L$  indicates where it is feasible to have these connections) in order to obtain a structure in  $\mathcal{F}$ .

While the objective of seeking a solution with few crossings is of interest from a theoretical viewpoint, it is also well motivated by practical considerations. For example, public tracks (such as roads, highways or even paths in amusement parks) with fewer crossings require the construction of less bridges, elevated tracks, traffic lights and roundabouts, and therefore they are likely to be cheaper to build [42], easier and faster to traverse [10], and cause less accidents [21]. Moreover, signals and moving objects would interrupt each other less often. This property may be crucial as frequent collision between signals can distort or weaken them [3]. Furthermore, for moving objects such as robots (cleaning robots, autonomous agents and self-driving cars) that cannot physically be present in an intersection point simultaneously, encountering a large number of crossings may require the development of more complex navigation and sensory systems [37]. Lastly, graphs with fewer crossings are easier to view and analyze – in graphic user interfaces, for example, visual clarity is a major issue [13].

<sup>1</sup> With respect to vertex-labeled graphs, isomorphism also preserves the labeling of vertices rather than only their adjacency relationships – that is, a vertex labeled  $i$  can only be mapped to a vertex labeled  $i$ .



■ **Figure 1** An instance of CROSSING MINIMIZATION (in black) where  $\mathcal{F}$  is the family of (a) perfect matchings, and (b) graphs that have an  $(s, t)$ -path. Solution edges are marked by squiggly lines – the number of crossings is 2 in (a) and 1 in (b). The  $d = 3$  colorful line segments display  $D$ .

Keeping the above applications in mind, three natural choices for the family  $\mathcal{F}$  are the family of (Hamiltonian) paths, the family of graphs that contain an  $(s, t)$ -path (identification of  $s$  and  $t$  is modeled by vertex labels), and the family of (possibly vertex-labeled) perfect matchings. Indeed, these families model the most basic scenarios where all points must be connected by a path (e.g., to plan tracks for sightseeing trains or maintenance equipment such as cleaning robots or lawn mowers), only a specific pair of points must be connected by a path (e.g., to transport goods between two destinations), or the points are to be matched with one another (e.g., to pair up robots and charging ports). Furthermore, the computational problems that correspond to these families – HAMILTONIAN PATH,  $(s, t)$ -PATH and PERFECT MATCHING, respectively – are among the most classical problems in computer science [22, 29, 18, 11].

As a starting point for a systematic study, we consider a special case of CROSSING MINIMIZATION. Already for this case, we obtain NP-hardness and W[1]-hardness results, and ETH-based lower bounds, alongside positive results. Specifically, suppose that the input also contains a collection  $D$  of  $d$  non-crossing line segments such that each point in  $P$  belongs to exactly one line in  $D$ , and  $L$  does not contain line segments between points on the same line in  $D$  (see Fig. 1).<sup>2</sup> Clearly, CROSSING MINIMIZATION is the case where  $d = n$  – then, the set  $P$  can be in general position. The case of  $d = 2$  is of interest not only because it is the most restricted non-trivial case, but also since it corresponds to a class of graphs that has been well studied in the literature – specifically, this case is precisely CROSSING MINIMIZATION where  $G = (P, L)$  is a (bipartite) graph with a so called *two-layer drawing*. Clearly, our hardness results carry over to any generalization of the case where  $d = 2$ . For this case, we consider the aforementioned three basic choices of  $\mathcal{F}$ , and obtain a comprehensive picture of their complexity. In what follows, we discuss our contribution, and then review related literature.

## 1.1 Our Contribution

Our study focuses on the class of two-layered graphs. Formally, a *two-layered graph* is a bipartite graph  $G$  with vertex bipartition  $V(G) = X \cup Y$  that has a *two-layer drawing* – that is, a placement of the vertices of  $X$  on distinct points on a straight line segment  $L_1$ , and the

<sup>2</sup> Having line segments between points on the same line in  $D$  only makes the problem more general.

## 7:4 Connecting the Dots (with Minimum Crossings)

vertices of  $Y$  on distinct points on a different straight line segment  $L_2$  such that  $L_1$  and  $L_2$  are parallel to each other. (For ease of understanding, we take  $L_1$  to be a segment of the line  $y = 1$  in the plane, and similarly,  $L_2$  to be a segment of the line  $y = 0$ .) The relative positions of the vertices in  $X$  and  $Y$  on  $L_1$  and  $L_2$ , respectively, are given by permutations  $\sigma_X$  and  $\sigma_Y$ . Each edge is drawn using a straight line segment connecting the points of its end-vertices. We refer to  $(\sigma_X, \sigma_Y)$  as the *two-layered embedding/drawing* of  $G$ . Note that  $(\sigma_X, \sigma_Y)$  uniquely determines which edges intersect. The crossing minimization problem that corresponds to PERFECT MATCHING on two-layered graphs is defined as follows.

CROSSING-MINIMIZING PERFECT MATCHING (CM-PM) **Parameter:**  $k$   
**Input:** A two-layered graph  $G$  (i.e., a bipartite graph  $G$  with bipartition  $V(G) = X \cup Y$ , and orderings  $\sigma_X$  and  $\sigma_Y$  of  $X$  and  $Y$ , respectively), and a non-negative integer  $k$ .  
**Question:** Does  $G$  have a perfect matching with at most  $k$  crossings?

Similarly, we define the crossing minimization variants of HAMILTONIAN PATH (the existence of a path that visits all vertices)<sup>3</sup> and  $(s, t)$ -PATH (the existence of a path between two designated vertices). We refer to these problems as CROSSING-MINIMIZING HAMILTONIAN PATH (CM-HP) and CROSSING-MINIMIZING  $(s, t)$ -PATH (CM-PATH).

**Our Results.** In this paper, we present a comprehensive picture of both the classical and parameterized computational complexities of these three problems as follows.<sup>4</sup>

### CM-PM.

- **Negative.** NP-complete *even on graphs of maximum degree 2*. Moreover, unless the ETH fails, it can be solved neither in time  $2^{o(n+m)}$  nor in time  $2^{o(\sqrt{k})} n^{\mathcal{O}(1)}$  on these graphs, (where  $n$  and  $m$  are respectively the number of vertices and edges of the input graph.)
- **Positive.** Admits a kernel with  $\mathcal{O}(k^2)$  vertices. Moreover, it admits a subexponential parameterized algorithm with running time  $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ . In light of the negative result above, the running time of this algorithm is optimal under ETH.

We briefly remark that the proof of NP-completeness of CM-PM resolves an open question related to a problem called TOKEN SWAPPING (see Section 1.2), introduced in 2014 by Yamanaka et al. [48, 49]. Two generalizations of TOKEN SWAPPING were introduced by Yamanaka et al. [48, 49] and Bonnet et al. [7], both known to be NP-complete due to Miltzow et al. [40]. One of the results of Bonnet et al. [7] is the analysis of the complexity of all three token swapping problems on simple graph classes, including trees, cliques, stars and paths. SUBSET TOKEN SWAPPING was shown to be NP-complete on the first three classes, but the status of the problem for paths was unknown. Since SUBSET TOKEN SWAPPING restricted to paths is equivalent to our CM-PM (noted by Miltzow [39]), we derive that SUBSET TOKEN SWAPPING restricted to paths is NP-complete as well.

<sup>3</sup> We remark that our results for HAMILTONIAN PATH can be extended to HAMILTONIAN CYCLE.

<sup>4</sup> Due to lack of space, several proofs have been omitted from the extended abstract.

**CM-HP.**

- **Negative.** NP-complete even on graphs that admit a Hamiltonian path. Unless the ETH fails, it can be solved neither in time  $2^{o(n+m)}$  nor in time  $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$  on these graphs.
- **Positive.** Admits a kernel with  $\mathcal{O}(k^2)$  vertices. Moreover, it admits a subexponential parameterized algorithm with running time  $2^{\mathcal{O}(\sqrt{k} \log k)}n^{\mathcal{O}(1)}$ . In light of the negative result above, the running time of this algorithm is almost optimal under ETH.

While HAMILTONIAN PATH is a classical NP-complete problem [22], we prove that in the case of CM-HP, the hardness holds even if we know of a Hamiltonian path in the input graph (in which case HAMILTONIAN PATH is trivial). We also comment that in the case of CM-HP (and also CM-PATH), unlike the case of CM-PM, the problem becomes trivially solvable in polynomial time on graphs of maximum degree 2. Indeed, graphs of maximum degree 2 are collections of paths and cycles, and hence admit only linearly in  $n$  many Hamiltonian paths that can be easily enumerated in polynomial time. Then, CM-HP is solved by testing whether at least one of these Hamiltonian paths has at most  $k$  crossings. In fact, most natural NP-complete graph problems become solvable in polynomial time on graphs of maximum degree 2, therefore we find the hardness of CM-PM on these graphs quite surprising.

**CM-Path.**

- **Negative.** NP-complete and W[1]-hard. Specifically, unless  $W[1] = FPT$ , it admits neither an algorithm with running time  $f(k)n^{\mathcal{O}(1)}$  nor a kernel of size  $f(k)$ , for any computable function  $f$  of  $k$ .
- **Positive.** Member in XP. Specifically, it is solvable in time  $n^{\mathcal{O}(k)}$ .

In light of our first two sets of results, we find our third set of results quite surprising:  $(s, t)$ -PATH is the easiest to solve among itself, PERFECT MATCHING and HAMILTONIAN PATH,<sup>5</sup> yet when crossing minimization is involved,  $(s, t)$ -PATH is substantially more difficult than the other two problems – indeed, CM-PM is not even FPT (unless  $W[1] = FPT$ ).

**Our Methods.** In what follows, we give a brief overview of our methods.

**CM-PM.** We prove that CM-PM on graphs of maximum degree 2 is NP-hard by a reduction from VERTEX COVER. The same reduction shows that CM-PM does not admit any  $2^{o(n+m)}$ -time (or  $2^{o(\sqrt{k})}n^{\mathcal{O}(1)}$ -time) algorithm unless the ETH fails.

For our algorithm and kernel, consider an instance  $(G, k)$  of CM-PM, where  $V(G) = X \cup Y$  is the vertex bipartition with  $|X| = |Y| = n$ . For  $i \in [n]$ , let  $x_i$  and  $y_i$  denote the  $i^{\text{th}}$  vertices of  $X$  and  $Y$ , respectively, in the given two-layered embedding of  $G$ . It is not difficult to see that the only perfect matching with no crossings, if such a matching exists, is  $\{x_i y_i \mid i \in [n]\}$ . Therefore, if  $M$  is a perfect matching and  $x_i y_j \in M$  with  $i \neq j$ , then the edge  $x_i y_i$  must

<sup>5</sup> In particular,  $(s, t)$ -PATH can be directly solved in linear time via BFS [11], while PERFECT MATCHING is only known to be solvable by more complex (non-linear time) algorithms such as Edmonds algorithm [18], and the status of HAMILTONIAN PATH is even worse given that it is NP-complete [22].

intersect another edge in  $M$ , which yields a crossing. In fact,  $x_i y_j$  must intersect at least  $|j - i|$  edges. Therefore, no feasible solution for CM-PM can contain an edge  $x_i y_j$  with  $|j - i| > k$ . This observation plays a key role in both our algorithm and kernel designs. Our algorithm is based on dynamic programming (DP), and its analysis is based on Hardy-Ramanujan numbers [26]. (By considering these numbers, we are able to derive a running time bound of  $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{k})})$ .) Very briefly, at stage  $i$  we consider the graph  $G_i$ , the subgraph of  $G$  induced by  $X_i \cup Y_i = \{x_j, y_j \mid j \leq i\}$ . Our algorithm “guesses” which subsets of  $V(G_i)$  are going to be matched to “future vertices”, i.e., vertices in  $V(G) \setminus V(G_i)$ , in an optimal solution, and solves the problem optimally on the graph induced by the remaining vertices. For the kernel, we show that either  $(G, k)$  is a no-instance or the number of “bad pairs”, i.e.,  $\{x_i, y_i\}$  where  $x_i y_i \notin E(G)$ , cannot exceed  $2k$ . We then bound the number of pairs  $\{x_i, y_i\}$  between two consecutive bad pairs by  $\mathcal{O}(k)$  again, which gives a kernel with  $\mathcal{O}(k^2)$  vertices.

**CM-HP.** By a reduction from a variant of HAMILTONIAN PATH on bipartite graphs, we show that CM-HP is NP-hard even if the input graph is assumed to have a Hamiltonian path. For our FPT algorithm and kernel, we adopt a strategy similar to the one we employed for CM-PM.

**CM-PATH.** We prove the W[1]-hardness of CM-PATH by giving an appropriate reduction from MULTI-COLORED CLIQUE, which is known to be W[1]-hard [19]. Given an instance  $(G, V_1, V_2, \dots, V_k)$  of MULTI-COLORED CLIQUE ( $G$  is a  $k$ -partite graph, and the problem is to check whether  $G$  contains a clique with exactly one vertex from each  $V_i$ ), we create an equivalent instance  $(G', X, Y, s, t, k')$  of CM-PATH, where  $G'$  is a two-layered graph, as follows. We create an  $s$ - $t$  path in  $G'$  that “selects” a vertex from each  $V_i$  and an edge for each (distinct) pair  $(V_i, V_j)$ . To this end, for each  $V_i$ , we have a vertex selection gadget  $\mathcal{V}_i$ , and for (distinct)  $V_i, V_j$ , we have an edge selection gadget  $\mathcal{E}_{ij}$ . The vertex and edge selection gadgets are arranged in a linear fashion to create an  $s - t$  path in  $G'$ . In the construction, we add a pair of non-adjacent vertices in  $\mathcal{E}_{ij}$  for each edge between  $V_i$  and  $V_j$ . We also add a path between the pair of (non-adjacent) vertices whose edges cross the gadgets  $\mathcal{V}_i$  and  $\mathcal{V}_j$ , which enforces compatibility between vertices and edges that are selected. Finally, by setting  $k'$  appropriately, we get the desired reduction.

As for the XP algorithm for CM-PATH, we guess which edges of  $G$  are going to be involved in crossings in a feasible solution. The problem then reduces to connecting these guessed edges using crossing-free subpaths, which can be done in polynomial time.

## 1.2 Related Works

**The Crossing Number Problem.** The *crossing number* of a graph  $G$  is the minimum number of crossings in a plane drawing of  $G$ . The notion of a crossing number originally arose in 1940 by Turán [46] for bipartite graphs in the context of the minimization of the number of crossings between tracks connecting brick kilns to storage sites. Computationally, the input of the CROSSING NUMBER problem is a graph  $G$  and a non-negative integer  $k$ , and the task is to decide whether the crossing number of  $G$  is at most  $k$ . This problem is among the most classical and fundamental graph layout problems in computer science. It was shown to be NP-complete by Garey and Johnson in 1983 [23]. Not only is the problem NP-complete on graphs of maximum degree 3 [27], but also it is surprisingly NP-complete even on graphs that can be made planar and hence crossing-free by the removal of just a single edge [8]. Nevertheless, CROSSING NUMBER was shown to be FPT by Grohe already in 2001 [24], who



developed an algorithm that runs in time  $f(k)n^2$  where  $f$  is at least double exponential.<sup>6</sup> A further development was achieved by Kawarabayashi and Reed [32], who showed that the problem is solvable in time  $f(k)n$ . On the negative side, Hlinený and Dernár [28] proved that CROSSING NUMBER does not admit a polynomial kernel unless  $\text{NP} \subseteq \text{coNP}/\text{poly}$ .

Variants of CROSSING NUMBER where the vertices can be placed only on prespecified curves are extensively studied. Closely related to our work is the well-known TWO-LAYER CROSSING MINIMIZATION problem: given a bipartite graph  $G$  with vertex bipartition  $V(G) = X \cup Y$ , and a non-negative integer  $k$ , the task is to decide whether  $G$  admits a two-layered drawing where the number of crossings is at most  $k$ . This problem originated in VLSI design [44]. A solution to the TWO-LAYER CROSSING MINIMIZATION problem is also useful in solving the rank aggregation problem, which has applications in meta-search and spam reduction on the Web [6]. We refer the reader to [50] and references therein for other applications. The TWO-LAYER CROSSING MINIMIZATION problem is long known to be NP-complete, even in its one sided version where we are allowed to permute vertices only from one (fixed) side [16, 17]. Further, the membership of TWO-LAYER CROSSING MINIMIZATION in FPT has already been proven close to two decades ago by Dujmovic et al. [15]. Noteworthy is also the well-studied variant of CROSSING NUMBER that restricts the vertices to be placed only on a prespecified circle and edges are drawn as straight line segments. Both of these variants as well as their various versions are subject to an active line of research [33]. Further, aesthetic display of these layouts are of importance in biology [36], and included in standard graph layout software [31] such as yFiles, Graphviz, or OGDF. For more information on CROSSING NUMBER and its variants, we refer to surveys such as [43].

**Problems on Fixed Point Sets.** Settings where we are given a set  $P$  of points in the plane that represent vertices, and edges are to be drawn as straight lines between them, are intensively studied since the early 80s. A large body of work has been devoted to the establishment of combinatorial bounds on the number of *crossing-free* graphs on  $P$ , where particular attention is given to crossing-free triangulations, perfect matchings and Hamiltonian paths and cycles. Originally, the study of these bounds was initiated by Newborn and Moser in 1980 [41] for crossing-free Hamiltonian cycles. For more information, we refer to the excellent Introduction of Sharir and Welzl [45] and the references therein. Computationally, the problem of *counting* the number of such crossing-free graphs (faster than the time required to enumerate them) is of great interest (see, e.g., [47, 4, 38]). Furthermore, the computation of a single crossing-free graph on  $P$  (such as a perfect matching), possibly with a special property of being “short” [2, 1, 9], has already been studied since 1993 [30]. To the best of our knowledge, the minimization of the number of crossings (rather than the detection of a crossing-free graph) has received only little attention, mostly in an ad-hoc fashion. An exception to this is the work of Halldórsson et al. [25] with respect to spanning trees. We remark that they study the problem in its full generality, where the computation of even a crossing-free spanning tree is already NP-complete [34, 30].

Related to our study is also the METRO LINE CROSSING MINIMIZATION problem, introduced by Benkert et al. [5]. Given an embedded graph  $G$  on  $P$ , as well as  $k$  pairs of vertices (called terminals), a solution to this problem is a set of paths that connect their respective pairs of terminals, and which has minimum number of “crossings” under a definition different

---

<sup>6</sup> We find the contrast between this result and our result on CM-PATH somewhat surprising. At first glance, our CM-PATH problem seems computationally simpler than CROSSING NUMBER (where the embedding is computed from scratch), yet our problem is W[1]-hard while CROSSING NUMBER is FPT.

than ours. Specifically, paths are thought of as being drawn in the plane “alongside” the edges of  $G$  rather than on the edges themselves. Such a formulation allows to reuse a single edge a large number of times. Therefore, the avoidance of crossings might come at the cost of congesting the same tracks by buses and trains (or building many parallel tracks).

## 2 Preliminaries

We use  $\mathbb{N}$  to denote  $\{0, 1, 2, \dots\}$ . For  $n \in \mathbb{N}$ , let  $[n] = \{1, 2, 3, \dots, n\}$ , and  $[n]_0 = [n] \cup \{0\}$ .

**Two-layered graphs.** Whenever context is clear, denote the vertex bipartition of a two-layered graph  $G$  (given by the two-layer drawing) by  $(X, Y)$ . Let  $n_X = |X|$  and  $n_Y = |Y|$ . For  $i \in [n_X]$ , let  $x_i$  be the  $i$ th vertex of  $X$ , and for  $j \in [n_Y]$ , let  $y_j$  be the  $j$ th vertex of  $Y$ . Moreover, we say that  $i$  is the index of the vertex  $x_i$ , and  $j$  is the index of the vertex  $y_j$ ; we write  $\text{index}(x_i) = i$  and  $\text{index}(y_j) = j$ . Similarly, let  $X_i$  denote  $\{x_r \mid 1 \leq r \leq i\}$ , and let  $Y_j$  denote  $\{y_r \mid 1 \leq r \leq j\}$ . For  $i, j \in [n_X]$ , where  $i \leq j$ , the set  $X_{i,j}$  denotes the set  $\{x_p \mid i \leq p \leq j\}$ . Moreover, if  $i < j$ , then  $X_{j,i} = \emptyset$ . The set  $Y_{i,j}$  is defined analogously for  $i, j \in [n_Y]$ . A *crossing* in  $G$  is a pair of edges intersecting at a point other than their possible common endpoints. Note that two edges  $x_i y_j$  and  $x_r y_s$ , where  $i, r \in [n_X]$  and  $j, s \in [n_Y]$ , form a crossing (or, cross each other) if and only if  $r > i, j > s$  or  $i > r, s > j$ . For a subgraph  $H$  of  $G$ ,  $\text{cr}(H)$  denotes the number of crossings in  $H$ . Similarly, for a set of edges  $E' \subseteq E(G)$ ,  $\text{cr}(E')$  denotes the number of crossings in the subgraph induced by  $E'$ .

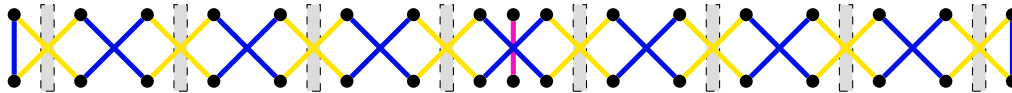
For an introduction to parameterized complexity and kernelization, see [12, 14, 20].

## 3 NP-hardness for CM-PM

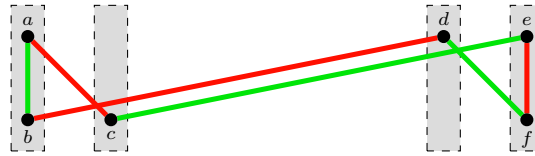
We show that CM-PM is NP-hard via a reduction from VERTEX COVER (known to be NP-hard from [35]). The VERTEX COVER problem takes as input a graph  $H$  and an integer  $l$ , and the goal is to check if there is  $S \subseteq V(H)$  of size at most  $l$ , such that  $H - S$  has no edges.

Let  $(H, l)$  be an instance of VERTEX COVER. We create a corresponding instance  $(G, k)$  of CM-PM. The general idea behind the reduction is to use two gadgets. The first one is created for every vertex of  $H$ . There are two possible perfect matchings in each copy of the gadget. Selecting one of these matchings corresponds to choosing whether the vertex belongs to the vertex cover or not. The second gadget is created for every edge of  $H$ . There are also two possible perfect matchings in each copy of the gadget, corresponding to orienting the edge in one of the two possible directions. The details of the construction ensure that the number of crossings is minimized when each edge is oriented towards a selected vertex and the number of selected vertices is minimal. We assume without loss of generality that the two straight line segments are parallel and horizontal. Vertex gadgets are aligned in such a way that there are no crossings between them, but each of them defines regions between the vertices, called slots, that are used to anchor edge gadgets. The heart of the argument is a careful analysis of the number of crossings between different gadgets.

For any integer  $s \geq 1$ , the *vertex gadget of size  $s$*  is a cycle on  $8s$  vertices together with a path on 2 vertices, positioned as shown in Figure 2. The vertex gadget defines  $2s$  slots. The slots are spaces between the vertices, whose exact location is marked in Figure 2 using gray rectangles. The ones to the left of the pink edge are called *left slots* and the ones to the right are called *right slots*. Furthermore, observe that there are only two ways to choose a perfect matching in this gadget: either take all the blue edges and the pink edge in the middle, or take all the yellow edges and the pink one. We interpret choosing the blue (yellow) matching as selecting (not selecting) the vertex to the vertex cover.



■ **Figure 2** The vertex gadget of size 4, with 8 slots colored gray.



■ **Figure 3** The edge gadget and the placement of its vertices in slots.

We fix an ordering  $v_1, \dots, v_n$  on the vertices of  $H$ . For every  $v \in V(H)$ , we create a copy of the vertex gadget of size  $2d(v)$ . We arrange the gadgets on the two line segments in such a way that each gadget occupies a separate range of the  $x$  axis and for every  $i < j$ , the gadget for  $v_i$  is to the left of the gadget for  $v_j$ .

We process the edges in any order. For every  $v_i v_j \in E(H)$ , where  $i < j$ , we select two first unselected right slots in the gadget for  $v_i$  and two first unselected left slots in the gadget for  $v_j$ . The *edge gadget* for  $v_i v_j$  is a cycle on 6 vertices that are labeled and arranged on the line segments as shown in Figure 3. These vertices are carefully placed in the slots as follows: vertices  $a$  and  $b$  in the first of the two selected slots of the gadget for  $v_i$ , vertex  $c$  in the second of these two slots, vertex  $d$  in the first of the two selected slots of the gadget for  $v_j$ , and vertices  $e$  and  $f$  in the second of these two slots. The edge gadget admits two different perfect matchings. We interpret choosing the green (red) matching as orienting the edge towards  $v_i$  ( $v_j$ ). For a complete example of the reduction for a small graph, see Figure 4.

Now by setting the “budget” for the number of edge crossings appropriately, we can obtain the following theorem (we refer to the full version for a detailed analysis).

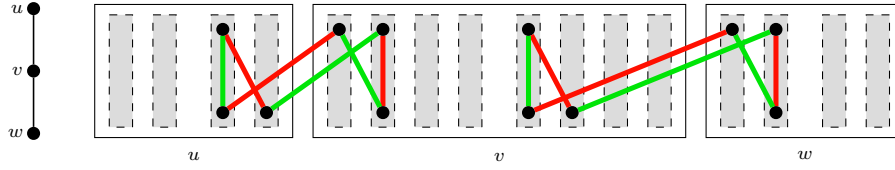
► **Theorem 1.** *CM-PM is NP-hard, even if the maximum degree of the input graph is 2.*

## 4 FPT Algorithm for CM-PM

Let  $(G, k)$  be an instance of CM-PM, with vertex bipartition  $X$  and  $Y$ , where  $|X| = |Y| = n$ . (Here, we note that if  $|X| \neq |Y|$  then  $(G, k)$  is a no-instance as it does not admit a perfect matching.) We will design an FPT algorithm for CM-PM running in time  $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ . Our algorithm will be a DP algorithm which processes the graph from left to right. That is to say, for each  $i = 1, 2, \dots, n$ , at stage  $i$ , we consider the graph  $G_i = G[X_i \cup Y_i]$ , the graph induced by  $\{x_1, \dots, x_i, y_1, \dots, y_i\}$ , and solve a family of subproblems, the solution for one of which will lead to an optimal solution for the entire graph  $G$ . We will bound the number of sub-instances that we need to solve at each stage  $i$ , for  $i \in [n]$ , by  $2^{\mathcal{O}(\sqrt{k})}$ . To achieve the above, we will use a well-known result on the number of partitions of an integer, (which says that the number of partitions of an integer  $k$  is  $2^{\mathcal{O}(\sqrt{k})}$ ). (For the integer 6, a partition of it is  $1 + 2 + 3$ .) We will rely on the fact that for a number  $t$ , we can compute all its partitions in time bounded by  $2^{\mathcal{O}(\sqrt{t})}$ . This bound will be crucial for achieving the running time.

We first explain the intuition behind our algorithm. Suppose  $(G, k)$  is a yes-instance and let  $M$  be a perfect matching of  $G$  with  $\text{cr}(M) \leq k$ . Fix  $i \in [n]$ . Consider how  $M$  saturates the “future vertices,” i.e., vertices in  $X_{i+1,n} \cup Y_{i+1,n}$ . Consider a future vertex, say  $x_j$  for some  $j > i$ . Using the fact that  $\text{cr}(M) \leq k$ , we will show that  $M$  cannot match  $x_j$  to a vertex

7:10 Connecting the Dots (with Minimum Crossings)



■ **Figure 4** A graph  $H$  and the two-layered graph  $G$  obtained by passing  $H$  to the reduction algorithm, vertex gadgets presented schematically.

in  $Y_{i-k}$ . Therefore, the only vertices in  $X_i \cup Y_i$  that can possibly be matched to vertices in the future belong to  $X_{i-k+1} \cup Y_{i-k+1}$ . In other words, while doing a DP from left to right, by the time we get to stage  $i$ , the intersection of the potential solution with  $X_{i-k} \cup Y_{i-k}$  is completely determined. This observation suggests the most obvious strategy: at stage  $i$ , “guess” how the solution matches (and saturates) the vertices in  $X_{i-k+1,i} \cup Y_{i-k+1,i}$ . But this strategy will only lead to an algorithm running in time  $k^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ . Observe that since we are only interested in a matching with the least possible number of crossings, we need not look at all possible matchings in  $G[X_{i-k+1,i} \cup Y_{i-k+1,i}]$ . We only need to look at which subsets of  $X_{i-k+1,i}$  and  $Y_{i-k+1,i}$  are saturated by  $M$ . Thus, from each collection of matchings that saturate the same subset of  $X_{i-k+1,i} \cup Y_{i-k+1,i}$ , we remember the matching that incurs the least number of crossings. This observation can be used to obtain an algorithm running in time  $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ . To further improve this running time, we show that the number of subsets of  $X_{i-k+1,i} \cup Y_{i-k+1,i}$  that are not saturated by the intersection of any potential solution with  $X_i \cup Y_i$  cannot exceed  $2^{\mathcal{O}(\sqrt{k})}$ . (This is where we will use the bound that the number of partitions of an integer  $t$  is bounded by  $2^{\mathcal{O}(\sqrt{t})}$ .) This will lead us to an algorithm with the claimed running time for the problem.

We start by giving some notations and preliminary results that will be helpful later. We let  $\text{Sat}(M) = \{u, v \mid uv \in M\}$ . That is,  $\text{Sat}(M)$  is the set of vertices saturated by  $M$  in  $G$ .

**Partitions of an integer.** For a positive integer  $\alpha$ , a partition of  $\alpha$  refers to writing  $\alpha$  as a sum of positive integers (greater than zero), where the order of the summands is immaterial. Each summand in such a sum is called a *part* of  $\alpha$ . For example,  $16 = 1 + 4 + 4 + 7$  is a partition of 16. Note that here two of the parts (the two 4s) are the same. We, however, are interested in only those partitions of  $\alpha$  in which the parts are all distinct. Let us call such partitions *distinct-part partitions*. For example,  $\{1, 2, 6, 7\}$  is a distinct-part partition of 16. It is known that the number of partitions (and hence the number of distinct-part partitions) of an integer  $k$  is bounded by  $2^{\mathcal{O}(\sqrt{k})}$  [26]. In light of this result, it is not difficult to see that given an integer  $k$ , all distinct-part partitions of  $k$  can be generated in time  $2^{\mathcal{O}(\sqrt{k})}$ . For future reference, we state these results below.

► **Lemma 2.** *The number of distinct-part partitions of any positive integer  $k$ , is at most  $2^{\mathcal{O}(\sqrt{k})}$ . Moreover, we can generate all of these distinct-part partitions in time  $2^{\mathcal{O}(\sqrt{k})}$ .*

**Some important sets for the algorithm.** For  $i \in [n]$ , we let  $\widehat{X}_i = \{x_{i-k+\ell} \mid \ell \in [k] \text{ and } i - k + \ell \geq 1\}$  and  $\widehat{Y}_i = \{y_{i-k+\ell} \mid \ell \in [k] \text{ and } i - k + \ell \geq 1\}$  (see Figure 5). We will argue that in any perfect matching  $M$  in  $G$  with  $\text{cr}(M) \leq k$ , the vertices from  $X_i$  which are matched to a vertex  $y_s$ , with  $s \geq i + 1$ , belong to the set  $\widehat{X}_i$ . Similarly, we can argue that  $\widehat{Y}_i$  is the set of vertices from  $Y_i$  which can possibly be matched to vertices  $x_s$ , with  $s \geq i + 1$ .

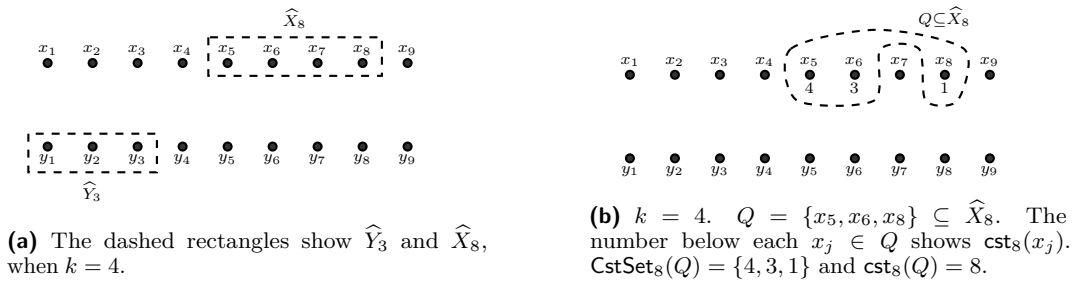


Figure 5 An example of  $\widehat{X}_i, \widehat{Y}_i, Q \subseteq \widehat{X}_i, \text{CstSet}_i(Q)$  and  $\text{cst}_i(Q)$ .

We will now associate costs to vertices (and subsets) of  $\widehat{X}_i$  (resp.  $\widehat{Y}_i$ ), which will be helpful in obtaining lower bounds on the number of crossings, when vertices from  $\widehat{X}_i$  (resp.  $\widehat{Y}_i$ ) are matched to vertices  $y_s$  (resp.  $x_s$ ), where  $s \geq i + 1$ . To this end, consider  $i \in [n]$  and a vertex  $x_r \in \widehat{X}_i$ . We let  $\text{cst}_i(x_r) = i + 1 - r$ . Since  $x_r \in \widehat{X}_i$ , we have  $r \leq i$ , and thus,  $\text{cst}_i(x_r) \geq 1$ . For a subset  $Q \subseteq \widehat{X}_i$ , we let  $\text{CstSet}_i(Q) = \{\text{cst}_i(x) \mid x \in Q\}$  and  $\text{cst}_i(Q) = \sum_{x \in Q} \text{cst}_i(x)$ . Similarly, for  $i \in [n]$  and a vertex  $y_r \in \widehat{Y}_i$ , we let  $\text{cst}_i(y_r) = i + 1 - r \geq 1$ . Moreover, for a subset  $Q \subseteq \widehat{Y}_i$ , we let  $\text{CstSet}_i(Q) = \{\text{cst}_i(y) \mid y \in Q\}$  and  $\text{cst}_i(Q) = \sum_{y \in Q} \text{cst}_i(y)$ . We note that, for each  $i \in [n]$ , we have  $\text{cst}_i(\emptyset) = 0$ . In order to understand the intuition behind these definitions, look at the  $i$ th stage in our algorithm. At stage  $i$ , we consider the graph  $G[X_i \cup Y_i]$ . Consider the vertices in  $\widehat{X}_i$  that are matched to vertices in the future (i.e., vertices  $y_s$  where  $s > i$ ). Note that if  $x_i$  gets matched to a future vertex, then  $x_i$  participates in at least one crossing (in the final solution), and if  $x_{i-1}$  gets matched to a future vertex, then  $x_{i-1}$  participates in at least two crossings and so on. In particular,  $x_r \in \widehat{X}_i$ , if matched to a future vertex participates in at least  $i + 1 - r$  crossings. So,  $\text{cst}_i(x_r)$  is a lower bound on the number of crossings in which  $x_r$  participates (or cost incurred by  $x_r$ ) if it gets matched to a future vertex. For a set  $Q \subseteq \widehat{X}_i$ ,  $\text{CstSet}_i(Q)$  is the set of minimum costs incurred by each element of  $Q$ . Moreover,  $\text{cst}_i(Q)$  is the cost incurred by  $Q$  if all its elements get matched to future vertices. Now using the notion of distinct-part partitions of an integer, we introduce some “special” sets of subsets of  $X$  and  $Y$ , respectively. These sets will be crucially used while creating the sub-instances in our DP algorithm. For  $\alpha \in [k]$ , let  $\mathcal{P}_\alpha$  be the set of all distinct-part partitions of  $\alpha$ . Furthermore, let  $\mathcal{P}_{\leq k} = \cup_{\alpha \in [k]} \mathcal{P}_\alpha$ . From Lemma 2, we have  $|\mathcal{P}_{\leq k}| = 2^{\mathcal{O}(\sqrt{k})}$ . Consider  $i \in [n]$ ,  $\alpha \in [k]$ , and  $P \in \mathcal{P}_{\leq \alpha}$ . We let  $S_X^i(P) = \{x_{i+1-\beta} \mid \beta \in P \text{ and } i + 1 - \beta \geq 1\}$ . (For example, for  $P = \{1, 2, 6, 7, 8\}$  and  $i = 6$ , we have  $S_X^i(P) = \{x_6, x_5, x_1\}$ .) Note that  $S_X^i(P) \subseteq \widehat{X}_i$ ,  $\text{CstSet}_i(S_X^i(P)) = P$ , and  $\text{cst}_i(S_X^i(P)) = \alpha$ , where  $P$  is a partition of  $\alpha \in [k]$ . Similarly, we define  $S_Y^i(P) = \{y_{i+1-\beta} \mid \beta \in P \text{ and } i + 1 - \beta \geq 1\} \subseteq \widehat{Y}_i$ . Again, note that  $\text{CstSet}_i(S_Y^i(P)) = P$  and  $\text{cst}_i(S_Y^i(P)) = \alpha$ . We let  $\mathcal{S}_X^i = \{S_X^i(P) \mid P \in \mathcal{P}_{\leq k}\} \cup \{\emptyset\} \subseteq \mathcal{2}^{\widehat{X}_i}$  and  $\mathcal{S}_Y^i = \{S_Y^i(P) \mid P \in \mathcal{P}_{\leq k}\} \cup \{\emptyset\} \subseteq \mathcal{2}^{\widehat{Y}_i}$ .

► **Lemma 3.** *The families  $\mathcal{S}_X^i$  and  $\mathcal{S}_Y^i$  contain at most  $|\mathcal{P}_{\leq k}| + 1 = 2^{\mathcal{O}(\sqrt{k})}$  sets each. Moreover, for each  $i \in [n]$ , the families  $\mathcal{S}_X^i$  and  $\mathcal{S}_Y^i$  can be generated in  $2^{\mathcal{O}(\sqrt{k})}$  time.*

We associate a set of integers to every pair  $(S, S') \in \mathcal{S}_X^i \times \mathcal{S}_Y^i$ , for each  $i \in [n]$ . These sets will give the “allowed” number of crossings for a matching in the graph  $G_i$ . Consider  $i \in [n]$ ,  $S \in \mathcal{S}_X^i$ , and  $S' \in \mathcal{S}_Y^i$ . We let  $\text{Alw}_i(S, S') = \{\ell \in [k]_0 \mid \ell \leq k - \max\{\text{cst}_i(S), \text{cst}_i(S')\}\}$ .

► **Observation 4.** *Consider  $i \in [n] \setminus \{1\}$ . For  $S \in \mathcal{S}_X^i$  and  $Q \subseteq S \setminus \{x_i\}$ , we have  $Q \in \mathcal{S}_X^{i-1}$ . Similarly, for  $S' \in \mathcal{S}_Y^i$  and  $Q' \subseteq S' \setminus \{y_i\}$ , we have  $Q' \in \mathcal{S}_Y^{i-1}$ .*

## 7:12 Connecting the Dots (with Minimum Crossings)

► **Observation 5.** Consider  $i \in [n] \setminus \{1\}$ . For  $S \in \mathcal{S}_X^i$  and  $Q \subseteq S \setminus \{x_i\}$ , we have  $\text{cst}_{i-1}(Q) \leq \text{cst}_i(S) - |S|$ . Similarly, for  $S' \in \mathcal{S}_Y^i$  and  $Q' \subseteq S' \setminus \{x_i\}$ , we have  $\text{cst}_{i-1}(Q') \leq \text{cst}_i(S') - |S'|$ .

We now define the notion of a “compatible matching.” Consider  $i \in [n]$ ,  $S \in \mathcal{S}_X^i$ , and  $S' \in \mathcal{S}_Y^i$ . We say that a matching  $M$  in  $G_i$  is  $(i, S, S')$ -compatible if  $S = \widehat{X}_i \setminus \text{Sat}(M)$ ,  $S' = \widehat{Y}_i \setminus \text{Sat}(M)$ , and  $\text{cr}(M) \leq k - \max\{\text{cst}_i(S), \text{cst}_i(S')\}$ . Compatible matchings will be helpful in establishing the correctness of our algorithm, in which we will be considering matchings of  $G_i$  that saturate exactly  $(X_i \cup Y_i) \setminus (S \cup S')$ , while incurring at most a certain allowed number of crossings. Suppose at the  $i$ th stage of our algorithm, we consider a matching, say  $M_i$ , of  $G_i$  that does not saturate  $S$ . We would like to extend  $M_i$  to a matching of  $G$  with at most  $k$  crossings. That is, at stage  $i$ ,  $M_i$  matches  $S$  to future vertices. Therefore, while extending  $M_i$  to a matching of the entire graph  $G$ , we will incur at least  $\text{cst}_i(S)$  more crossings (in addition to  $\text{cr}(M_i)$ ). Therefore, in order to be able to extend  $M_i$  to matching of  $G$  with at most  $k$  crossings,  $\text{cr}(M_i)$  cannot exceed  $k - \text{cst}_i(S)$ . Identical reasoning holds for the set  $S'$ .

We are now ready to define the states of our DP table. For each  $i \in [n]$ ,  $S \in \mathcal{S}_X^i$  and  $S' \in \mathcal{S}_Y^i$  with  $|S| = |S'|$ , and an integer  $\ell \in \text{Alw}_i(S, S') = \{\ell \in [k]_0 \mid \ell \leq k - \max\{\text{cst}_i(S), \text{cst}_i(S')\}\}$ , we define

$$T[i, S, S', \ell] = \begin{cases} 1, & \text{if there is a matching } M \text{ in } G_i, \text{ such that } \text{cr}(M) = \ell \text{ and} \\ & \text{Sat}(M) = (X_i \setminus S) \cup (Y_i \setminus S'), \\ 0, & \text{otherwise.} \end{cases}$$

Observe that  $(G, k)$  is a yes-instance of CM-PM if and only if there is  $\ell \in [k]_0$ , such that  $T[n, \emptyset, \emptyset, \ell] = 1$ . A matching  $M$  in  $G_i$  is said to *realize*  $T[i, S, S', \ell]$ , if  $\text{cr}(M) = \ell$  and  $M$  is  $(i, S, S')$ -compatible. In the above we note that  $\ell \leq k - \max\{\text{cst}_i(S), \text{cst}_i(S')\}$ , as  $\ell \in \text{Alw}_i(S, S')$ . Let us now see how  $T[i, S, S', \ell]$  can be computed.

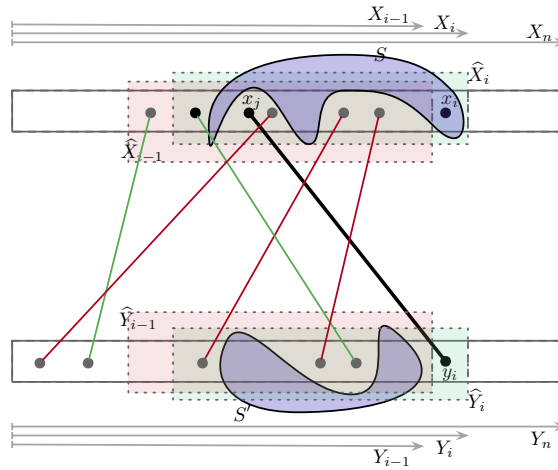
**Base Case.** Consider the entry  $T[1, S, S', \ell]$ . Note that  $\text{cr}(G_1) = 0$ . Thus, if  $\ell > 0$ , we have  $T[1, S, S', \ell] = 0$ . Now we consider the case when  $\ell = 0$ . Recall that by definition, we have  $|S| = |S'|$ . If  $S = \{x_1\}$  and  $S' = \{y_1\}$ , then we should not match any vertex. Thus, we have a matching (which is the empty set) with 0 crossings, and thus,  $T[1, S, S', \ell] = 1$ . Otherwise, we have  $S = S' = \emptyset$ . Note that the only possible matching in the graph  $G[\{x_1, y_1\}]$  is  $\{x_1 y_1\}$ . So, if  $x_1 y_1 \in E(G)$ , then  $\{x_1 y_1\}$  is a matching with 0 crossings, and hence  $T[1, S, S', \ell] = 0$ . Otherwise, we have  $x_1 y_1 \notin E(G)$ , and hence  $T[1, S, S', \ell] = 0$ .

We now move to our recursive formulae for the computation of the entries of our table. We set the value of  $T[i, S, S', \ell]$  (recursively) based on the following cases, where  $i > 1$ .

**Case 1:  $x_i \in S$  and  $y_i \in S'$ .** From Observation 4, we have that  $S \setminus \{x_i\} \in \mathcal{S}_X^{i-1}$  and  $S' \setminus \{y_i\} \in \mathcal{S}_Y^{i-1}$ . Also, from Observation 5 it follows that  $\ell \in \text{Alw}_{i-1}(S \setminus \{x_i\}, S' \setminus \{y_i\})$ . We set  $T[i, S, S', \ell] = T[i-1, S \setminus \{x_i\}, S' \setminus \{y_i\}, \ell]$ .

► **Lemma 6.** The computation of  $T[i, S, S', \ell]$  in Case 1 is correct.

**Case 2:  $x_i \in S$  and  $y_i \notin S'$ , or  $x_i \notin S$  and  $y_i \in S'$ .** We will only argue for the case when  $x_i \in S$  and  $y_i \notin S'$ . (The other case can be handled symmetrically.) Thus, hereafter we assume that  $x_i \in S$  and  $y_i \notin S'$ . In this case, a matching, say  $M$ , which realizes  $T[i, S, S', \ell]$ , must saturate the vertex  $y_i$  and must not saturate the vertex  $x_i$ . Thus,  $M$  must have an edge  $x_j y_i$ , where  $j < i$  (here we rely on the fact that  $y_i$  cannot be matched to  $x_i$ , as  $x_i \in S$ ).



■ **Figure 6** An illustration of the edges intersecting  $x_j y_i$ , where  $x_j \in \widehat{X}_{i-1} \setminus S$ . Here, the red edges intersect  $x_j y_i$  and the green edges do not intersect  $x_j y_i$ .

As  $M$  must satisfy the constraint  $\text{cr}(M) = \ell \leq k$ , we must have  $i - k \leq j < i$ . That is, the vertex to which  $y_i$  is matched, must belong to the set  $\widehat{X}_{i-1}$ . We will construct a set  $\mathcal{Q} \subseteq \mathcal{S}_X^{i-1} \subseteq 2^{\widehat{X}_{i-1}}$ . This set will be used for creating sub-instances whose values are needed for the computation of  $T[i, S, S', \ell]$ . Intuitively speaking, each set in  $\mathcal{Q}$  will determine a vertex to which  $y_i$  is matched, in the matching that we are seeking for. Note that as  $y_i$  must be saturated by any matching that realizes (or complies) with  $T[i, S, S', \ell]$ , the edge, say  $\widehat{x}y_i$  in the matching might intersect other edges of the matching. Therefore, we will have to account for this extra overhead in the number of crossing edges. To count these extra crossings incurred, we will define an “overhead” function.

To construct  $\mathcal{Q}$ , we first construct two sets  $\widehat{\mathcal{Q}}, \widetilde{\mathcal{Q}} \subseteq 2^{\widehat{X}_{i-1}}$  (each of size at most  $\mathcal{O}(k)$ ). We will obtain  $\widehat{\mathcal{Q}} \supseteq \widetilde{\mathcal{Q}} \supseteq \mathcal{Q}$  (in that order, by removing some “bad sets”). For a vertex  $x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S$ , let  $Q_j = (S \setminus \{x_i\}) \cup \{x_j\}$ . Intuitively, the vertex  $y_i$  will be matched to  $x_j$ , when  $Q_j$  is under consideration. Note that  $Q_j \subseteq \widehat{X}_{i-1}$ . We let  $\widehat{\mathcal{Q}} = \{Q_j \mid x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S\}$ . In the above definition, we only consider the neighbors of  $y_i$  from  $\widehat{X}_{i-1} \setminus S$ , because we require that the desired matching must not saturate a vertex from  $S$ . We let  $\widetilde{\mathcal{Q}} = \widehat{\mathcal{Q}} \cap \mathcal{S}_X^{i-1}$ . We now define a function  $\text{ovh} : \widetilde{\mathcal{Q}} \rightarrow \mathbb{N}$  (see Figure 6 for an intuitive illustration). For  $Q_j \in \widetilde{\mathcal{Q}}$ , we set  $\text{ovh}(Q_j) = |X_{j+1, i} \setminus S|$ . To obtain  $\mathcal{Q}$ , we will delete those sets from  $\widetilde{\mathcal{Q}}$  which will incur an “overhead” of crossings more than the “allowed” budget. Before constructing  $\mathcal{Q}$ , we first recall the following facts. By the definition of  $\widetilde{\mathcal{Q}}$ , we have  $Q \in \mathcal{S}_X^{i-1}$ . Moreover, from Observation 4 it follows that  $S' \in \mathcal{S}_Y^{i-1}$  (as  $y_i \notin S'$ ). We set  $\mathcal{Q} = \{Q \in \widetilde{\mathcal{Q}} \mid \ell - \text{ovh}(Q) \in \text{Alw}_{i-1}(Q, S')\}$ . Now we set  $T[i, S, S', \ell]$  as follows.

$$T[i, S, S', \ell] = \begin{cases} 0, & \text{if } \mathcal{Q} = \emptyset, \\ \bigvee_{Q \in \mathcal{Q}} T[i-1, Q, S', \ell - \text{ovh}(Q)], & \text{otherwise.} \end{cases}$$

► **Lemma 7.** *The computation of  $T[i, S, S', \ell]$  in Case 2 is correct.*

**Case 3:  $x_i \notin S$  and  $y_i \notin S'$ .** In this case, a matching, say  $M$ , which realizes  $T[i, S, S', \ell]$ , must saturate both the vertices  $x_i$  and  $y_i$ . Thus,  $M$  must have edges  $x_j y_i$  and  $x_i y_{j'}$ , where  $j \leq i$  and  $j' \leq i$ . (Assuming  $x_i$  is adjacent to  $y_i$  in  $G$ , it can be the case that  $j = j' = i$ , in which case  $x_i y_i \in M$ .) We will thus have  $T[i, S, S', \ell] = T_1[i, S, S', \ell] \vee T_2[i, S, S', \ell]$ , where  $T_1[i, S, S', \ell]$  and  $T_2[i, S, S', \ell]$  are boolean variables that correspond respectively to the cases  $j = j' = i$  and  $j \neq i$  (and  $j' \neq i$ ). We now define  $T_1[i, S, S', \ell]$  and  $T_2[i, S, S', \ell]$ , formally.



**Defining  $T_1[i, S, S', \ell]$ .** Since  $x_i \notin S$ , we have  $S \subseteq \widehat{X}_{i-1}$ . As  $y_i \notin S'$ , we have  $S' \subseteq \widehat{Y}_{i-1}$ . By Observation 4,  $S \in \mathcal{S}_X^{i-1}$  and  $S' \in \mathcal{S}_Y^{i-1}$ . Note that if a matching  $M$  that realizes  $T[i, S, S', \ell]$  contains the edge  $x_i y_i$  (assuming  $x_i y_i$  is indeed an edge in the graph  $G$ ), then  $\text{cr}(M) = \text{cr}(M \setminus \{x_i y_i\})$ . That is, no additional crossing is incurred by adding the edge  $x_i y_i$  to the matching  $M \setminus \{x_i y_i\}$ . Also, note that  $\ell \in \text{Alw}_{i-1}(S, S')$ . With these observations, we define  $T_1[i, S, S', \ell]$  as follows.

$$T_1[i, S, S', \ell] = \begin{cases} 0, & \text{if } x_i y_i \notin E(G), \\ T[i-1, S, S', \ell], & \text{otherwise.} \end{cases}$$

**Defining  $T_2[i, S, S', \ell]$ .** Now, to define  $T_2[i, S, S', \ell]$ , we proceed as in Case 2. For a vertex  $x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S$ , let  $Q_j = S \cup \{x_j\}$ . We let  $\widehat{Q} = \{Q_j \mid x_j \in (N(y_i) \cap \widehat{X}_{i-1}) \setminus S\}$ , and  $\mathcal{Q} = \widehat{Q} \cap \mathcal{S}_X^{i-1}$ . Similarly, for a vertex  $y_{j'} \in (N(x_i) \cap \widehat{Y}_{i-1}) \setminus S'$ , let  $R_{j'} = S' \cup \{y_{j'}\}$ . We let  $\widehat{R} = \{R_{j'} \mid y_{j'} \in (N(x_i) \cap \widehat{Y}_{i-1}) \setminus S'\}$ , and  $\mathcal{R} = \widehat{R} \cap \mathcal{S}_Y^{i-1}$ . We will now construct a set of ‘‘crucial pairs’’ from  $\mathcal{Q} \times \mathcal{R}$ , for the computation of  $T_2[i, S, S', \ell]$ . Towards this, we define a function  $\text{ovh} : \mathcal{Q} \times \mathcal{R} \rightarrow \mathbb{N}$ . We set  $\text{ovh}(Q_j, R_{j'}) = |X_{j+1, i} \setminus S| + |Y_{j'+1, i} \setminus S'| - 1$ , for  $Q_j \in \mathcal{Q}$  and  $R_{j'} \in \mathcal{R}$ . Finally, we let  $\mathcal{C} = \{(Q, R) \in \mathcal{Q} \times \mathcal{R} \mid \ell - \text{ovh}(Q, R) \in \text{Alw}_{i-1}(Q, R)\}$ . Now we set  $T_2[i, S, S', \ell]$  as follows.

$$T_2[i, S, S', \ell] = \begin{cases} 0, & \text{if } \mathcal{C} = \emptyset, \\ \bigvee_{(Q, R) \in \mathcal{C}} T[i-1, Q, R, \ell - \text{ovh}(Q, R)], & \text{otherwise.} \end{cases}$$

► **Lemma 8.** *The computation of  $T[i, S, S', \ell]$  in Case 3 is correct.*

As observed earlier,  $(G, k)$  is a yes-instance of CM-PM if and only if there is  $\ell \in [k]_0$ , such that  $T[n, \emptyset, \emptyset, \ell] = 1$ . Note that for each  $i \in [n]$ ,  $S \in \mathcal{S}_X^i$ ,  $S' \in \mathcal{S}_Y^i$ , and  $\ell \in \text{Alw}_i(S, S')$ , we can compute the entry  $T[i, S, S', \ell]$  in time bounded by  $n^{\mathcal{O}(1)}$ . Moreover, the number of entries in our table is bounded by  $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$  (see Lemma 3). Thus, the running time of the algorithm is bounded by  $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ . The correctness of the algorithm follows from the correctness of base case and recursive formulae. Thus, we obtain the following theorem.

► **Theorem 9.** *CM-PM admits an algorithm running in time  $2^{\mathcal{O}(\sqrt{k})} n^{\mathcal{O}(1)}$ .*

---

## References

- 1 A. Karim Abu-Affash, Ahmad Biniiaz, Paz Carmi, Anil Maheshwari, and Michiel H. M. Smid. Approximating the bottleneck plane perfect matching of a point set. *Comput. Geom.*, 48(9):718–731, 2015.
- 2 A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Yohai Trabelsi. Bottleneck non-crossing matching in the plane. *Comput. Geom.*, 47(3):447–457, 2014.
- 3 Jihad Al-Oudatallah, Fariz Abboud, Mazen Khoury, and Hassan Ibrahim. Overlapping Signal Separation Method Using Superresolution Technique Based on Experimental Echo Shape. *Advances in Acoustics and Vibration*, pages 1–9, 2017.
- 4 Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting crossing-free structures. In *Symposium on Computational Geometry 2012, SoCG '12, Chapel Hill, NC, USA, June 17-20, 2012*, pages 61–68, 2012.
- 5 Marc Benkert, Herman J. Haverkort, Moritz Kroll, and Martin Nöllenburg. Algorithms for Multi-criteria One-Sided Boundary Labeling. In *Graph Drawing, 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007. Revised Papers*, pages 243–254, 2007.
- 6 Therese C. Biedl, Franz-Josef Brandenburg, and Xiaotie Deng. Crossings and Permutations. In *Proceeding of the 13th International Symposium on Graph Drawing, GD*, volume 3843 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.



- 7 Édouard Bonnet, Tillmann Miltzow, and Paweł Rzażewski. Complexity of Token Swapping and Its Variants. *Algorithmica*, October 2017.
- 8 Sergio Cabello and Bojan Mohar. Adding One Edge to Planar Graphs Makes Crossing Number and 1-Planarity Hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013.
- 9 John Gunnar Carlsson, Benjamin Armbruster, Saladi Rahul, and Haritha Bellam. A Bottleneck Matching Problem with Edge-Crossing Constraints. *Int. J. Comput. Geometry Appl.*, 25(4):245–262, 2015.
- 10 Xuanwu Chen and Ming S. Lee. A case study on multi-lane roundabouts under congestion: Comparing software capacity and delay estimates with field data. *Journal of Traffic and Transportation Engineering (English Edition)*, 3(2):154–165, 2016.
- 11 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.
- 12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.
- 13 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.
- 14 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
- 15 Vida Dujmovic, Michael R. Fellows, Michael T. Hallett, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Matthew Suderman, Sue Whitesides, and David R. Wood. On the Parameterized Complexity of Layered Graph Drawing. In *9th Annual European Symposium on Algorithms, ESA 2001, Proceedings*, pages 488–499, 2001.
- 16 Peter Eades and Sue Whitesides. Drawing graphs in two layers. *Theoretical Computer Science*, 131(2):361–374, 1994.
- 17 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.
- 18 Jack Edmonds. Paths, Trees and Flowers. *Canadian Journal of Mathematics*, pages 449–467, 1965.
- 19 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical computer science*, 410(1):53–61, 2009.
- 20 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.
- 21 Per Garder. Pedestrian safety at traffic signals: A study carried out with the help of a traffic conflicts technique. *Accident Analysis & Prevention*, 21(5):435–444, 1989.
- 22 M R Garey and D S Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman, New York, 1979.
- 23 Michael R Garey and David S Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 4(3):312–316, 1983.
- 24 Martin Grohe. Computing crossing numbers in quadratic time. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 231–236, 2001.
- 25 Magnús M. Halldórsson, Christian Knauer, Andreas Spillner, and Takeshi Tokuyama. Fixed-Parameter Tractability for Non-Crossing Spanning Trees. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 410–421, 2007.
- 26 Godfrey H Hardy and Srinivasa Ramanujan. Asymptotic formulæ in combinatory analysis. *Proceedings of the London Mathematical Society*, 2(1):75–115, 1918.
- 27 Petr Hliněný. Crossing number is hard for cubic graphs. *J. Comb. Theory, Ser. B*, 96(4):455–471, 2006.

## 7:16 Connecting the Dots (with Minimum Crossings)

- 28 Petr Hlinený and Marek Dernár. Crossing Number is Hard for Kernelization. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 42:1–42:10, 2016.
- 29 John E. Hopcroft and Robert Endre Tarjan. Efficient Algorithms for Graph Manipulation [H] (Algorithm 447). *Commun. ACM*, 16(6):372–378, 1973.
- 30 Klaus Jansen and Gerhard J. Woeginger. The Complexity of Detecting Crossingfree Configurations in the Plane. *BIT*, 33(4):580–595, 1993.
- 31 Michael Junger and Petra Mutzel. *Graph Drawing Software*. Springer-Verlag, Berlin, Heidelberg, 2003.
- 32 Ken-ichi Kawarabayashi and Bruce A. Reed. Computing crossing number in linear time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 382–390, 2007.
- 33 Fabian Klute and Martin Nöllenburg. Minimizing Crossings in Constrained Two-Sided Circular Graph Layouts. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 53:1–53:14, 2018.
- 34 Jan Kratochvíl, Anna Lubiw, and Jaroslav Nešetřil. Noncrossing Subgraphs in Topological Layouts. *SIAM J. Discret. Math.*, 4(2):223–244, March 1991.
- 35 Mukkai S Krishnamoorthy and Narsingh Deo. Node-deletion NP-complete problems. *SIAM Journal on Computing*, 8(4):619–625, 1979.
- 36 Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: An information aesthetic for comparative genomics. *Genome Research*, 19(9):1639–1645, 2009.
- 37 J. Malik, J. Weber, Q. T. Luong, and D. Roller. Smart cars and smart roads. In *Proceedings 6th. British Machine Vision Conference*, pages 367–381, 1995.
- 38 Dániel Marx and Tillmann Miltzow. Peeling and Nibbling the Cactus: Subexponential-Time Algorithms for Counting Triangulations and Related Problems. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 52:1–52:16, 2016.
- 39 Tillmann Miltzow. Subset token swapping on a path and bipartite minimum crossing matchings. In *Order and Geometry Workshop, Problem booklet.*, pages 5–6, 2016. URL: <http://orderandgeometry2016.tcs.uj.edu.pl/docs/OG2016-ProblemBooklet.pdf>.
- 40 Tillmann Miltzow, Lothar Narins, Yoshio Okamoto, Günter Rote, Antonis Thomas, and Takeaki Uno. Approximation and Hardness of Token Swapping. In *24th Annual European Symposium on Algorithms, ESA 2016*, pages 66:1–66:15, 2016.
- 41 Monroe M. Newborn and William O. J. Moser. Optimal crossing-free Hamiltonian circuit drawings of  $K_n$ . *J. Comb. Theory, Ser. B*, 29(1):13–26, 1980.
- 42 Michael Osigbemeh, Michael Onuu, and Olumuyiwa Asaolu. Design and development of an improved traffic light control system using hybrid lighting system. *Journal of Traffic and Transportation Engineering (English Edition)*, 4(1):88–95, 2017. Special Issue: Driver Behavior, Highway Capacity and Transportation Resilience.
- 43 Marcus Schaefer. The Graph Crossing Number and its Variants: A Survey. *The Electronic Journal of Combinatorics*, 20, April 2013.
- 44 Carl Sechen. *VLSI placement and global routing using simulated annealing*, volume 54. Springer Science & Business Media, 2012.
- 45 Micha Sharir and Emo Welzl. On the Number of Crossing-Free Matchings, Cycles, and Partitions. *SIAM J. Comput.*, 36(3):695–720, 2006.
- 46 Paul Turán. A note of welcome. *Journal of Graph Theory*, 1(1):7–9, 1997.
- 47 Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. *JoCG*, 8(1):47–77, 2017.
- 48 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping Labeled

- Tokens on Graphs. In Alfredo Ferro, Fabrizio Luccio, and Peter Widmayer, editors, *Fun with Algorithms*, pages 364–375, 2014.
- 49 Katsuhisa Yamanaka, Erik D. Demaine, Takehiro Ito, Jun Kawahara, Masashi Kiyomi, Yoshio Okamoto, Toshiki Saitoh, Akira Suzuki, Kei Uchizawa, and Takeaki Uno. Swapping labeled tokens on graphs. *Theoretical Computer Science*, 586:81–94, 2015. Fun with Algorithms.
- 50 Lanbo Zheng and Christoph Buchheim. A New Exact Algorithm for the Two-Sided Crossing Minimization Problem. In *Proceedings of the First International Conference on Combinatorial Optimization and Applications, COCOA*, volume 4616 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 2007.



# General Techniques for Approximate Incidences and Their Application to the Camera Posing Problem

**Dror Aiger**

Google, Tel Aviv, Israel  
<https://ai.google/research/people/DrorAiger>  
aigerd@google.com

**Haim Kaplan**

School of Computer Science, Tel Aviv University, Tel Aviv, Israel  
Google, Tel Aviv, Israel  
haimk@tau.ac.il

**Efi Kokiopoulou**

Google, Zurich, Switzerland  
efi@google.com

**Micha Sharir**

School of Computer Science, Tel Aviv University, Tel Aviv, Israel  
michas@tau.ac.il

**Bernhard Zeisl**

Google, Zurich, Switzerland  
bzeisl@google.com

---

## Abstract

We consider the classical camera pose estimation problem that arises in many computer vision applications, in which we are given  $n$  2D-3D correspondences between points in the scene and points in the camera image (some of which are incorrect associations), and where we aim to determine the camera pose (the position and orientation of the camera in the scene) from this data. We demonstrate that this posing problem can be reduced to the problem of computing  $\varepsilon$ -approximate incidences between two-dimensional surfaces (derived from the input correspondences) and points (on a grid) in a four-dimensional pose space. Similar reductions can be applied to other camera pose problems, as well as to similar problems in related application areas.

We describe and analyze three techniques for solving the resulting  $\varepsilon$ -approximate incidences problem in the context of our camera posing application. The first is a straightforward assignment of surfaces to the cells of a grid (of side-length  $\varepsilon$ ) that they intersect. The second is a variant of a primal-dual technique, recently introduced by a subset of the authors [3] for different (and simpler) applications. The third is a non-trivial generalization of a data structure Fonseca and Mount [4], originally designed for the case of hyperplanes. We present and analyze this technique in full generality, and then apply it to the camera posing problem at hand.

We compare our methods experimentally on real and synthetic data. Our experiments show that for the typical values of  $n$  and  $\varepsilon$ , the primal-dual method is the fastest, also in practice.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Camera positioning, Approximate incidences, Incidences

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.8

**Related Version** A full version of the paper is available at <http://arxiv.org/abs/1903.07047>.

**Funding** *Haim Kaplan*: Partially supported by ISF grant 1841/14, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.

*Micha Sharir*: Partially supported by ISF Grant 260/18, by grant 1367/2016 from the German-Israeli Science Foundation (GIF), and by Blavatnik Research Fund in Computer Science at Tel Aviv University.



© Dror Aiger, Haim Kaplan, Efi Kokiopoulou, Micha Sharir, and Bernhard Zeisl; licensed under Creative Commons License CC-BY

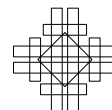
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 8; pp. 8:1–8:14

Leibniz International Proceedings in Informatics



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



## 1 Introduction

Camera pose estimation is a fundamental problem in computer vision, which aims at determining the pose and orientation of a camera solely from an image. This localization problem appears in many interesting real-world applications, such as for the navigation of self-driving cars [6], in incremental environment mapping such as Structure-from-Motion (SfM) [1, 10, 11], or for augmented reality [8, 9, 12], where a significant component are algorithms that aim to estimate an accurate camera pose in the world from image data.

Given a three-dimensional point-cloud model of a scene, the classical, but also state-of-the-art approach to absolute camera pose estimation consists of a two-step procedure. First, one matches a large number of features in the two-dimensional camera image with corresponding features in the three-dimensional scene. Then one uses these putative correspondences to determine the pose and orientation of the camera. Typically, the matches obtained in the first step contain many incorrect associations, forcing the second step to use filtering techniques to reject incorrect matches. Subsequently, the absolute 6 degrees-of-freedom (DoF) camera pose is estimated, for example, with a perspective  $n$ -point pose solver [7] within a RANSAC scheme [5].

In this work we concentrate on the second step of the camera pose problem. That is, we consider the task of estimating the camera pose and orientation from a (potentially large) set of  $n$  already calculated image-to-scene correspondences.

Further, we assume that we are given a common direction between the world and camera frames. For example, inertial sensors, available on any smart-phone nowadays, allow to estimate the vertical gravity direction in the three-dimensional camera coordinate system. This alignment of the vertical direction fixes two degrees of freedom for the rotation between the frames and we are left to estimate four degrees of freedom out of the general six. To obtain four equations (in the four remaining degrees of freedom), this setup requires two pairs of image-to-scene correspondences<sup>1</sup> for a minimal solver. Hence a corresponding naive RANSAC-based scheme requires  $O(n^2)$  filtering steps, where in each iterations a pose hypothesis based on a different pair of correspondences is computed and verified against all other correspondences.

Recently, Zeisl et al. [13] proposed a Hough-voting inspired outlier filtering and camera posing approach, which computes the camera pose up to an accuracy of  $\varepsilon > 0$  from a set of 2D-3D correspondences, in  $O(n/\varepsilon^2)$  time, under the same alignment assumptions of the vertical direction. In this paper we propose new algorithms that work considerably faster in practice, but under milder assumptions. Our method is based on a reduction of the problem to a problem of counting  $\varepsilon$ -approximate incidences between points and surfaces, where a point  $p$  is  $\varepsilon$ -approximately incident (or just  $\varepsilon$ -incident) to a surface  $\sigma$  if the (suitably defined) distance between  $p$  and  $\sigma$  is at most  $\varepsilon$ . This notion has recently been introduced by a subset of the authors in [3], and applied in a variety of instances, involving somewhat simpler scenarios than the one considered here. Our approach enables us to compute a camera pose when the number of correspondences  $n$  is large, and many of which are expected to be outliers. In contrast, a direct application of RANSAC-based methods on such inputs is very slow, since the fraction of inliers is small. In the limit, trying all pairs of matches involves  $\Omega(n^2)$  RANSAC iterations. Moreover, our methods enhance the quality of the posing considerably [13], since each generated candidate pose is close to (i.e., consistent with) with many of the correspondences.

---

<sup>1</sup> As we will see later in detail, each correspondence imposes two constraints on the camera pose.

**Our results.** We formalize the four degree-of-freedom camera pose problem as an approximate incidences problem in Section 2. Each 2D-3D correspondence is represented as a two-dimensional surface in the 4-dimensional pose-space, which is the locus of all possible positions and orientations of the camera that fit the correspondence exactly. Ideally, we would like to find a point (a pose) that lies on as many surfaces as possible, but since we expect the data to be noisy, and the exact problem is inefficient to solve anyway, we settle for an approximate version, in which we seek a point with a large number of approximate incidences with the surfaces.

Formally, we solve the following problem. We have an error parameter  $\varepsilon > 0$ , we lay down a grid on  $[0, 1]^d$  of side length  $\varepsilon$ , and compute, for each vertex  $v$  of the grid, a count  $I(v)$  of surfaces that are approximately incident to  $v$ , so that (i) every surface that is  $\varepsilon$ -incident to  $v$  is counted in  $I(v)$ , and (ii) every surface that is counted in  $I(v)$  is  $\alpha\varepsilon$ -incident to  $v$ , for some small constant  $\alpha > 1$  (but not all  $\alpha\varepsilon$ -incident surfaces are necessarily counted). We output the grid vertex  $v$  with the largest count  $I(v)$  (or a list of vertices with the highest counts, if so desired).

As we will comment later, (a) restricting the algorithm to grid vertices only does not miss a good pose  $v$ : a vertex of the grid cell containing  $v$  serves as a good substitute for  $v$ , and (b) we have no real control on the value of  $I(v)$ , which might be much larger than the number of surfaces that are  $\varepsilon$ -incident to  $v$ , but all the surfaces that we count are “good” – they are reasonably close to  $v$ . In the computer vision application, and in many related applications, neither of these issues is significant.

We give three algorithms for this camera-pose approximate-incidences problem. The first algorithm simply computes the grid cells that each surface intersects, and considers the number of intersecting surfaces per cell as its approximate  $\varepsilon$ -incidences count. This method takes time  $O\left(\frac{n}{\varepsilon^2}\right)$  for all vertices of our  $\varepsilon$ -size grid. We then describe a faster algorithm using geometric duality, in Section 3. It uses a coarser grid in the primal space and switches to a dual 5-dimensional space (a 5-tuple is needed to specify a 2D-3D correspondence and its surface, now dualized to a point). In the dual space each query (i.e., a vertex of the grid) becomes a 3-dimensional surface, and each original 2-dimensional surface in the primal 4-dimensional space becomes a point. This algorithm takes  $O\left(\frac{n^{3/5}}{\varepsilon^{14/5}} + n + \frac{1}{\varepsilon^4}\right)$  time, and is asymptotically faster than the simple algorithm for  $n > 1/\varepsilon^2$ .

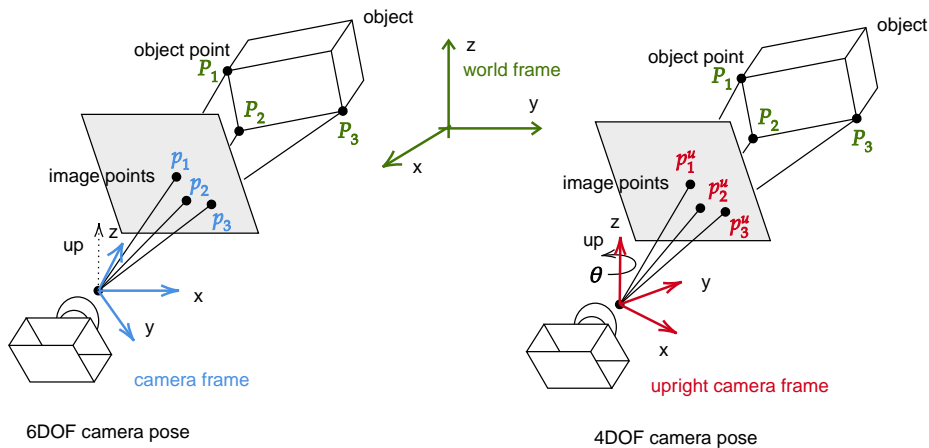
Finally, we give a general method for constructing an approximate incidences data structure for general  $k$ -dimensional algebraic surfaces (that satisfy certain mild conditions) in  $\mathbb{R}^d$ , in Section 4. It extends the technique of Fonseca and Mount [4], designed for the case of hyperplanes, and takes  $O(n + \text{poly}(1/\varepsilon))$  time, where the degree of the polynomial in  $1/\varepsilon$  depends on the number of parameters needed to specify a surface, the dimension of the surfaces, and the dimension of the ambient space. We first present and analyze this technique in full generality, and then apply it to the surfaces obtained for our camera posing problem. In this case, the data structure requires  $O(n + 1/\varepsilon^6)$  storage and is constructed in roughly the same time. This is asymptotically faster than our primal-dual scheme when  $n \geq 1/\varepsilon^{16/3}$  (for  $n \geq 1/\varepsilon^7$  the  $O(n)$  term dominates and these two methods are asymptotically the same). Due to its generality, the latter technique is easily adapted to other surfaces and thus is of general interest and potential. In contrast, the primal-dual method requires nontrivial adaptation as it switches from one approximate-incidences problem to another and the dual space and its distance function depend on the type of the input surfaces.

We implemented our algorithms and compared their performance on real and synthetic data. Our experimentation shows that, for commonly used values of  $n$  and  $\varepsilon$  in practical scenarios ( $n \in [8K, 32K]$ ,  $\varepsilon \in [0.02, 0.03]$ ), the primal-dual scheme is considerably faster than

the other algorithms, and should thus be the method of choice. Due to lack of space, the experimentation details are omitted in this version, with the exception of a few highlights. They can be found in the full version of the paper [2].

## 2 From camera positioning to approximate incidences

Suppose we are given a pre-computed three-dimensional scene and a two-dimensional picture of it. Our goal is to deduce from this image the location and orientation of the camera in the scene. In general, the camera, as a rigid body in 3-space, has six degrees of freedom, three of translation and three of rotation (commonly referred to as the *yaw*, *pitch* and *roll*). We simplify the problem by making the realistic assumption, that the vertical direction of the scene is known in the camera coordinate frame (e.g., estimated by an inertial sensor on smart phones). This allows us to rotate the camera coordinate frame such that its  $z$ -axis is parallel to the world  $z$ -axis, thereby fixing the pitch and roll of the camera and leaving only four degrees of freedom  $(x, y, z, \theta)$ , where  $c = (x, y, z)$  is the location of the camera center, say, and  $\theta$  is its yaw, i.e. horizontal the orientation of the optical axis around the vertical direction. See Figure 1.



■ **Figure 1** With the knowledge of a common vertical direction between the camera and world frame the general 6DoF camera posing problem reduces to estimating 4 parameters. This is the setup we consider in our work.

By preprocessing the scene, we record the spatial coordinates  $w = (w_1, w_2, w_3)$  of a discrete (large) set of salient points. We assume that some (ideally a large number) of the distinguished points are identified in the camera image, resulting in a set of image-to-scene correspondences. Each correspondence  $\mathbf{w} = \{w_1, w_2, w_3, \xi, \eta\}$  is parameterized by five parameters, the spatial position  $w$  and the position  $v = (\xi, \eta)$  in the camera plane of view of the same salient point. Our goal is to find a camera pose  $(x, y, z, \theta)$  so that as many correspondences as possible are (approximately) *consistent* with it, i.e., the ray from the camera center  $c$  to  $w$  goes approximately through  $(\xi, \eta)$  in the image plane, when the yaw of the camera is  $\theta$ .



### 2.1 Camera posing as an $\varepsilon$ -incidences problem

Each correspondence and its 5-tuple  $\mathbf{w}$  define a two-dimensional surface  $\sigma_{\mathbf{w}}$  in parametric 4-space, which is the locus of all poses  $(x, y, z, \theta)$  of the camera at which it sees  $w$  at coordinates  $(\xi, \eta)$  in its image. For  $n$  correspondences, we have a set of  $n$  such surfaces. We prove that each point in the parametric 4-space of camera poses that is close to a surface  $\sigma_{\mathbf{w}}$ , in a suitable metric defined in that 4-space, represents a camera pose where  $w$  is projected to a point in the camera viewing plane that is close to  $(\xi, \eta)$ , and vice versa (see Section 2.2 for the actual expressions for these projections). Therefore, a point in 4-space that is close to a large number of surfaces represents a camera pose with many approximately consistent correspondences, which is a strong indication of being close to the correct pose.

Extending the notation used in the earlier work [3], we say that a point  $q$  is  $\varepsilon$ -incident to a surface  $\sigma$  if  $\text{dist}(q, \sigma) \leq \varepsilon$ . Our algorithms approximate, for each vertex of a grid  $G^\varepsilon$  of side length  $\varepsilon$ , the number of  $\varepsilon$ -incident surfaces and suggest the vertex with the largest count as the best candidate for the camera pose. This work extends the approximate incidences methodology in [3] to the (considerably more involved) case at hand.

### 2.2 The surfaces $\sigma_{\mathbf{w}}$

Let  $w = (w_1, w_2, w_3)$  be a salient point in  $\mathbb{R}^3$ , and assume that the camera is positioned at  $(c, \theta) = (x, y, z, \theta)$ . We represent the orientation of the vector  $w - c$ , within the world frame, by its spherical coordinates  $(\varphi, \psi)$ , except that, unlike the standard convention, we take  $\psi$  to be the angle with the  $xy$ -plane (rather than with the  $z$ -axis):

$$\tan \psi = \frac{w_3 - z}{\sqrt{(w_1 - x)^2 + (w_2 - y)^2}} \quad \tan \varphi = \frac{w_2 - y}{w_1 - x}$$

In the two-dimensional frame of the camera the  $(\xi, \eta)$ -coordinates model the *view* of  $w$ , which differs from above polar representation of the vector  $w - c$  only by the polar orientation  $\theta$  of the viewing plane itself. Writing  $\kappa$  for  $\tan \theta$ , we have

$$\xi = \tan(\varphi - \theta) = \frac{\tan \varphi - \tan \theta}{1 + \tan \varphi \tan \theta} = \frac{(w_2 - y) - \kappa(w_1 - x)}{(w_1 - x) + \kappa(w_2 - y)}, \tag{1}$$

$$\eta = \tan \psi = \frac{w_3 - z}{\sqrt{(w_1 - x)^2 + (w_2 - y)^2}}.$$

We note that using  $\tan \theta$  does not distinguish between  $\theta$  and  $\theta + \pi$ , but we will restrict  $\theta$  to lie in  $[-\pi/4, \pi/4]$  or in similar narrower ranges, thereby resolving this issue.

We use  $\mathbb{R}^4$  with coordinates  $(x, y, z, \kappa)$  as our primal space, where each point models a possible pose of the camera. Each correspondence  $\mathbf{w}$  is parameterized by the triple  $(w, \xi, \eta)$ , and defines a two-dimensional algebraic surface  $\sigma_{\mathbf{w}}$  of degree at most 4, whose equations (in  $x, y, z, \kappa$ ) are given in (1). It is the locus of all camera poses  $v = (x, y, z, \kappa)$  at which it sees  $w$  at image coordinates  $(\xi, \eta)$ . We can rewrite these equations into the following parametric representation of  $\sigma_{\mathbf{w}}$ , expressing  $z$  and  $\kappa$  as functions of  $x$  and  $y$ :

$$\kappa = \frac{(w_2 - y) - \xi(w_1 - x)}{(w_1 - x) + \xi(w_2 - y)} \quad z = w_3 - \eta \sqrt{(w_1 - x)^2 + (w_2 - y)^2}. \tag{2}$$

For a camera pose  $v = (x, y, z, \kappa)$ , and a point  $w = (w_1, w_2, w_3)$ , we write

$$F(v; w) = \frac{(w_2 - y) - \kappa(w_1 - x)}{(w_1 - x) + \kappa(w_2 - y)} \quad G(v; w) = \frac{w_3 - z}{\sqrt{(w_1 - x)^2 + (w_2 - y)^2}}. \tag{3}$$

In this notation we can write the Equations (1) characterizing  $\sigma_{\mathbf{w}}$  (when regarded as equations in  $v$ ) as  $\xi = F(v; w)$  and  $\eta = G(v; w)$ .

### 2.3 Measuring proximity

Given a guessed pose  $v = (x, y, z, \kappa)$  of the camera, we want to measure how well it fits the scene that the camera sees. For this, given a correspondence  $\mathbf{w} = (w, \xi, \eta)$ , we define the *frame distance*  $\text{fd}$  between  $v$  and  $\mathbf{w}$  as the  $L_\infty$ -distance between  $(\xi, \eta)$  and  $(\xi_v, \eta_v)$ , where, as in Eq. (3),  $\xi_v = F(v; w)$ ,  $\eta_v = G(v; w)$ . That is,

$$\text{fd}(v, \mathbf{w}) = \max \{ |\xi_v - \xi|, |\eta_v - \eta| \}. \quad (4)$$

Note that  $(\xi_v, \eta_v)$  are the coordinates at which the camera would see  $w$  if it were placed at position  $v$ , so the frame distance is the  $L_\infty$ -distance between these coordinates and the actual coordinates  $(\xi, \eta)$  at which the camera sees  $w$ ; this serves as a natural measure of how close  $v$  is to the actual pose of the camera.

We are given a viewed scene of  $n$  distinguished points (correspondences)  $\mathbf{w} = (w, \xi, \eta)$ . Let  $S$  denote the set of  $n$  surfaces  $\sigma_{\mathbf{w}}$ , representing these correspondences. We assume that the salient features  $w$  and the camera are all located within some bounded region, say  $[0, 1]^3$ . The replacement of  $\theta$  by  $\kappa = \tan \theta$  makes its range unbounded, so we break the problem into four subproblems, in each of which  $\theta$  is confined to some sector. In the first subproblem we assume that  $-\pi/4 \leq \theta \leq \pi/4$ , so  $-1 \leq \kappa \leq 1$ . The other three subproblems involve the ranges  $[\pi/4, 3\pi/4]$ ,  $[3\pi/4, 5\pi/4]$ , and  $[5\pi/4, 7\pi/4]$ . We only consider here the first subproblem; the treatment of the others is fully analogous. In each such range, replacing  $\theta$  by  $\tan \theta$  does not incur the ambiguity of identifying  $\theta$  with  $\theta + \pi$ .

Given an error parameter  $\varepsilon > 0$ , we seek an approximate pose  $v$  of the camera, at which many correspondences  $\mathbf{w}$  are within frame distance at most  $\varepsilon$  from  $v$ , as given in (4).

The following two lemmas relate our frame distance to the Euclidean distance. Their (rather technical) proofs are given in the full version of this paper [2].

► **Lemma 1.** *Let  $v = (x, y, z, \kappa)$ , and let  $\sigma_{\mathbf{w}}$  be the surface associated with a correspondence  $\mathbf{w} = \{w_1, w_2, w_3, \xi, \eta\}$ . Let  $v'$  be a point on  $\sigma_{\mathbf{w}}$  such that  $|v - v'| \leq \varepsilon$  (where  $|\cdot|$  denotes the Euclidean norm). If*

- (i)  $|(w_1 - x) + \kappa(w_2 - y)| \geq a > 0$ , and
- (ii)  $(w_1 - x)^2 + (w_2 - y)^2 \geq a > 0$ , for some absolute constant  $a$ ,

*then  $\text{fd}(v, \mathbf{w}) \leq \beta\varepsilon$  for some constant  $\beta$  that depends on  $a$ .*

Informally, Condition (i) requires that the absolute value of the  $\xi = \tan(\varphi - \theta)$  coordinate of the position of  $w$  in the viewing plane, with the camera positioned at  $v$ , is not too large (i.e., that  $|(\varphi - \theta)|$  is not too close to  $\pi/2$ ). We can ensure this property by restricting the camera image to some suitably bounded  $\xi$ -range.

Similarly, Condition (ii) requires that the  $xy$ -projection of the vector  $w - c$  is not too small. It can be violated in two scenarios. Either we look at a data point that is too close to  $c$ , or we see it looking too much “upwards” or “downwards”. We can ensure that the latter situation does not arise, by restricting the camera image, as in the preceding paragraph, to some suitably bounded  $\eta$ -range too. That done, we ensure that the former situation does not arise by requiring that the physical distance between  $c$  and  $w$  be at least some multiple of  $a$ .

The next lemma establishes the converse connection.

► **Lemma 2.** *Let  $v = (x, y, z, \kappa)$  be a camera pose and  $\mathbf{w} = \{w_1, w_2, w_3, \xi, \eta\}$  a correspondence, such that  $\text{fd}(v, \mathbf{w}) \leq \varepsilon$ . Assume that  $|(w_1 - x) + \xi(w_2 - y)| \geq a > 0$ , for some absolute constant  $a$ , and consider the point  $v' = (x, y, z', \kappa') \in \sigma_{\mathbf{w}}$  where (see Eq. (2))*

$$z' = w_3 - \eta\sqrt{(w_1 - x)^2 + (w_2 - y)^2} \quad \kappa' = \frac{(w_2 - y) - \xi(w_1 - x)}{(w_1 - x) + \xi(w_2 - y)}.$$

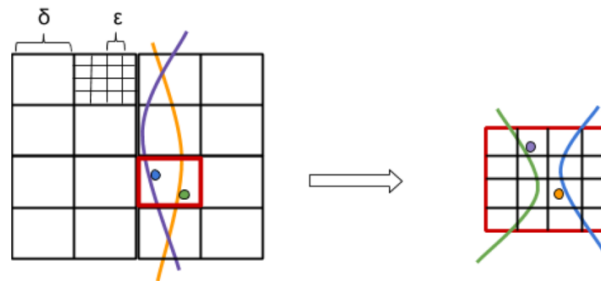
Then  $|z - z'| \leq \sqrt{2}\varepsilon$  and  $|\kappa - \kappa'| \leq c\varepsilon$ , for some constant  $c$ , again depending on  $a$ .

Informally, the condition  $|(w_1 - x) + \xi(w_2 - y)| \geq a > 0$  means that the orientation of the camera, when it is positioned at  $(x, y)$  and sees  $w$  at coordinate  $\xi$  of the viewing plane is not too close to  $\pm\pi/2$ . This is a somewhat artificial constraint that is satisfied by our restriction on the allowed yaw of the camera (the range of  $\kappa$ ).

**A Simple algorithm.** Using Lemma 2 and Lemma 1 we can derive a simple naive solution which does not require any of the sophisticated machinery developed in this work. We construct a grid  $G$  over  $Q = [0, 1]^3 \times [-1, 1]$ , of cells  $\tau$ , each of dimensions  $\varepsilon \times \varepsilon \times 2\sqrt{2}\varepsilon \times 2a\varepsilon$ , where  $a$  is the constant of Lemma 2. We use this non-square grid  $G$  since we want to find  $\varepsilon$ -approximate incidences in terms of frame distance. For each cell  $\tau$  of  $G$  we compute the number of surfaces  $\sigma_w$  that intersect  $\tau$ . This gives an approximate incidences count for the center of  $\tau$ . Further details and a precise statement can be found in the full version [2].

### 3 Primal-dual algorithm for geometric proximity

Following the general approach in [3], we use a suitable duality, with some care. We write  $\varepsilon = 2\gamma\delta_1\delta_2$ , for suitable parameters  $\gamma$ , and  $\varepsilon/(2\gamma) \leq \delta_1, \delta_2 \leq 1$ , whose concrete values are fixed later, and apply the decomposition scheme developed in [3] tailored to the case at hand. Specifically, we consider the coarser grid  $G_{\delta_1}$  in the primal space, of cell dimensions  $\delta_1 \times \delta_1 \times \sqrt{2}\delta_1 \times c\delta_1$ , where  $c$  is the constant from Lemma 2, that tiles up the domain  $Q = [0, 1]^3 \times [-1, 1]$  of possible camera positions. For each cell  $\tau$  of  $G_{\delta_1}$ , let  $S_\tau$  denote the set of surfaces that cross either  $\tau$  or one of the eight cells adjacent to  $\tau$  in the  $(z, \kappa)$ -directions.<sup>2</sup> The duality is illustrated in Figure 2.



■ **Figure 2** A schematic illustration of our duality-based algorithm.

We discretize the set of all possible positions of the camera by the vertices of the finer grid  $G_\varepsilon$ , defined as  $G_{\delta_1}$ , with  $\varepsilon$  replacing  $\delta_1$ , that tiles up  $Q$ . The number of these candidate positions is  $m := O(1/\varepsilon^4)$ . For each vertex  $q \in G_\varepsilon$ , we want to approximate the number of surfaces that are  $\varepsilon$ -incident to  $q$ , and output the vertex with the largest count as the best candidate for the position of the camera. Let  $V_\tau$  be the subset of  $G_\varepsilon$  contained in  $\tau$ . We ensure that the boxes of  $G_{\delta_1}$  are pairwise disjoint by making them half open, in the sense that if  $(x_0, y_0, z_0, \kappa_0)$  is the vertex of a box that has the smallest coordinates, then the box is defined by  $x_0 \leq x < x_0 + \delta_1, y_0 \leq y < y_0 + \delta_1, z_0 \leq z < z_0 + \sqrt{2}\delta_1, \kappa_0 \leq \kappa < \kappa_0 + c\delta_1$ . This makes the sets  $V_\tau$  pairwise disjoint as well. Put  $m_\tau = |V_\tau|$  and  $n_\tau = |S_\tau|$ . We have

<sup>2</sup> The choice of  $z, \kappa$  is arbitrary, but it is natural for the analysis, given in the full version [2].

$m_\tau = O((\delta_1/\varepsilon)^4)$  for each  $\tau$ . Since the surfaces  $\sigma_{\mathbf{w}}$  are two-dimensional algebraic surfaces of constant degree, each of them crosses  $O(1/\delta_1^2)$  cells of  $G_{\delta_1}$ , so we have  $\sum_\tau n_\tau = O(n/\delta_1^2)$ .

We now pass to the dual five-dimensional space. Each point in that space represents a correspondence  $\mathbf{w} = (w_1, w_2, w_3, \xi, \eta)$ . We use the first three components  $(w_1, w_2, w_3)$  as the first three coordinates, but modify the  $\xi$ - and  $\eta$ -coordinates in a manner that depends on the primal cell  $\tau$ . Let  $c_\tau = (x_\tau, y_\tau, z_\tau, \kappa_\tau)$  be the midpoint of the primal box  $\tau$ . For each  $\sigma_{\mathbf{w}} \in S_\tau$  we map  $\mathbf{w} = (w, \xi, \eta)$ , where  $w = (w_1, w_2, w_3)$ , to the point  $\mathbf{w}_\tau = (w_1, w_2, w_3, \xi_\tau, \eta_\tau)$ , where  $\xi_\tau = \xi - F(c_\tau; w)$  and  $\eta_\tau = \eta - G(c_\tau; w)$ , with  $F$  and  $G$  as given in (3). We have

► **Corollary 3.** *If  $\sigma_{\mathbf{w}}$  crosses  $\tau$  then  $|\xi_\tau|, |\eta_\tau| \leq \gamma\delta_1$ , for some absolute constant  $\gamma$ , provided that the following two properties hold, for some absolute constant  $a > 0$  (the constant  $\gamma$  depends on  $a$ ).*

- (i)  $|(w_1 - x_\tau) + \kappa_\tau(w_2 - y_\tau)| \geq a$ , and
- (ii)  $(w_1 - x_\tau)^2 + (w_2 - y_\tau)^2 \geq a$ , where  $(x_\tau, y_\tau)$  are the  $(x, y)$ -coordinates of the center of  $\tau$ .

**Proof.** If  $\sigma_{\mathbf{w}} \in S_\tau$  then it contains a point  $v'$  such that  $|v' - c_\tau| \leq c'\delta_1$ , for a suitable absolute constant  $c'$  (that depends on  $c$ ). We now apply Lemma 1, recalling (4). ◀

We take the  $\gamma$  provided by Corollary 3 as the  $\gamma$  in the definition of  $\delta_1$  and  $\delta_2$ . We map each point  $v \in V_\tau$  to the dual surface  $\sigma_v^* = \sigma_{v;\tau}^* = \{\mathbf{w}_\tau \mid v \in \sigma_{\mathbf{w}}\}$ . Using (3), we have

$$\sigma_{v;\tau}^* = \{(w, F(v; w) - F(c_\tau; w), G(v; w) - G(c_\tau; w)) \mid w = (w_1, w_2, w_3) \in [0, 1]^3\}.$$

By Corollary 3, the points  $\mathbf{w}_\tau$ , for the surfaces  $\sigma_{\mathbf{w}}$  that cross  $\tau$ , lie in the region  $R_\tau = [0, 1]^3 \times [-\gamma\delta_1, \gamma\delta_1]^2$ . We partition  $R_\tau$  into a grid  $G_{\delta_2}$  of  $1/\delta_2^5$  small congruent boxes, each of dimensions  $\delta_2 \times \delta_2 \times \delta_2 \times (2\gamma\delta_1\delta_2) \times (2\gamma\delta_1\delta_2) = \delta_2 \times \delta_2 \times \delta_2 \times \varepsilon \times \varepsilon$ .

Exactly as in the primal setup, we make each of these boxes half-open, thereby making the sets of dual vertices in the smaller boxes pairwise disjoint. We assign to each of these dual cells  $\tau^*$  the set  $S_{\tau^*}^*$  of dual points that lie in  $\tau^*$ , and the set  $V_{\tau^*}^*$  of the dual surfaces that cross either  $\tau^*$  or one of the eight cells adjacent to  $\tau^*$  in the  $(\xi_\tau, \eta_\tau)$ -directions. Put  $n_{\tau^*} = |S_{\tau^*}^*|$  and  $m_{\tau^*} = |V_{\tau^*}^*|$ . Since the dual cells are pairwise disjoint, we have  $\sum_{\tau^*} n_{\tau^*} = n_\tau$ . Since the dual surfaces are three-dimensional algebraic surfaces of constant degree, each of them crosses  $O(1/\delta_2^3)$  grid cells, so  $\sum_{\tau^*} m_{\tau^*} = O(m_\tau/\delta_2^3)$ .

We compute, for each dual surface  $\sigma_v^*$ , the sum  $\sum_{\tau^*} |S_{\tau^*}^*|$ , over the dual cells  $\tau^*$  that are either crossed by  $\sigma_v^*$  or that one of their adjacent cells in the  $(\xi_\tau, \eta_\tau)$ -directions is crossed by  $\sigma_v^*$ . We output the vertex  $v$  of  $G_\varepsilon$  with the largest resulting count, over all primal cells  $\tau$ .

The following theorem establishes the correctness of our technique. Its proof is given in Appendix B of the full version [2].

► **Theorem 4.** *Suppose that for every cell  $\tau \in G_{\delta_1}$  and for every point  $v = (x, y, z, \kappa) \in V_\tau$  and every  $\mathbf{w} = ((w_1, w_2, w_3), \xi, \eta)$  such that  $\sigma_{\mathbf{w}}$  intersects either  $\tau$  or one of its adjacent cells in the  $(\xi_\tau, \eta_\tau)$ -directions, we have that, for some absolute constant  $a > 0$ ,*

- (i)  $|(w_1 - x) + \kappa(w_2 - y)| \geq a$ ,
- (ii)  $(w_1 - x)^2 + (w_2 - y)^2 \geq a$ , and
- (iii)  $|(w_1 - x) + \xi(w_2 - y)| \geq a$ .

*Then (a) For each  $v \in V$ , every pair  $(v, \mathbf{w})$  at frame distance  $\leq \varepsilon$  is counted (as an  $\varepsilon$ -incidence of  $v$ ) by the algorithm. (b) For each  $v \in V$ , every pair  $(v, \mathbf{w})$  that we count lies at frame distance  $\leq \alpha\varepsilon$ , for some constant  $\alpha > 0$  depending on  $a$ .*

### 3.1 Running time analysis

The cost of the algorithm is clearly proportional to  $\sum_{\tau} \sum_{\tau^*} (m_{\tau^*} + n_{\tau^*})$ , over all primal cells  $\tau$  and the dual cells  $\tau^*$  associated with each cell  $\tau$ . We have

$$\sum_{\tau} \sum_{\tau^*} (m_{\tau^*} + n_{\tau^*}) = O\left(\sum_{\tau} (m_{\tau}/\delta_2^3 + n_{\tau})\right) = O(m/\delta_2^3 + n/\delta_1^2).$$

Optimizing the choice of  $\delta_1$  and  $\delta_2$ , we choose  $\delta_1 = \left(\frac{\varepsilon^3 n}{m}\right)^{1/5}$  and  $\delta_2 = \left(\frac{\varepsilon^2 m}{n}\right)^{1/5}$ . These choices make sense as long as each of  $\delta_1, \delta_2$  lies between  $\varepsilon/(2\gamma)$  and 1. That is,  $\frac{\varepsilon}{2\gamma} \leq \left(\frac{\varepsilon^3 n}{m}\right)^{1/5} \leq 1$  and  $\frac{\varepsilon}{2\gamma} \leq \left(\frac{\varepsilon^2 m}{n}\right)^{1/5} \leq 1$ , or  $c'\varepsilon^2 m \leq n \leq \frac{c''m}{\varepsilon^3}$ , where  $c'$  and  $c''$  are absolute constants (that depend on  $\gamma$ ).

If  $n < c'\varepsilon^2 m$ , we use only the primal setup, taking  $\delta_1 = \varepsilon$  (for the primal subdivision). The cost is then  $O(n/\varepsilon^2 + m) = O(m)$ . Similarly, if  $n > \frac{c''m}{\varepsilon^3}$ , we use only the dual setup, taking  $\delta_1 = 1$  and  $\delta_2 = \varepsilon/(2\gamma)$ , and the cost is thus  $O(n + m/\varepsilon^3) = O(n)$ . Adding everything together, to cover all three subranges, the running time is then  $O\left(\frac{m^{2/5}n^{3/5}}{\varepsilon^{6/5}} + n + m\right)$ . Substituting  $m = O(1/\varepsilon^4)$ , we get a running time of  $O\left(\frac{n^{3/5}}{\varepsilon^{14/5}} + n + \frac{1}{\varepsilon^4}\right)$ . The first term dominates when  $n = \Omega\left(\frac{1}{\varepsilon^2}\right)$  and  $n = O\left(\frac{1}{\varepsilon^7}\right)$ . In conclusion, we have the following result.

► **Theorem 5.** *Given  $n$  data points that are seen (and identified) in a two-dimensional image taken by a vertically positioned camera, and an error parameter  $\varepsilon > 0$ , where the viewed points satisfy the assumptions made in Theorem 4, we can compute, in  $O\left(\frac{n^{3/5}}{\varepsilon^{14/5}} + n + \frac{1}{\varepsilon^4}\right)$  time, a vertex  $v$  of  $G_{\varepsilon}$  that maximizes the approximate count of  $\varepsilon$ -incident correspondences, where “approximate” means that every correspondence  $\mathbf{w}$  whose surface  $\sigma_{\mathbf{w}}$  is at frame distance at most  $\varepsilon$  from  $v$  is counted and every correspondence that we count lies at frame distance at most  $\alpha\varepsilon$  from  $v$ , for some fixed constant  $\alpha$ .*

Restricting ourselves only to grid vertices does not really miss any solution. We only lose a bit in the quality of approximation, replacing  $\varepsilon$  by a slightly large constant multiple thereof, when we move from the best solution to a vertex of its grid cell.

## 4 Geometric proximity via canonical surfaces

In this section we present a general technique to preprocess a set of algebraic surfaces into a data structure that can answer approximate incidences queries. In this technique we round the  $n$  original surfaces into a set of canonical surfaces, whose size depends only on  $\varepsilon$ , such that each original surface has a canonical surface that is “close” to it. Then we build an octree-based data structure for approximate incidences queries with respect to the canonical surfaces. However, to reduce the number of intersections between the cells of the octree and the surfaces, we further reduce the number of surfaces as we go from one level of the octree to the next, by rounding them in a coarser manner into a smaller set of surfaces.

This technique has been introduced by Fonseca and Mount [4] for the case of hyperplanes. We describe as a warmup step, in Appendix C of the full version [2], our interpretation of their technique applied to hyperplanes. We then extend here the technique to general surfaces, and apply it to the specific instance of 2-surfaces in 4-space that arise in the camera pose problem.

## 8:10 Approximation Algorithms for Camera Posing

We have a set  $S$  of  $n$   $k$ -dimensional surfaces in  $\mathbb{R}^d$  that cross the unit cube  $[0, 1]^d$ , and a given error parameter  $\varepsilon$ . We assume that each surface  $\sigma \in S$  is given in parametric form, where the first  $k$  coordinates are the parameters, so its equations are

$$x_j = F_j^{(\sigma)}(x_1, \dots, x_k), \quad \text{for } j = k + 1, \dots, d.$$

Moreover, we assume that each  $\sigma \in S$  is defined in terms of  $\ell$  *essential parameters*  $\mathbf{t} = (t_1, \dots, t_\ell)$ , and  $d-k$  additional *free additive parameters*  $\mathbf{f} = (f_{k+1}, \dots, f_d)$ , one free parameter for each dependent coordinate. Concretely, we assume that the equations defining the surface  $\sigma \in S$ , parameterized by  $\mathbf{t}$  and  $\mathbf{f}$  (we then denote  $\sigma$  as  $\sigma_{\mathbf{t}, \mathbf{f}}$ ), are

$$x_j = F_j(\mathbf{x}; \mathbf{t}) + f_j = F_j(x_1, \dots, x_k; t_1, \dots, t_\ell) + f_j, \quad \text{for } j = k + 1, \dots, d.$$

For each equation of the surface that does not have a free parameter in the original expression, we introduce an *artificial* free parameter, and initialize its value to 0. (We need this separation into essential and free parameters for technical reasons that will become clear later.) We assume that  $\mathbf{t}$  (resp.,  $\mathbf{f}$ ) varies over  $[0, 1]^\ell$  (resp.,  $[0, 1]^{d-k}$ ).

**Remark.** The distinction between free and essential parameters seems to be artificial, but yet free parameters do arise in certain basic cases, such as the case of hyperplanes discussed in Appendix C of [2]. In the case of our 2-surfaces in 4-space, the parameter  $w_3$  is free, and we introduce a second artificial free parameter into the equation for  $\kappa$ . The number of essential parameters is  $\ell = 4$  (they are  $w_1, w_2, \xi$ , and  $\eta$ ).

We assume that the functions  $F_j$  are all continuous and differentiable, in all of their dependent variables  $\mathbf{x}$ ,  $\mathbf{t}$  and  $\mathbf{f}$  (this is a trivial assumption for  $\mathbf{f}$ ), and that they satisfy the following two conditions.

- (i) **Bounded gradients.**  $|\nabla_{\mathbf{x}} F_j(\mathbf{x}; \mathbf{t})| \leq c_1$ ,  $|\nabla_{\mathbf{t}} F_j(\mathbf{x}; \mathbf{t})| \leq c_1$ , for each  $j = k + 1, \dots, d$ , for any  $\mathbf{x} \in [0, 1]^k$  and any  $\mathbf{t} \in [0, 1]^\ell$ , where  $c_1$  is some absolute constant. Here  $\nabla_{\mathbf{x}}$  (resp.,  $\nabla_{\mathbf{t}}$ ) means the gradient with respect to only the variables  $\mathbf{x}$  (resp.,  $\mathbf{t}$ ).
- (ii) **Lipschitz gradients.**  $|\nabla_{\mathbf{x}} F_j(\mathbf{x}; \mathbf{t}) - \nabla_{\mathbf{x}} F_j(\mathbf{x}; \mathbf{t}')| \leq c_2 |\mathbf{t} - \mathbf{t}'|$ , for each  $j = k + 1, \dots, d$ , for any  $\mathbf{x} \in [0, 1]^k$  and any  $\mathbf{t}, \mathbf{t}' \in [0, 1]^\ell$ , where  $c_2$  is some absolute constant. This assumption is implied by the assumption that all the eigenvalues of the mixed part of the Hessian matrix  $\nabla_{\mathbf{t}} \nabla_{\mathbf{x}} F_j(\mathbf{x}; \mathbf{t})$  have absolute value bounded by  $c_2$ .

### 4.1 Canonizing the input surfaces

We first replace each surface  $\sigma_{\mathbf{t}, \mathbf{f}} \in S$  by a canonical “nearby” surface  $\sigma_{\mathbf{s}, \mathbf{g}}$ . Let  $\varepsilon' = \frac{\varepsilon}{c_2 \log(1/\varepsilon)}$  where  $c_2$  is the constant from Condition (ii). We get  $\mathbf{s}$  from  $\mathbf{t}$  (resp.,  $\mathbf{g}$  from  $\mathbf{f}$ ) by rounding each coordinate in the essential parametric domain  $L$  (resp., in the parametric domain  $\Phi$ ) to a multiple of  $\varepsilon'/(\ell + 1)$ . Note that each of the artificial free parameters (those that did not exist in the original equations) has the initial value 0 for all surfaces, and remains 0 in the rounded surfaces. We get  $O\left((1/\varepsilon')^{\ell'}\right)$  *canonical* rounded surfaces, where  $\ell' \geq \ell$  is the number of *original* parameters, that is, the number of essential parameters plus the number of non-artificial free parameters; in the worst case we have  $\ell' = \ell + d - k$ .

For a surface  $\sigma_{\mathbf{t}, \mathbf{f}}$  and its rounded version  $\sigma_{\mathbf{s}, \mathbf{g}}$  we have, for each  $j$ ,

$$\begin{aligned} |(F_j(\mathbf{x}; \mathbf{t}) + f_j) - (F_j(\mathbf{x}; \mathbf{s}) + g_j)| &\leq |\nabla_{\mathbf{t}} F_j(\mathbf{x}; \mathbf{t}')| \cdot |\mathbf{t} - \mathbf{s}| + |f_j - g_j| \\ &\leq c_1 |\mathbf{t} - \mathbf{s}| + |f_j - g_j| \leq (c_1 + 1) \varepsilon', \end{aligned}$$

where  $\mathbf{t}'$  is some intermediate value, which is irrelevant due to Condition (i).

We will use the  $\ell_2$ -norm of the difference vector  $((F_j(\mathbf{x}; \mathbf{t}) + f_j) - (F_j(\mathbf{x}; \mathbf{s}) + g_j))_{j=k+1}^d$  as the measure of proximity between the surfaces  $\sigma_{\mathbf{t}, \mathbf{f}}$  and  $\sigma_{\mathbf{s}, \mathbf{g}}$  at  $\mathbf{x}$ , and denote it as  $\text{dist}(\sigma_{\mathbf{t}, \mathbf{f}}, \sigma_{\mathbf{s}, \mathbf{g}}; \mathbf{x})$ . The maximum  $\text{dist}(\sigma_{\mathbf{t}, \mathbf{f}}, \sigma_{\mathbf{s}, \mathbf{g}}) := \max_{\mathbf{x} \in [0, 1]^k} \text{dist}(\sigma_{\mathbf{t}, \mathbf{f}}, \sigma_{\mathbf{s}, \mathbf{g}}; \mathbf{x})$  measures the global proximity of the two surfaces. (Note that it is an upper bound on the Hausdorff distance between the two surfaces.) We thus have  $\text{dist}(\sigma_{\mathbf{t}, \mathbf{f}}, \sigma_{\mathbf{s}, \mathbf{g}}) \leq (c_1 + 1)\varepsilon'$  when  $\sigma_{\mathbf{s}, \mathbf{g}}$  is the canonical surface approximating  $\sigma_{\mathbf{t}, \mathbf{f}}$ .

We define the *weight* of each canonical surface to be the number of original surfaces that got rounded to it, and we refer to the set of all canonical surfaces by  $S^c$ .

### 4.2 Approximately counting $\varepsilon$ -incidences

We describe an algorithm for approximating the  $\varepsilon$ -incidences counts of the surfaces in  $S$  and the vertices of a grid  $G$  of side length  $4\varepsilon$ .

We construct an octree decomposition of  $\tau_0 := [0, 1]^d$ , all the way to subcubes of side length  $4\varepsilon$  such that each vertex of  $G$  is the center of a leaf-cube. We propagate the surfaces of  $S^c$  down this octree, further rounding each of them within each subcube that it crosses.

The root of the octree corresponds to  $\tau_0$ , and we set  $S_{\tau_0} = S^c$ . At level  $j \geq 1$  of the recursion, we have subcubes  $\tau$  of  $\tau_0$  of side length  $\delta = 1/2^j$ . For each such  $\tau$ , we set  $\tilde{S}_\tau$  to be the subset of the surfaces in  $S_{p(\tau)}$  (that have been produced at the parent cube  $p(\tau)$  of  $\tau$ ) that intersect  $\tau$ . We now show how to further round the surfaces of  $\tilde{S}_\tau$ , so as to get a coarser set  $S_\tau$  of surfaces that we associate with  $\tau$ , and that we process recursively within  $\tau$ .

At any node  $\tau$  at level  $j$  of our rounding process, each surface  $\sigma$  of  $S_\tau$  is of the form  $x_j = H_j(\mathbf{x}; \mathbf{t}) + f_j$ , for  $j = k + 1, \dots, d$  where  $\mathbf{x} = (x_1, \dots, x_k)$ , and  $\mathbf{t} = (t_1, \dots, t_\ell)$ .

- (a) For each  $j = k + 1, \dots, d$  the function  $H_j$  is a translation of  $F_j$ . That is  $H_j(\mathbf{x}; \mathbf{t}) = F_j(\mathbf{x}; \mathbf{t}) + c$  for some constant  $c$ . Thus the gradients of  $H_j$  also satisfy Conditions (i) and (ii).
- (b)  $\mathbf{t}$  is some vector of  $\ell$  essential parameters, and each coordinate of  $\mathbf{t}$  is an integer multiple of  $\frac{\varepsilon'}{(\ell+1)\delta}$ , where  $\delta = 1/2^j$ .
- (c)  $\mathbf{f} = (f_{k+1}, \dots, f_d)$  is a vector of free parameters, each is a multiple of  $\varepsilon' / (\ell + 1)$ .

Note that the surfaces in  $S_{\tau_0} = S^c$ , namely the set of initial canonical surfaces constructed in Section 4.1, are of this form (for  $j = 0$  and  $H_j = F_j$ ). We get  $S_\tau$  from  $\tilde{S}_\tau \subseteq S_{p(\tau)}$  by the following steps. The first step just changes the presentation of  $\tau$  and  $\tilde{S}_\tau$ , and the following steps do the actual rounding to obtain  $S_\tau$ .

1. Let  $(\xi_1, \dots, \xi_k, \xi_{k+1}, \dots, \xi_d)$  be the point in  $\tau$  of smallest coordinates and set  $\xi = (\xi_1, \dots, \xi_k)$ . We rewrite the equations of each surface of  $\tilde{S}_\tau$  as follows:  $x_j = G_j(\mathbf{x}; \mathbf{t}) + f'_j$ , for  $j = k + 1, \dots, d$ , where  $G_j(\mathbf{x}; \mathbf{t}) = H_j(\mathbf{x}; \mathbf{t}) - H_j(\xi; \mathbf{t}) + \xi_j$ , and  $f'_j = f_j + H_j(\xi; \mathbf{t}) - \xi_j$ , for  $j = k + 1, \dots, d$ . Note that in this reformulation we have not changed the essential parameters, but we did change the free parameters from  $f_j$  to  $f'_j$ , where  $f'_j$  depends on  $f_j$ ,  $\mathbf{t}$ ,  $\xi$ , and  $\xi_j$ . Note also that  $G_j(\xi; \mathbf{t}) = \xi_j$  for  $j = k + 1, \dots, d$ .
2. We replace the essential parameters  $\mathbf{t}$  of a surface  $\sigma_{\mathbf{t}, \mathbf{f}}$  by  $\mathbf{s}$ , which we obtain by rounding each coordinate of  $\mathbf{t}$  to the nearest integer multiple of  $\frac{\varepsilon'}{(\ell+1)\delta}$ . So the rounded surface has the equations  $x_j = G_j(\mathbf{x}; \mathbf{s}) + f'_j$ , for  $j = k + 1, \dots, d$ . Note that we also have that  $G_j(\xi; \mathbf{s}) = \xi_j$ , for  $j = k + 1, \dots, d$ .
3. For each surface, we round each free parameter  $f'_j$ ,  $j = k + 1, \dots, d$ , to an integral multiple of  $\frac{\varepsilon'}{\ell+1}$ , and denote the rounded vector by  $\mathbf{g}$ . Our final equations for each rounded surface that we put in  $S_\tau$  are  $x_j = G_j(\mathbf{x}; \mathbf{s}) + g_j$  for  $j = k + 1, \dots, d$ .



## 8:12 Approximation Algorithms for Camera Posing

By construction, when  $\mathbf{t}_1$  and  $\mathbf{f}'_1$  and  $\mathbf{t}_2$  and  $\mathbf{f}'_2$  get rounded to the same vectors  $\mathbf{s}$  and  $\mathbf{g}$  then the corresponding two surfaces in  $\tilde{S}_\tau$  get rounded to the same surface in  $S_\tau$ . The weight of each surface in  $S_\tau$  is the sum of the weights of the surfaces in  $S_{p(\tau)}$  that got rounded to it, which, by induction, is the number of original surfaces that are recursively rounded to it. In the next step of the recursion the  $H_j$ 's of the parametrization of the surfaces in  $S_\tau$  are the functions  $G_j$  defined above.

The total weight of the surface in  $S_\tau$  for a leaf cell  $\tau$  is the approximate  $\varepsilon$ -incidences count that we associate with the center of  $\tau$ .

### 4.3 Error analysis

We now bound the error incurred by our discretization. We start with the following lemma, whose proof is given in Appendix A of the full version [2].

► **Lemma 6.** *Let  $\tau$  be a cell of the octtree and let  $x_j = G_j(\mathbf{x}; \mathbf{t}) + f'_j$ , for  $j = k+1, \dots, d$  be a surface obtained in Step 1 of the rounding process described above. For any  $\mathbf{x} = (x_1, \dots, x_k) \in [0, \delta]^k$ , for any  $\mathbf{t}, \mathbf{s} \in [0, 1]^\ell$ , and for each  $j = k+1, \dots, d$ , we have*

$$|G_j(\mathbf{x}; \mathbf{s}) - G_j(\mathbf{x}; \mathbf{t})| \leq c_2 |\mathbf{x} - \xi| \cdot |\mathbf{t} - \mathbf{s}|, \quad (5)$$

where  $c_2$  is the constant of Condition (ii), and  $\xi = (\xi_1, \dots, \xi_k)$  consists of the first  $k$  coordinates of the point in  $\tau$  of smallest coordinates.

► **Lemma 7.** *For any  $\mathbf{x} = (x_1, \dots, x_k) \in [0, \delta]^k$ , for any  $\mathbf{t}, \mathbf{s} \in [0, 1]^\ell$ , and for each  $j = k+1, \dots, d$ , we have*

$$|G_j(\mathbf{x}; \mathbf{s}) + g_j - (G_j(\mathbf{x}; \mathbf{t}) + f'_j)| \leq c_2 \varepsilon' \leq \frac{\varepsilon}{\log(1/\varepsilon)}, \quad (6)$$

where  $c_2$  is the constant of Condition (ii).

**Proof.** Using the triangle inequality and Lemma 6, we get that

$$\begin{aligned} & |G_j(\mathbf{x}; \mathbf{s}) + g_j - (G_j(\mathbf{x}; \mathbf{t}) + f'_j)| \\ & \leq |G_j(\mathbf{x}; \mathbf{s}) - G_j(\mathbf{x}; \mathbf{t})| + |g_j - f'_j| \leq c_2 |\mathbf{x} - \xi| |\mathbf{t} - \mathbf{s}| + \frac{\varepsilon'}{\ell+1}. \end{aligned}$$

Since  $|\mathbf{x} - \xi| \leq \delta$ ,  $|\mathbf{t} - \mathbf{s}| \leq \frac{\ell \varepsilon'}{(\ell+1)\delta}$ , and  $|g_j - f'_j| \leq \frac{\varepsilon'}{\ell+1}$ , the lemma follows. ◀

We now bound the number of surfaces in  $S_\tau$ . Since  $\mathbf{s} \in [0, 1]^\ell$  and each of its coordinates is a multiple of  $\frac{\varepsilon'}{(\ell+1)\delta}$ , we have at most  $(\frac{\delta}{\varepsilon'})^\ell$  different values for  $\mathbf{s}$ . To bound the number of possible values of  $\mathbf{g}$ , we prove the following lemma (see [2] for the proof).

► **Lemma 8.** *Let  $x_j = G_j(\mathbf{x}; \mathbf{t}) + f'_j$ , for  $j = k+1, \dots, d$ , be a surface  $\sigma_{\mathbf{t}, \mathbf{f}}$  in  $\tilde{S}_\tau$ . For each  $j = k+1, \dots, d$ , we have  $|f'_j| \leq (c_1 + 1)\delta$ , where  $c_1$  is the constant of Condition (i).*

Lemma 8 implies that each  $g_j$ ,  $j = k+1, \dots, d$ , has only  $O(\frac{\delta}{\varepsilon'})$  possible values, for a total of at most  $O((\frac{\delta}{\varepsilon'})^{d-k})$  possible values for  $\mathbf{g}$ . Combining the number of possible values for  $\mathbf{s}$  and  $\mathbf{g}$ , we get that the number of newly discretized surfaces in  $S_\tau$  is

$$O\left(\left(\frac{\delta}{\varepsilon'}\right)^\ell \cdot \left(\frac{\delta}{\varepsilon'}\right)^{d-k}\right) = O\left(\left(\frac{\delta}{\varepsilon'}\right)^{\ell+d-k}\right). \quad (7)$$



It follows that each level of the recursive octree decomposition generates

$$O\left(\left(\frac{1}{\delta}\right)^d \cdot \left(\frac{\delta}{\varepsilon'}\right)^{\ell+d-k}\right) = O\left(\frac{\delta^{\ell-k}}{(\varepsilon')^{\ell+d-k}}\right)$$

re-discretized surfaces, where the first factor in the left-hand side expression is the number of cubes generated at this recursive level, and the second factor is the one in (7).

Summing over the recursive levels  $j = 0, \dots, \log \frac{1}{\varepsilon}$ , where the cube size  $\delta$  is  $1/2^j$  at level  $j$ , we get a total size of  $O\left(\frac{1}{(\varepsilon')^{\ell+d-k}} \sum_{j=0}^{\log \frac{1}{\varepsilon}} \frac{1}{2^{j(\ell-k)}}\right)$ . We get different estimates for the sum according to the sign of  $\ell - k$ . If  $\ell > k$  the sum is  $O(1)$ . If  $\ell = k$  the sum is  $O(\log \frac{1}{\varepsilon})$ . If  $\ell < k$  the sum is  $O(2^{j_{\max}(k-\ell)}) = O\left(\frac{1}{(\varepsilon')^{k-\ell}}\right)$ . Accordingly, the overall size of the structure, taking also into account the cost of the first phase, is

$$\begin{cases} O\left(\frac{1}{(\varepsilon')^{\ell+d-k}}\right) & \text{for } \ell > k \\ O\left(\frac{1}{(\varepsilon')^d} \log \frac{1}{\varepsilon}\right) & \text{for } \ell = k \\ O\left(\frac{1}{(\varepsilon')^d}\right) & \text{for } \ell < k. \end{cases} \quad (8)$$

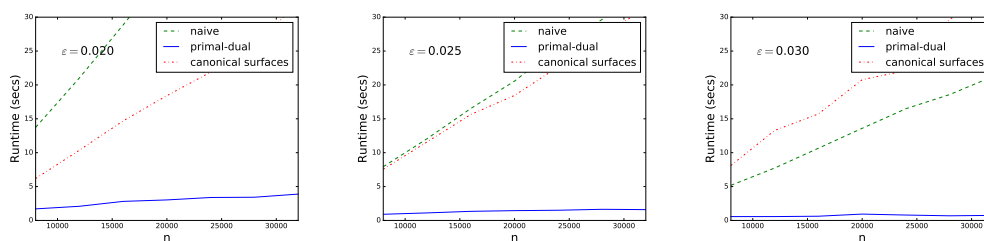
The following theorem summarizes the result of this section. Its proof follows in a straightforward way from the preceding discussion from Lemma 7, analogously to the proof of Lemma 8 in the appendix of the full version [2].

► **Theorem 9.** *Let  $S$  be a set of  $n$  surfaces in  $\mathbb{R}^d$  that cross the unit cube  $[0, 1]^d$ , given parametrically as  $x_j = F_j(\mathbf{x}; \mathbf{t}) + f_j$  for  $j = k + 1, \dots, d$ , where the functions  $F_j$  satisfy conditions (i) and (ii), and  $\mathbf{t} = (t_1, \dots, t_\ell)$ . Let  $G$  be the  $(4\varepsilon)$ -grid within  $[0, 1]^d$ . The algorithm described above reports for each vertex  $v$  of  $G$  an approximate  $\varepsilon$ -incidences count that includes all surfaces at distance at most  $\varepsilon$  from  $v$  and may include some surfaces at distance at most  $(2\sqrt{d} + 1)\varepsilon$  from  $v$ . The running time of this algorithm is proportional to the total number of rounded surfaces that it generates, which is given by Equation (8), plus an additive  $O(n)$  term for the initial canonization of the surfaces.*

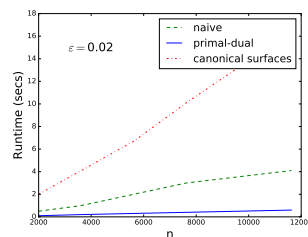
We can modify our data structure so that it can answer approximate or exact  $\varepsilon$ -incidence queries as we describe in Appendix C of [2] for the case of hyperplanes.

#### 4.4 Experimental Results: Summary

We implemented our algorithms and compared their performance on real and synthetic data. Some of our results are shown in Figures 3 and 4. They show that in practical scenarios ( $n \in [8K, 32K]$ ,  $\varepsilon \in [0.02, 0.03]$ ), the primal-dual scheme is considerably faster than the other algorithms. See full details in the full version of the paper [2].



■ **Figure 3** The run-time of the three methods on synthetic data for various values of  $\varepsilon$ .



■ **Figure 4** Run-time for real-world data.

---

## References

- 1 Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building Rome in a day. In *Commun. ACM* 54(10), pages 105–112. ACM, 2011.
- 2 D. Aiger, H. Kaplan, E. Kokopoulou, M. Sharis, and B. Zeisl. General techniques for approximate incidences and their application to the camera posing problem. *CoRR*, 2019. [arXiv:1903.07047](https://arxiv.org/abs/1903.07047).
- 3 Dror Aiger, Haim Kaplan, and Micha Sharir. Output Sensitive Algorithms for Approximate Incidences and Their Applications. In *Computational Geometry, to appear. Also in European Symposium on Algorithms*, volume 5, pages 1–13, 2017.
- 4 Guilherme D Da Fonseca and David M Mount. Approximate range searching: The absolute model. *Computational Geometry*, 43(4):434–444, 2010.
- 5 Martin A Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- 6 Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler, and Marc Pollefeys. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, 2017.
- 7 Bert M Haralick, Chung-Nan Lee, Karsten Ottenberg, and Michael Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356, 1994.
- 8 Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, pages 83–86. IEEE, 2009.
- 9 Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-DOF Localization on Mobile Devices. In *European Conference on Computer Vision*, pages 268–283. Springer, 2014.
- 10 Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual Modeling with a Hand-Held Camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.
- 11 Johannes L Schonberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.
- 12 Chris Sweeney, John Flynn, Benjamin Nuernberger, Matthew Turk, and Tobias Höllerer. Efficient Computation of Absolute Pose for Gravity-Aware Augmented Reality. In *ISMAR*, pages 19–24. IEEE, 2015.
- 13 Bernhard Zeisl, Torsten Sattler, and Marc Pollefeys. Camera Pose Voting for Large-Scale Image-Based Localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2704–2712, 2015.

# Circumscribing Polygons and Polygonizations for Disjoint Line Segments

**Hugo A. Akitaya**

Department of Computer Science, Tufts University, Medford, MA, USA  
hugo.alves\_akitaya@tufts.edu

**Matias Korman**

Department of Computer Science, Tufts University, Medford, MA, USA  
matias.korman@tufts.edu

**Mikhail Rudoy**

CSAIL, Massachusetts Institute of Technology, Cambridge, MA, USA  
Google Inc., Cambridge, MA, USA  
mrudoy@gmail.com

**Diane L. Souvaine**

Department of Computer Science, Tufts University, Medford, MA, USA  
diane.souvaine@tufts.edu

**Csaba D. Tóth**

Department of Mathematics, California State University Northridge, Los Angeles, CA  
Department of Computer Science, Tufts University, Medford, MA, USA  
csaba.toth@csun.edu

---

## Abstract

---

Given a planar straight-line graph  $G = (V, E)$  in  $\mathbb{R}^2$ , a **circumscribing polygon** of  $G$  is a simple polygon  $P$  whose vertex set is  $V$ , and every edge in  $E$  is either an edge or an internal diagonal of  $P$ . A circumscribing polygon is a **polygonization** for  $G$  if every edge in  $E$  is an edge of  $P$ .

We prove that every arrangement of  $n$  disjoint line segments in the plane has a subset of size  $\Omega(\sqrt{n})$  that admits a circumscribing polygon, which is the first improvement on this bound in 20 years. We explore relations between circumscribing polygons and other problems in combinatorial geometry, and generalizations to  $\mathbb{R}^3$ .

We show that it is NP-complete to decide whether a given graph  $G$  admits a circumscribing polygon, even if  $G$  is 2-regular. Settling a 30-year old conjecture by Rappaport, we also show that it is NP-complete to determine whether a geometric matching admits a polygonization.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Mathematics of computing  $\rightarrow$  Combinatoric problems

**Keywords and phrases** circumscribing polygon, Hamiltonicity, extremal combinatorics

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.9

**Related Version** A full version of the paper is available on arXiv [2], <http://arxiv.org/abs/1903.07019>.

**Funding** Research supported in part by the NSF awards CCF-1422311 and CCF-1423615.

*Matias Korman*: Partially supported by MEXT KAKENHI No. 17K12635.

*Diane L. Souvaine*: Partially supported by the Erwin Schrödinger Institute for Mathematics and Physics (ESI).



© H. A. Akitaya, M. Korman, M. Rudoy, C. D. Tóth, and D. L. Souvaine;  
licensed under Creative Commons License CC-BY

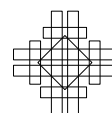
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 9; pp. 9:1–9:17

Leibniz International Proceedings in Informatics



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



## 1 Introduction

Reconstruction of geometric objects from partial information is a classical problem in computational geometry. In this paper, we revisit the problem of reconstructing a simple polygon (alternatively, a triangulated simple polygon)  $P$  when some of its edges have been lost. Given a set  $V$  of  $n$  points in the plane, a polygonization of  $V$  is a simple polygon  $P$  whose vertex set is  $V$ . It is easy to see that, unless all points are collinear,  $V$  has a simple polygonization. The number of polygonizations is exponential in  $n$ , and there is extensive work on determining the minimum and maximum number of polygonizations for  $n$  points in general position as a function of  $n$  (see [6] and [20] for the latest upper and lower bounds, and [9] for a survey on this and related problems).

A natural generalization of this problem is to augment a given planar straight-line graph (PSLG)  $G = (V, E)$  into a simple polygon or a Hamiltonian PSLG. In particular, three variants have been considered: A simple polygon  $P$  on a vertex set  $V$  is a **polygonization** if every edge in  $E$  is an edge of  $P$ ; a **circumscribing polygon** if every edge in  $E$  is an edge or an internal diagonal in  $P$ ; and a **compatible Hamiltonian polygon** if every edge in  $E$  is an edge, an internal diagonal, or an external diagonal in  $P$ .

Hoffmann and Tóth [8] proved that every planar straight-line matching admits a compatible Hamiltonian polygon, unless all segments are collinear, in which case no such polygon exists. Urabe and Watanabe [21] constructed an arrangement of 16 disjoint segments that does not admit a circumscribing polygon. However, a circumscribing polygon is known to exist when (i) each segment has at least one endpoint on the boundary of the convex hull [13], or (ii) no segment intersects the supporting line of any other segment [14]. Pach and Rivera-Campo [16] proved in 1998 that every set of  $n$  disjoint segments contains a subset of  $\Omega(n^{1/3})$  segments that admits a circumscribing polygon; no nontrivial upper bound is known.

Rappaport [18] proved that it is NP-complete to decide whether  $G$  can be augmented into a simple polygon. In the reduction,  $G$  consists of disjoint paths, and Rappaport conjectured that the problem remains hard even if  $G$  is a perfect matching (i.e., disjoint line segments in the plane). In the special case that  $G$  is perfect matching and every segment has at least one endpoint on the boundary of the convex hull, then an  $O(n \log n)$  time algorithm can compute a polygonization (or report that none exists [19]). If  $S$  is a set of  $n \geq 3$  parallel chords of a circle, then neither  $S$  nor any subset of 3 or more segments from  $S$  admits a polygonization (so the analogue of the problem of Pach and Rivera-Campo [16] has a trivial answer in this case). In a related result, Ishaque et al. [10] proved that  $n$  disjoint line segments in general position, where  $n$  is even, can be augmented to a 2-regular PSLG (i.e., a union of disjoint simple polygons).

**Our Results.** In this paper, we obtain the following results.

- We prove that every set of  $n$  disjoint line segments in general position contains a subset of  $\Omega(\sqrt{n})$  segments that admit a circumscribing polygon (Theorem 1 in Section 2). This is the first improvement over the previous bound of  $\Omega(n^{1/3})$  [16] in the last 20 years.
- While we do not have any nontrivial upper bound for circumscribing polygons proper, we relate that problem to the extensibility of disjoint line segments to disjoint rays. For every  $n \in \mathbb{N}$ , we construct a set of  $n$  disjoint line segments in the plane such that the size of any subset extensible to disjoint rays is  $O(\sqrt{n})$  (Section 3).
- We prove that it is NP-complete to determine whether a given set of disjoint cycles in the plane admits a circumscribing polygon (Theorem 13 in Section 4). The reduction is from Hamiltonian paths in 3-connected cubic planar graphs.
- We prove that it is NP-complete to determine whether a given set of disjoint line segments admits a polygonization (Theorem 14 in Section 5). This settles a 30-year old conjecture by Rappaport [18] in the affirmative.

We conclude with a few open problems and three-dimensional generalizations in Section 6. All omitted proofs are available in the full version of this paper [2].

**Further Related Previous Work.** Hamiltonicity fascinated graph theorists and geometers for centuries. Some planar graph results hold for PSLGs, as well (i.e., planar graphs with a fixed straight-line embeddings). Hamiltonicity is NP-complete for planar cubic graphs [7], but can be solved in linear time in 4-connected planar graphs [4], and all 4-connected triangulations (i.e., edge-maximal planar graphs) are Hamiltonian [22]. In terms of augmentation, a non-Hamiltonian triangulation cannot be augmented to a Hamiltonian planar graph by adding edges or vertices. However, Pach and Wenger [17] proved that every planar graph on  $n$  vertices can be transformed into a Hamiltonian planar graph on at most  $5n$  vertices by subdividing some of the edges, with at most two new vertices per edge, and by adding new edges. See also the surveys [5, 15] on Hamiltonicity of planar graphs and their applications.

## 2 Large Subsets with Circumscribing Polygons

For every integer  $n \geq 2$ , let  $f(n)$  be the maximum integer such that every set of  $n$  disjoint segments in the plane in general position contains a subset of  $f(n)$  segments that admit a circumscribing polygon. Pach and Rivera-Campo [16] proved 20 years ago that  $f(n) = \Omega(n^{1/3})$ . In this section, we improve the lower bound to  $f(n) = \Omega(\sqrt{n})$ .

► **Theorem 1.** *Every set of  $n \geq 2$  disjoint line segments in the plane in general position contains  $\Omega(\sqrt{n})$  segments that admit a circumscribing polygon.*

**Proof. Segment Selection.** Let  $S$  be a set of  $n \geq 2$  disjoint line segments in the plane. We may assume without loss of generality that none of the segments is vertical, and all segment endpoints have distinct  $x$ -coordinates. For a subset  $S' \subseteq S$ , a **halving line** is a vertical line  $\ell$  such that the number of segments in  $S'$  contained in the left and right open halfplanes bounded by  $\ell$  differ by at most one. In particular, each halfplane contains at most  $|S'|/2$  segments from  $S'$ .

We partition  $S$  recursively as follows. Find a halving line  $\ell$  for  $S$ , and recurse on the nonempty subsets of segments lying in each open halfplane determined by  $\ell$ . Denote by  $T$  the recursion tree, which is a binary tree of depth at most  $\log n$ . We denote by  $V(T)$  the set of nodes of  $T$ , and by  $V_i(T)$  the set of nodes at level  $i$  of  $T$  for  $i = 0, 1, \dots, \lfloor \log n \rfloor$ . Associate each node  $v \in V(T)$  to a halving line  $\ell_v$  and to the subset  $S_v \subseteq S$  of segments that intersect  $\ell_v$  without intersecting the halving lines associated with any ancestor of  $v$ . This defines a partition of  $S$  into subsets  $S_v$ ,  $v \in V(T)$ .

For every  $v \in V(T)$ , sort the segments in  $S_v$  by the  $y$ -coordinates of their intersections with the line  $\ell_v$ ; and let  $Q_v \subseteq S_v$  be a maximum subset of segments that have monotonically increasing or decreasing slopes. By the Erdős-Szkeres theorem, we have  $|Q_v| \geq \sqrt{|S_v|}$  for every  $v \in V(T)$ . For a refined analysis, we consider the union of the sets  $Q_v$  for  $v \in V_i(T)$  for  $i = 0, \dots, \lfloor \log n \rfloor$ , and then take one such union of maximal cardinality.

We need some additional notation. For every  $v \in V(T)$ , let  $n_v = |S_v|$  and  $m_v = |Q_v|$ . For every integer  $i = 0, 1, \dots, \lfloor \log n \rfloor$ , let  $\mathcal{S}_i$  (resp.,  $\mathcal{Q}_i$ ) be the union of  $S_v$  (resp.,  $Q_v$ ) over all vertices  $v \in V_i(T)$ . Let  $\nu_i = |\mathcal{S}_i|$  and  $\mu_i = |\mathcal{Q}_i|$ . By definition, we have  $n = \sum_{i=0}^{\lfloor \log n \rfloor} \nu_i$ .

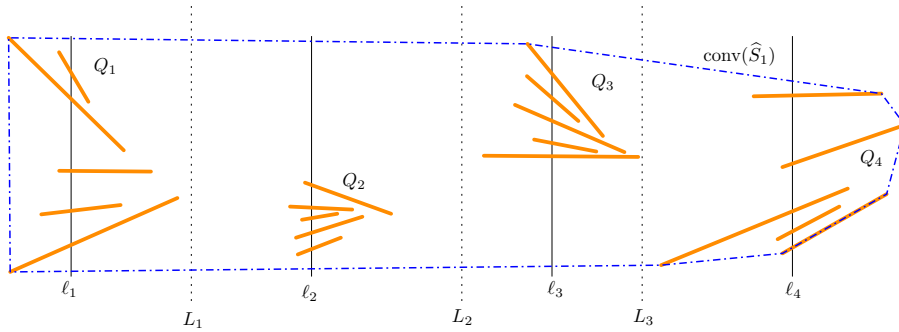
Let  $M = \max\{\mu_i : 0 \leq i \leq \lfloor \log n \rfloor\}$ . We claim that

$$M \geq \sqrt{n}/2. \tag{1}$$

By the Erdős-Szekeres Theorem, we have  $m_v \geq \sqrt{n_v}$  for every  $v \in V(T)$ . Since  $n_v \leq n/2^i$  for every  $v \in V_i(T)$ , then  $m_v \geq \sqrt{n_v} = n_v/\sqrt{n_v} \geq n_v/\sqrt{n/2^i} = \sqrt{2^i/n} \cdot n_v$ . Summation over all  $v \in V_i(T)$  yields  $M \geq \mu_i \geq \sqrt{2^i/n} \cdot \nu_i$ , which in turn gives  $\nu_i \leq M\sqrt{n/2^i}$ . Summation over all  $i = 0, \dots, \lfloor \log n \rfloor$  now gives  $n = \sum_{i=0}^{\lfloor \log n \rfloor} \nu_i \leq M\sqrt{n} \sum_{i=0}^{\lfloor \log n \rfloor} 2^{-i/2} \leq 2M\sqrt{n}$ , hence  $M \geq \sqrt{n}/2$ , which proves (1).

Let  $i^* \in \{0, 1, \dots, \lfloor \log n \rfloor\}$  be an index where  $M = \mu_{i^*}$ , and put  $\widehat{S}_0 = Q_{i^*}$ . By construction,  $\widehat{S}_0 = \bigcup\{Q_v : v \in V_{i^*}(T)\}$ . We further partition  $\widehat{S}_0$  into two subsets as follows. Let  $V_{i^*}^<$  (resp.,  $V_{i^*}^>$ ) be the set of nodes in  $V_{i^*}(T)$  such that the slopes in  $Q_v$  monotonically increase (resp., decrease). Let  $\widehat{S}_1$  be the larger of  $\bigcup\{Q_v : v \in V_{i^*}^<\}$  and  $\bigcup\{Q_v : v \in V_{i^*}^>\}$ , breaking ties arbitrarily. Note that  $|\widehat{S}_1| \geq \sqrt{n}/4$ . We may assume, by a reflection in the  $y$ -axis if necessary, that  $\widehat{S}_1 = \bigcup\{Q_v : v \in V_{i^*}^>\}$ ; see Fig. 1 for an example.

**Construction of a Circumscribing Polygon.** We construct a simple polygon that is a circumscribing polygon for a subset  $\widehat{S}_2 \subseteq \widehat{S}_1$  of size  $|\widehat{S}_2| \geq |\widehat{S}_1|/2 \geq \sqrt{n}/8$ . Pach and Rivera-Campo [16] proved that an arrangement of disjoint line segments admits a circumscribing polygon if they are (1) stabbed by a vertical line, and (2) have monotonically increasing or decreasing slopes. In particular, each  $Q_v, v \in V(T)$ , admits a circumscribing polygon. In contrast, we construct a circumscribing polygon for at least half of the segments in  $\widehat{S}_1$ , where  $\widehat{S}_1$  is the union of all  $Q_v, v \in V_{i^*}(T)$ , separated by vertical lines.



■ **Figure 1** A set  $\widehat{S}_1 = \bigcup\{Q_v : v \in V_{i^*}^>\}$  of 25 line segments for  $r = 4$ ; and  $P = \text{conv}(\widehat{S}_1)$ .

For ease of presentation, we introduce new notation for  $\widehat{S}_1 = \bigcup\{Q_v : v \in V_{i^*}^<\}$ ; see Fig. 1. Denote by  $\ell_1, \ell_2, \dots, \ell_r$  the halving lines  $\{\ell_v : v \in V_{i^*}^<\}$  sorted from left-to-right, and let  $Q_1, Q_2, \dots, Q_r$  be the corresponding sets in  $\{Q_v : v \in V_{i^*}^<\}$ . Denote by  $L_i$  ( $i = 1, \dots, r - 1$ ) the vertical lines that separate  $Q_i$  and  $Q_{i+1}$ . Refer to Fig. 1.

**Overview.** We construct a circumscribing polygon for a subset  $\widehat{S}_2 \subseteq \widehat{S}_1$  incrementally in two phases, while maintaining a polygon in the segment endpoint visibility graph of  $\widehat{S}_1$ . We use a machinery developed in [8], with several important new elements. Initially, let  $P$  be the boundary of  $\text{conv}(\widehat{S}_1)$ , which is a simple polygon. Intuitively, think of polygon  $P$  as a rubber band, and stretch it successively to visit more segment endpoints from  $\widehat{S}_1$ , maintaining the property that all segments in  $\widehat{S}_1$  remain in the closed polygonal domain of  $P$ . A key invariant of  $P$  will be that if  $P$  visits only one endpoint of some segment in  $\widehat{S}_1$ , then we can stretch it to visit the other endpoint (a strategy previously used in [8, 13]). This tool allows us to produce a circumscribing polygon for a subset of  $\widehat{S}_1$ . It is enough to ensure that  $P$  reaches an endpoint of **at least half** of the segments in  $\widehat{S}_1$ . To do this, we use the fact that each set  $Q_v, v \in V_{i^*}^>$ , is sorted along the halving lines in decreasing order by slope, and we ensure

that  $P$  reaches the left endpoint of at least half of the segments (later, we stretch  $P$  to visit the right endpoints). At the end, we define  $\widehat{S}_2$  as the set of segments in  $\widehat{S}_1$  visited by  $P$  (i.e., we discard the remaining segments lying in the interior of  $P$ ).

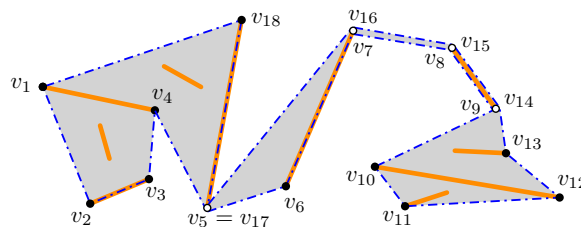
We maintain a polygon with the properties listed in Definition 2 below. There are a few important features to note:  $P$  is not necessarily a simple polygon in intermediate steps of the algorithm: it is a weakly simple polygon that does not have self-crossings; it has clearly defined interior and exterior; and it can have repeated vertices. Specifically, each vertex can repeat at most twice (i.e., multiplicity at most 2), and if its multiplicity is 2, then one occurrence is a reflex vertex and the other is convex. Furthermore, all such reflex vertices can be removed simultaneously by suitable shortcuts (cf. property (F5) below) to obtain a simple polygon. We need to be very careful about reflex vertices in  $P$ : for each reflex vertex in  $P$ , we ensure either that it will not become a repeated vertex later, or that if it becomes a repeated vertex, then its reflex occurrence can be removed by a suitable shortcut.

**Invariants.** As in [8], we maintain a weakly simple polygon, called a **frame** (defined below). A weakly simple polygon is a closed polygonal chain  $P = (v_1, \dots, v_k)$  in counterclockwise order such that, for every  $\varepsilon > 0$ , displacing the vertices by at most  $\varepsilon$  can produce a simple polygon. Denote by  $\widehat{P}$  the union of the interior and the boundary of  $P$ . A weakly simple polygon may have repeated vertices. Three consecutive vertices  $(v_{i-1}, v_i, v_{i+1})$  define an interior angle  $\angle(v_{i-1}, v_i, v_{i+1})$ , or  $\angle v_i$ , which is either convex ( $\leq 180^\circ$ ) or reflex ( $> 180^\circ$ ).

The following definitions summarizes the properties that we maintain for a polygon  $P$ . It is based on a similar concept in [8]: we do not allow segments to be external diagonal (cf. (F2)) and relax the conditions on the possible occurrences of reflex vertices. Property (F6) is related to the vertical lines  $\ell_i$  ( $i = 1, \dots, r - 1$ ) in the instance  $S = \bigcup_{i=1}^r Q_i$ . Reflex vertices play an important role. We distinguish two types of reflex vertices: A reflex vertex  $v$  of a frame  $P$  is **safe** if the (unique) line segment in  $S$  incident to  $v$  subdivides the reflex angle  $\angle v$  into two convex angles; otherwise  $v$  is **unsafe**.

► **Definition 2.** A weakly simple polygon  $P = (v_1, \dots, v_k)$  is called **frame** for a set  $S$  of disjoint line segments in the plane, if (cf. Fig. 2)

- (F1) every vertex of  $P$  is an endpoint of some segment in  $S$ ;
- (F2)  $\widehat{P}$  contains every segment in  $S$ ;
- (F3) every vertex in  $P$  has multiplicity at most 2;
- (F4) if a vertex in  $P$  has multiplicity 2, say  $v_i = v_j$ , then one of  $\angle v_i$  or  $\angle v_j$  is convex (and the other angle is reflex);
- (F5) if  $(v_i, \dots, v_j)$  is a maximal chain of unsafe reflex vertices of  $P$  that each have multiplicity 2, then  $(v_{j+1}, v_j, \dots, v_i, v_{i-1})$  is a simple polygon that is interior-disjoint from  $P$ ;
- (F6) the vertical line  $\ell_i$  ( $i = 1, \dots, r - 1$ ) crosses  $P$  exactly twice.

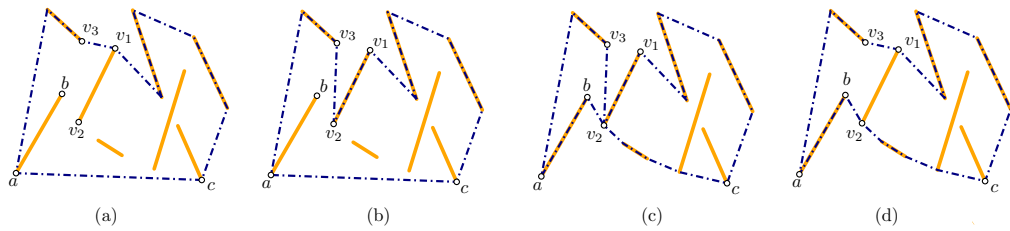


■ **Figure 2** A frame  $P = (v_1, \dots, v_{18})$  for 10 disjoint line segments (orange). The closed region  $\widehat{P}$  is shaded gray. The vertices of multiplicity 1 (resp., 2) are marked with full (resp., empty) dots.



**Elementary Operations.** Let  $S$  be a set of disjoint line segments in general position, and let  $P$  be a frame. We define four elementary operations that each transform  $P$  into a new frame for  $S$ . The first operation is the “shortcut” that eliminates reflex vertices of multiplicity 2, and increases the area of the interior. The remaining three operations each increase the number of vertices of the frame (possibly creating vertices of multiplicity 2) and decrease the area of its interior.

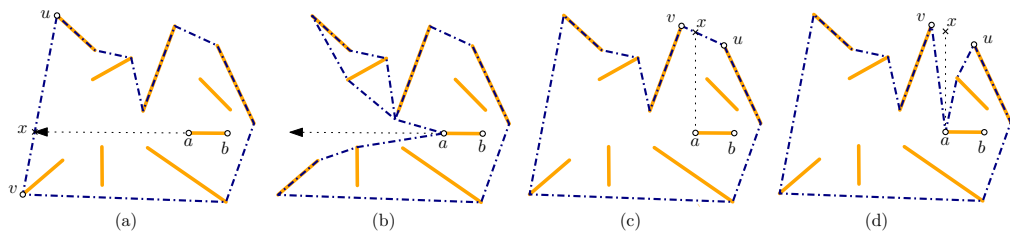
For shortest path and ray shooting computations, we consider the line segments in  $S$  and the current frame  $P$  to be obstacles. For a polygonal path  $(a, b, c)$  that does not cross any segment in  $S$ , we define the **convex arc**  $\text{carc}(a, b, c)$  to be the shortest polygonal path between  $a$  and  $c$  that is homotopic to  $(a, b, c)$ .



■ **Figure 3** (a) A frame  $P$ . (b)  $P := \text{BuildCap}(P, 1, v_1)$ . (c)  $P := \text{BuildCap}(P, 1, a)$ . (d)  $P := \text{ChopWedges}(P)$ .

► **Operation 1. ( $\text{ChopWedges}(P)$ )** Refer to Fig. 3(c-d). **Input:** a frame  $P$ . **Action:** While there is a vertex of multiplicity 2, do: let  $(v_i, \dots, v_j)$  be a maximal chain of convex vertices of  $P$  that each have multiplicity 2, and replace the path  $(v_{i-1}, v_i, \dots, v_j, v_{j+1})$  in  $P$  by a single edge  $v_{i-1}v_{j+1}$ .

► **Operation 2. ( $\text{BuildCap}(P, \varrho, a)$ )** Refer to Fig. 3(a-c) **Input:** a frame  $P$ , an orientation  $\varrho \in \{-1, +1\}$ , and a convex vertex  $a$  of multiplicity 1 in  $P$  such that  $ab \in S$  and  $b$  is not a vertex of  $P$ . **Action:** Let  $c$  be the neighbor of  $a$  in polygon  $P$  in orientation  $\varrho$  (where  $\text{ccw} = 1$ ,  $\text{cw} = -1$ ). Replace the edge  $ac$  of  $P$  with the polygonal path  $ab + \text{carc}(b, a, c)$ .



■ **Figure 4** (a) A frame. (b) The result of operation  $\text{Dip}(P, a, b)$ . (c) A frame. (d) The result of operation  $\text{ShearDip}(P, a, x)$ .

► **Operation 3. ( $\text{Dip}(P, a, b)$ )** Refer to Fig. 4(a-b). **Input:** a frame  $P$ , a segment  $ab \in S$  such that neither  $a$  nor  $b$  is a vertex of  $P$ , and the ray  $\vec{ba}$  hits an edge of  $P$  that is not a segment in  $S$ . **Action:** Assume that  $\vec{ba}$  hits edge  $uv$  of  $P$  at the point  $x$ . Replace the edge  $uv$  of  $P$  with the polygonal path  $\text{carc}(u, x, a) + \text{carc}(a, x, v)$ .



► **Operation 4.** (*ShearDip*( $P, a, x$ )) Refer to Fig. 4(c-d). **Input:** a frame  $P$ , a segment endpoint  $a$  in the interior of  $P$ , and a point  $x$  in the interior of an edge of  $P$  that is not a segment in  $S$  such that  $ax$  does not cross  $P$  or any segment in  $S$ . **Action:** Let  $uv$  be the edge of  $P$  that contains  $x$ . Replace the edge  $uv$  of  $P$  with the polygonal path  $\text{carc}(u, x, a) + \text{carc}(a, x, v)$ .

Operations 1 and 2 have been previously used in [8, 13]; it was shown that if **all** reflex vertices in  $P$  have been created by BuildCap operations, then property (F5) automatically holds [8, Sec. 2]. It is not difficult to see that if **all** reflex vertices in  $P$  have been created by Dip operations, then property (F5) holds. However, this property does not extend to a mixed sequence of BuildCap and Dip operations, and certainly not for ShearDip operations. We maintain property (F5) by a careful application of these operations, using the fact that each set  $Q_i$  ( $i = 1, \dots, r$ ) is stabbed by a vertical line.

Note that Operations 1–4 can only increase the vertex set of the frame (the shortcut operation decreases the multiplicity of repeated vertices from 2 to 1, but maintains the same vertex set). Initially,  $P = \partial\text{conv}(S)$ , and so all vertices of  $\text{conv}(S)$  remain vertices in  $P$  in our algorithm. In particular, the leftmost and rightmost segment endpoint in  $S$  are always vertices in  $P$  (with multiplicity 1 by property (F4)). These vertices subdivide  $P$  into an **upper arc** and a **lower arc**. As a convention, the leftmost (resp., rightmost) vertex is part of the lower arc (upper arc). We define an orientation for every vertex  $v$  in a frame  $P$ : If  $v$  is in the lower arc and the left endpoint of a segment in  $S$ , or if it is in the upper arc the right endpoint of a segment in  $S$ , then  $\varphi(v) = 1$ ; otherwise  $\varphi(v) = -1$ . When our algorithm invokes the BuildCap operation at a vertex  $v$ , we use  $\text{BuildCap}(P, \varphi(v), v)$ .

We now can justify the distinction between safe and unsafe reflex vertices.

► **Lemma 3.** *Let  $v$  be a reflex vertex of multiplicity 1 in a frame  $P$  such that  $v$  is safe. Then after any sequence of the above four operations, the multiplicity of  $v$  remains 1.*

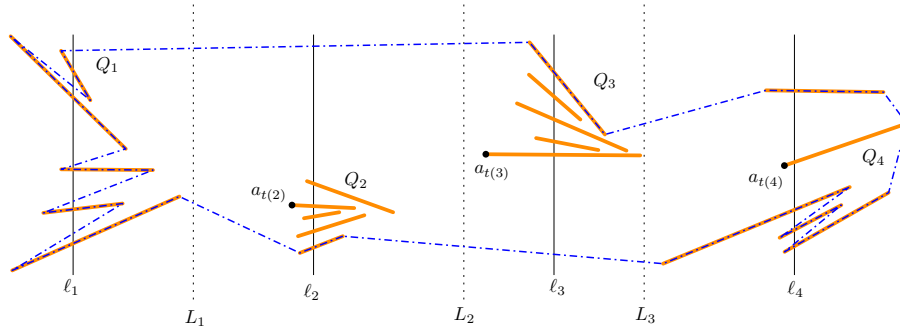
**Proof.** Each operation creates at most one new reflex vertex, which has multiplicity 1; and possibly many convex vertices along the convex arcs, which may have multiplicity 1 or 2. However, each point can be an interior vertex of at most one convex arc. Consequently, the multiplicity of a vertex  $v$  can possibly increase from 1 to 2 if it is first a reflex vertex of multiplicity 1, and then visited for a second time (by a convex arc) as a convex vertex; see Fig. 3(c) and Fig. 4(d) for examples. If  $v$  is a **safe** reflex vertex, then it cannot be an interior vertex of a convex arc, and so its multiplicity cannot increase from 1 to 2. ◀

**Phase 1: Left Endpoints.** Initially, the frame  $P$  is the boundary of the convex hull  $\text{conv}(S)$ . In the first phase of our algorithm, we use operations BuildCap and Dip as follows:

1. Let  $P = \partial\text{conv}(S)$ .
2. While condition (a) or (b) below is applicable, do:
  - a. If there exists a segment  $ab \in S$  such that the left endpoint  $a$  is a vertex of  $P$ , but the right endpoint is not, then set  $P := \text{BuildCap}(P, \varphi(a), a)$ .
  - b. Else if there exists a segment  $ab \in Q_i$  for some  $i \in \{1, \dots, r\}$  such that  $a$  is the left endpoint,  $a$  lies in the interior of  $P$ , and  $\vec{ba}$  hits an edge  $uv$  where  $uv \notin S$ , and the left endpoint of  $uv$  is an endpoint of some segment in  $Q_i$ , then set  $P := \text{Dip}(P, a, b)$ .
3. Return  $P$ , and terminate Phase 1.

An example is shown in Fig. 5. First we show that Phase 1 returns a frame.

► **Lemma 4.** *All operations in Phase 2 maintain properties (F1)–(F6) for  $P$ .*



■ **Figure 5** A set  $\widehat{S}_1$  of 25 line segments; and the frame  $P$  at the end of Phase 1.

We also make simple observation about the frame at the end of Phase 1:

► **Lemma 5.** *Let  $P$  be the frame returned by Phase 1, and let  $ab \in \widehat{S}_1$ . If the left endpoint of  $ab$  is a vertex of  $P$ , then it is a convex vertex of multiplicity 1.*

The next lemma helps identify the segments in  $Q_i$  whose left endpoints are **not** in  $P$ .

► **Lemma 6.** *Let  $P$  be the frame returned by Phase 1. Let  $i \in \{1, \dots, r\}$ , and let  $Q_i = \{a_j b_j : j = 1, \dots, |Q_i|\}$ , be sorted in increasing order by the  $y$ -coordinates of  $a_j b_j \cap \ell_i$ . If  $a_j$  is a vertex in the lower (resp., upper) arc of  $P$ , then so is  $a_{j'}$  for all  $j' < j$  (resp.,  $j' > j$ ).*

By Lemma 6, the line segments in  $\widehat{S}_1$  whose left endpoints are not in  $P$  form a continuous interval. That is, for every  $i \in \{1, \dots, r\}$ , there is a set of consecutive indices  $M_i \subseteq \{1, \dots, |Q_i|\}$  (possibly  $M_i = \emptyset$  or  $M_i = \{1, \dots, |Q_i|\}$ ) such that  $j \in M_i$  if and only if the left endpoint of  $a_j b_j$  is **not** in  $P$ . Let  $Q'_i = \{a_j b_j : j \in M_i\}$  and  $S' = \bigcup_{i=1}^r S'_i$

**Phase 2: Middle Segments.** In Phase 2, we use ShearDip and Dip operations to reach the left endpoints of **at least half** of the segments in  $S'$ , followed by BuildCap operation to reach the right endpoints of those segments if necessary. For  $i = 1, \dots, r$ , let  $a_{t(i)}$  be the leftmost left endpoint in the set  $S'_i$ . From some suitable point  $x_i$  on the upper or the lower arc of  $P$ , we use ShearDip( $P, a_{t(i)}, x_i$ ) to reach  $a_{t(i)}$ . Choose the points  $x_i, i = 1, \dots, r$  on the same (upper or lower) arc of  $P$  by comparing the number of segments in  $S'_i$  above and below  $a_{t(i)} b_{t(i)}$  for  $i = 1, \dots, r$ . The set of segments above and below are  $A$  and  $B$ , respectively, defined as follows:

$$A = \bigcup_{i=1}^r A_i, \text{ where } A_i = \{a_j b_j : j \geq \ell(i), j \in M_i, \},$$

$$B = \bigcup_{i=1}^r B_i, \text{ where } B_i = \{a_j b_j : j \leq \ell(i), j \in M_i, \}.$$

If  $|A| \geq |B|$ , then we reach the vertices  $a_{t(i)}$ , for  $i = 1, \dots, r$ , from the upper arc; otherwise we reach them from the lower arc. Without loss of generality, assume that  $|A| \geq |B|$ .

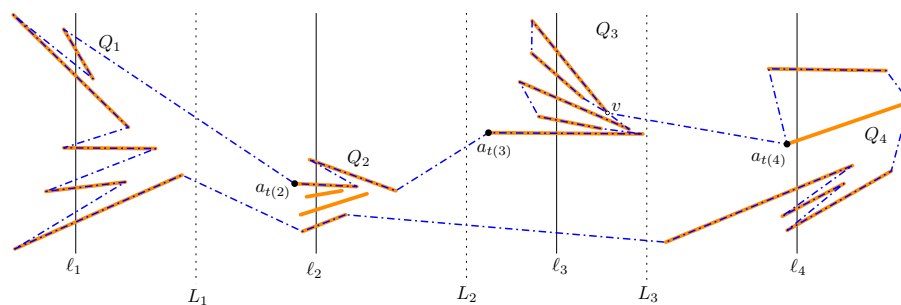
It remains to specify the points  $x_i$  for the operations ShearDip( $P, a_{t(i)}, x_i$ ). Consider the vertical upward ray from  $a_{t(i)}$ , and let  $u_i$  be the first point on the ray that lies in the upper arc of  $P$  or on a segment in  $S$ . If  $u_i$  is in an edge of the upper arc, but not in a segment in  $S$ , then let  $x_i := u_i$ . Otherwise,  $u_i$  lies in some segment  $ab \in Q_i$ , which is either an edge or an internal diagonal of  $P$ . Since  $a_{t(i)}$  is the leftmost left endpoint of a segment in  $Q_i$  that

is not a vertex in  $P$ , we know that the triangle  $\Delta(a_{t(i)}u_i a)$  is empty, and in particular  $a_{t(i)}$  sees vertex  $a$ . Furthermore,  $a$  is a convex vertex of  $P$  of multiplicity 1 (cf. Lemma 5). In this case, let  $x_i$  be an interior point of edge  $a_0 a$ . Phase 2 proceeds as follows:

1. For  $i = 1$  to  $r$ :
  - If  $M_i \neq \emptyset$ , then set  $P := \text{ShearDip}(P, a_{t(i)}, x_i)$ .
2. While condition (a) or (b) below is applicable, do:
  - a. If there exists a segment  $ab \in S$  such that the left endpoint  $a$  is a vertex of  $P$ , but the right endpoint is not, then set  $P := \text{BuildCap}(P, \varrho(a), a)$ .
  - b. Else if there exists a segment  $ab \in Q_i$  for some  $i \in \{1, \dots, r\}$  such that  $ab$  lies in the interior of  $P$ ,  $a$  is the left endpoint, and  $\vec{ba}$  hits an edge  $uv$  where  $uv \notin S$ , but both  $u$  and  $v$  are endpoints of some segments in  $Q_j$ ,  $j \geq i$ , then set  $P := \text{Dip}(P, a, b)$ .
3. Return  $P$ , and terminate Phase 2.

► **Lemma 7.** *All operations in Phase 2 maintain properties (F1)–(F6) for  $P$ ; and at the end of Phase 2, every vertex  $a_{t(i)}$ ,  $i = 1, \dots, r$ , has multiplicity 1 in  $P$ .*

► **Lemma 8.** *At the end of Phase 2,  $P$  visits both endpoints of all segments in  $A$ . Consequently,  $P$  visits both endpoints of at least half of the segments in  $\widehat{S}_1$ .*



■ **Figure 6** The set  $\widehat{S}_1$  from Fig. 5, and frame  $P$  at the end of Phase 2. Vertex  $v$  has multiplicity 2.

**Phase 3: Right Endpoints.** At the end of Phase 2,  $P$  visits the left endpoint of every segment in  $A$ , and none in  $B \setminus A$ . However, it may visit the **right** endpoint of some segments in  $B \setminus A$ . In this phase, we use BuildCap operations to ensure that  $P$  visits **both** endpoints of these segments. Phase 3 proceeds as follows.

1. While condition (a) below is applicable, do
  - a. If there exists a segment  $ab \in S$  such that the one endpoint, say  $b$ , is a vertex of  $P$ , but the other endpoint is not, then set  $P := \text{BuildCap}(P, \varrho(b), b)$ .
2. Return  $P$ , and terminate Phase 3.

At the end of Phase 3, we obtain a frame  $P$  that contains, for each segment, either both endpoints or neither endpoint; see Fig. 6. Some vertices may have multiplicity 2, but the multiplicity of the special vertices  $a_{t(i)}$  ( $i = 1, \dots, r$ ) remains 1.

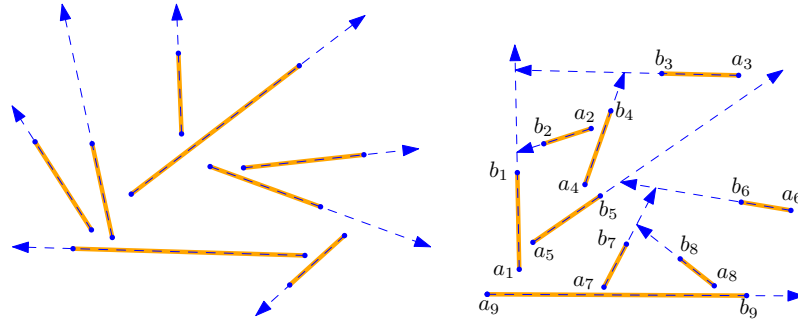
► **Lemma 9.** *All operations in Phase 3 maintain properties (F1)–(F6) for  $P$ ; and the multiplicity of every vertex  $a_{t(i)}$ ,  $i = 1, \dots, r$ , remains 1.*

► **Lemma 10.** *Let  $P$  be the frame at the end of Phase 3. If one endpoint of a segment in  $\widehat{S}_1$  is a vertex in  $P$ , then so is the other endpoint.*

**Phase 4: Obtaining a Simple Polygon.** In the last phase of our algorithm, we set  $P = \text{ChopWedges}(P)$ . This is a valid operation by property (F5). The resulting frame  $P$  is a simple polygon whose vertex set is the same as at the end of Phase 3. By Lemma 10, if one endpoint of a segment in  $\widehat{S}_1$  is a vertex in  $P$ , then so is the other endpoint. Consequently,  $P$  is a circumscribing polygon for a set of segments in  $\widehat{S}_1$ , which we denote by  $\widehat{S}_2$ . By Lemma 8, we have  $|\widehat{S}_2| \geq |\widehat{S}_1|/2$ , as claimed. This completes the proof of Theorem 1. ◀

### 3 Disjoint Segments versus Disjoint Rays

In this section, we give two sufficient conditions for an arrangement of disjoint segments to admit a circumscribing polygon. Both conditions involve extending the segments.



■ **Figure 7** Left: an arrangement of disjoint segments extensible to rays. Right: an arrangement of disjoint segments that is not extensible to rays, but admits escape routes.

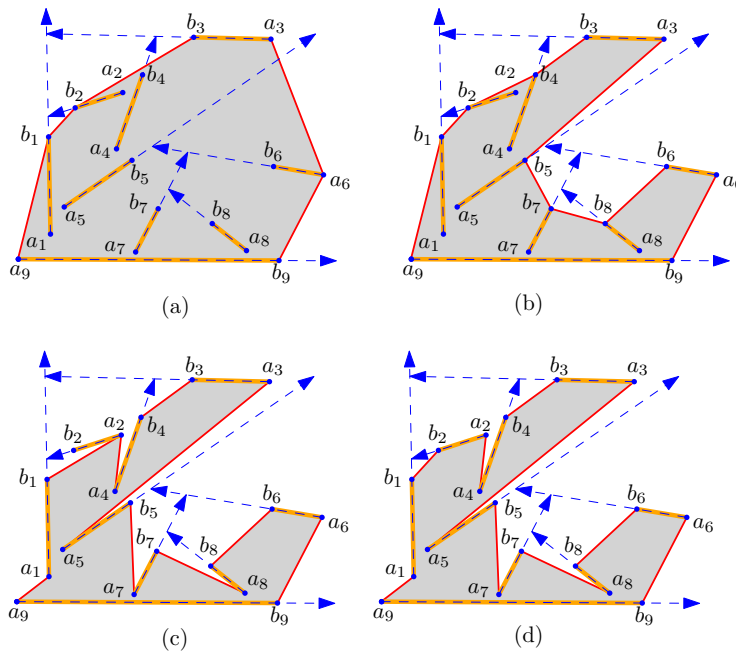
- (C1) A set  $S$  of  $n$  disjoint line segments is **extensible to rays** if there exists a set  $R$  of  $n$  disjoint rays, each of which contains a segment from  $S$ ; see Fig. 7(left).
- (C2) A set  $S$  of  $n$  disjoint line segments admits **escape routes** if there exists an ordering and orientation of the segments  $S = \{a_i, b_i : i = 1, \dots, n\}$  if the following process produces a set of  $n$  rays or directed segments that do not cross any segment in  $S$ : For  $i = 1, \dots, n$ , shoot a ray from  $b_i$  in direction  $\overrightarrow{a_i b_i}$  that ends at the first point where it hits a previous ray or goes to infinity; see Fig. 7(right).

Clearly, (C1) implies (C2), but the converse is false in general. We can test property (C1) in  $O(n \log n)$  time. Indeed, there are two possible directions to extend each segment into a ray, which can be encoded by a Boolean variable, and pairwise disjointness can be expressed by a 2SAT formula. For given ordering and orientation, it takes  $O(n \log^2 n)$  time to test whether the extensions form escape routes [11]. However, we do not know whether condition (C2) can be tested efficiently. Here we show that (C1) and (C2) each imply the existence of a circumscribing polygon.

► **Theorem 11.** *If  $S$  is a set of disjoint line segments satisfying (C2), then there is a circumscribing polygon for  $S$ .*

**Proof sketch.** By (C2), we may assume that  $S = \{a_i b_i : i = 1, \dots, n\}$  such that if we shoot a ray from  $b_i$  in direction  $\overrightarrow{a_i b_i}$  for  $i = 1, \dots, n$ , then each ray either goes to infinity or intersects a previous ray. We call the part of the ray  $\overrightarrow{a_i b_i}$  from  $b_i$  to the first point where it intersects a previous ray or  $\partial \text{conv}(S)$  the **extension** of segment  $a_i b_i$ .

Given the ordering and orientation of the segments in  $S$ , we construct a circumscribing polygon using the following algorithm, using the operations BuildCap and Dip introduced in Section 2; see Fig. 8 for an example.

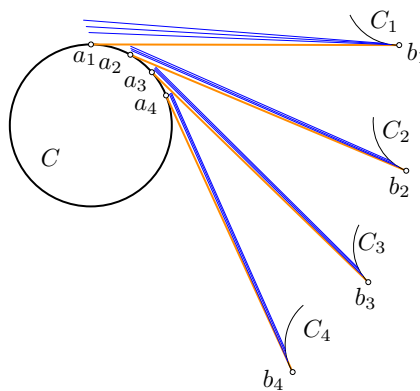


■ **Figure 8** (a) An arrangement of 9 segments that admit escape routes from Fig. 7(right), and  $P = \text{conv}(S)$ . (b) Polygon  $P$  after the `for` loop of Dip operations. (c) Polygon  $P$  after the `for` loop of BuildCap operations. (d) The circumscribing polygon for  $S$  after the ChopWedges operation.

1. Initialize  $P := \partial\text{conv}(S)$ .
2. For  $i = 1$  to  $n$ : if  $b_i$  is not a vertex of  $P$ , then set  $P := \text{Dip}(P, b_i, a_i)$ .
3. For  $i = 1$  to  $n$ : if  $a_i$  is not a vertex of  $P$ , then set  $P := \text{BuildCap}(P, 1, b_i)$ .
4. ChopWedges( $P$ ).
5. Return  $P$ . ◀

The above results link the circumscribing polygon problem to the problem of extending line segments to rays. We now give an upper bound for the latter problem that seems to imply that the lower bound of the former problem (Theorem 1) is tight.

► **Lemma 12.** *For every  $n \in \mathbb{N}$ , there is a set  $S$  of  $n$  disjoint line segments in the plane such that the cardinality of every subset  $S' \subseteq S$  that admits an escape route is  $|S'| \leq 2\lceil\sqrt{n}\rceil - 1$ .*



■ **Figure 9** Our lower bound construction for  $k = 4$ .

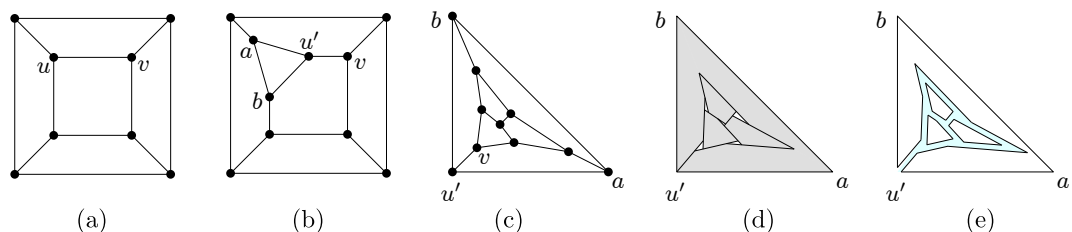
**Proof sketch.** We create a problem instance  $S$  with  $\sqrt{n}$  groups, each containing  $\sqrt{n}$  segments (see Fig. 9). The key idea is that in any subset  $Q$  of  $S$  that admits escape routes, only segments of one group can extend to the left. Moreover, we show that from each group only one segment can extend to the right, giving the  $2\lceil\sqrt{n}\rceil - 1$  bound. ◀

#### 4 Hardness for Circumscribing Polygons

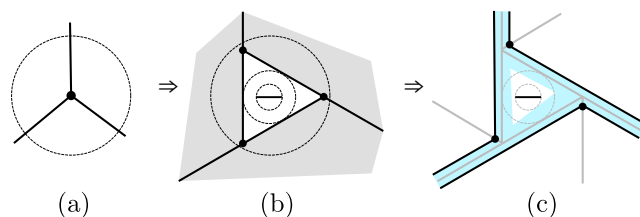
In this section we prove that it is NP-hard to decide whether a given set of disjoint line segments admits a circumscribing polygon. We reduce from a problem that we call HAMILTONIAN PATH IN 3-CONNECTED PLANAR CUBIC GRAPHS WITH START EDGE (HP3CPG-SE): Given a 3-connected cubic planar graph  $G = (V, E)$  and an edge  $uv \in E$ , decide whether  $G$  has Hamiltonian path whose first edge is  $uv$ , which is NP-complete [7, p. 713].

► **Theorem 13.** *It is NP-complete to decide whether a given PSLG admits a circumscribing polygon, even if the PSLG is max-degree-2.*

**Proof sketch.** The membership to NP is trivial. Let  $G = (V, E)$  and  $uv \in E$  be an instance of HP3CPG-SE, and let  $n = |V|$  (Fig. 10 (a)). We modify  $G$  by a  $Y\Delta$ -transform at  $u$  producing a graph  $G'$  with a new triangular face  $\Delta(abu')$  (Fig. 10 (b)). It is clear that  $G$  admits a Hamiltonian path starting with  $uv$  if and only if  $G'$  admits a Hamiltonian path starting with  $abu'v$ . Embed  $G'$  so that the outer face is  $\Delta(abu')$  (Fig. 10 (c)). Delete the edge incident to  $a$  ( $b$ ) that does not bound the outer face. Rotate by a small amount all edges that do not bound the outer face, splitting each internal degree-3 vertex  $w$  into a triangle  $\Delta(w_1w_2w_3)$  (Fig. 10(d) and Fig. 11(a–b)). We call these newly created triangles **transparent** faces, and every remaining internal face **opaque**. Create a small new edge in each transparent face, and shrink each opaque face by a small amount using its straight skeleton (Fig. 10 (e)). After this step, each added small segment can still only see the three vertices (Fig. 11 (c)). This construction defines a max-degree-2 PSLG  $\hat{G} = (\hat{V}, \hat{E})$ .

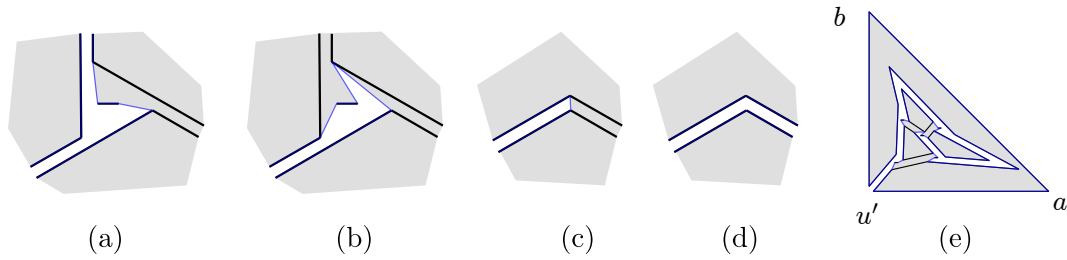


■ **Figure 10** (a) A 3-connected cubic graph  $G$  with special edge  $uv$ . (b) Graph  $G'$  after a  $Y\Delta$ -transform around vertex  $u$ . (c) A convex embedding of  $G'$  such that the outer face is  $\Delta_u$ . (d) Rotating the supporting line of internal edges. (e) Thickening the edges into corridors.



■ **Figure 11** A chamber incident to three corridors and a segment in the chamber that sees only three other vertices.

A circumscribing polygon of  $\widehat{G}$  must enclose all of its opaque faces. We partition the regions of the outer face of  $\widehat{G}$  that lies in the convex hull of the embedding into **corridors** and **chambers**. They correspond to edges and vertices of  $G'$  respectively. We show that a circumscribing polygon of  $\widehat{G}$  must behave as in Figs. 12 (a–d) in the vicinity of the endpoints of a corridor, up to symmetry. Then, such a circumscribing polygon defines a Hamiltonian path on the chambers. Hence,  $\widehat{G}$  admits a circumscribing polygon if and only if  $G$  admits a Hamiltonian path starting with  $uv$ . ◀

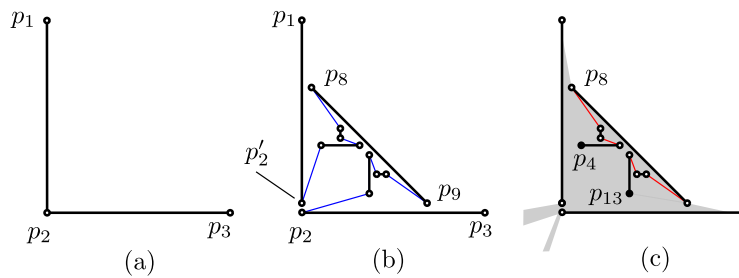


■ **Figure 12** A circumscribing polygon  $\widehat{P}$  at a chamber and a solution to the reduction in Fig. 10.

### 5 Simple Polygonizations of Disjoint Segments

Rappaport [18] proved that it is NP-hard to decide whether a given PSLG  $G = (V, E)$  admits a polygonization. The reduction in [18] is from HAMILTONIAN PATH IN PLANAR CUBIC GRAPHS (HPPCG) and produces an instance in which  $G$  is a union of disjoint paths, every edge in  $E$  is horizontal or vertical, and the vertices in  $V$  have integer coordinates, bounded by a polynomial in  $n = |V|$ . In this section, we describe the **connection gadget** made of disjoint line segments that simulates a pair of line segments that share an endpoint. Using this gadget, we show that finding a simple polygonization of disjoint line segments is NP-hard.

**Informal description of the connection gadget.** Refer to Figure 13. Given a PSLG  $G = (V, E)$ , with a vertex  $p_2 \in V$  of degree 2, incident to  $p_1p_2, p_2p_3 \in E$ , delete the edge  $p_1p_2$ , and insert 6 new edges  $p_1p'_2, p_4p_5, p_6p_7, p_8p_9, p_{10}p_{11}, p_{12}p_{13}$ , and 11 new vertices  $p'_2$  and  $p_i$  ( $i = 3, \dots, 13$ ). Denote by  $G' = (V', E')$  the resulting new PSLG. We choose the position of the new vertices close to  $p_2$  so that: (i) the two small segments  $p_6p_7$  and  $p_{10}p_{11}$  are only visible from points  $p_4, p_5, p_8$ , and  $p_9, p_{12}, p_{13}$  respectively; (ii) the union of the visibility regions of  $p_4, p_5, p_{12}$ , and  $p_{13}$  contain only vertices  $p_2, p'_2, p_4, \dots, p_{13}$  and no other vertices.



■ **Figure 13** (a) Two line segments  $p_1p_2$  and  $p_2p_3$ . (b) Connection gadget that simulates (a) using seven disjoint line segments. The polygonal path shown with black and blue line segments is  $[p_1, p'_2, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_2, p_3]$ . (c) The union of the visibility region of the solid black points  $p_4$  and  $p_{13}$ .



► **Theorem 14.** *It is NP-complete to decide whether a set  $S$  of disjoint line segments admits a simple polygonization, even if  $S$  contains only segments with 4 distinct slopes.*

**Proof.** Membership in NP is proven in [18]. We reduce NP-hardness from finding polygonizations for a disjoint union of paths. Let  $G = (V, E)$  be a PSLG produced by the reduction in [18], and let  $n = |V|$ . We modify  $G$  by simultaneously replacing every vertex of degree 2 by a connection gadget (described above), and show that the resulting planar straight-line matching  $M$  admits a polygonization if and only if  $G$  does. Since each gadget is constructed independently, all coordinates can be described by polynomials as they are each obtained by a constant number of intersections between lines and circles determined by  $G$ . Since  $G$  contains only axis-parallel edges, edges of  $M$  have up to four distinct slopes. The reduction runs in polynomial time.

We now show that  $M$  admits a polygonization if and only if  $G$  does. Note that the connection gadget places edges in the convex corner of a degree-2 vertex in  $G$ , and it does not block or create visibility between two leaves of  $G$ . By construction, if  $p$  is a leaf in  $G$ , then the set of other leaves visible from  $p$  remains same in  $M$ . Since  $G$  is max-degree-2, it remains to prove that, for every connection gadget, a polygonization of  $M$  must contain a chain of length 11 from  $p'_2$  to  $p_2$  that uses only edges of the connection gadget.

By property (i) of the connection gadget, if a simple polygonization  $P$  of  $M$  exists,  $P$  must connect  $p_8$  with  $p_6$  or  $p_7$ , and  $p_6$  or  $p_7$  to  $p_4$  or  $p_5$ , otherwise  $P$  would contain a cycle of length 4 and  $P$  would be disconnected. The same argument applies to vertices  $p_9, \dots, p_{13}$ . Fig. 13(c) shows the forced edges in a polygonization in red. By property (ii),  $p'_2$  must be adjacent to  $p_4$  or  $p_5$ , and  $p_2$  must be adjacent to  $p_{12}$  or  $p_{13}$ , or else either  $P$  would contain a cycle of length 10 and  $P$  would be disconnected, or  $P$  would not be simple. ◀

## 6 Conclusions

Our results raise interesting open problems, among others, about circumscribing polygons in the plane (Section 6.1), and about higher dimensional generalizations (Section 6.2).

### 6.1 Geometric Matching or Few Slopes

As noted above, Urabe and Watanabe [21] constructed an arrangement of 16 disjoint segments in  $\mathbb{R}^2$  that does not admit a circumscribing polygon. If all segments have the same slope (but they are not all collinear), then there always exists a circumscribing polygon. We conjecture that disjoint segments with two distinct slopes still admit a circumscribing polygon. Here we present negative instances with three slopes.

► **Proposition 15.** *For every  $n \geq 9$ , there is a set of  $n$  disjoint segments of 3 different slopes that do not admit a circumscribing polygon.*

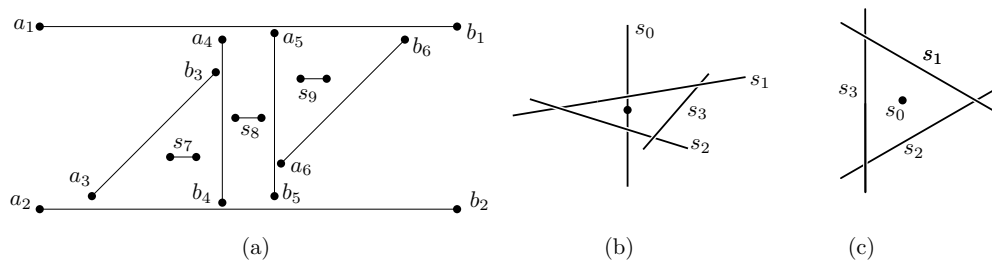
Our construction for  $n = 9$  is depicted in Figure 14(a).

### 6.2 Higher Dimensions

Generalizations to higher dimensions are also of interest. For a set  $V$  of points in  $\mathbb{R}^3$ , a **polyhedralization** is a polyhedron homotopic to a sphere whose vertex set is  $V$ <sup>1</sup>. It is known that every set of  $n \geq 4$  points in general position admits a polyhedralization [1], and even a polyhedralization of bounded vertex degree [3].

<sup>1</sup> We thank Joe Mitchell for introducing us to the high dimensional variations of this problem.





■ **Figure 14** (a) A set of 9 disjoint line segments of slopes 0, 1, and  $\infty$ , that do not admit a circumscribing polygon. (b–c) A set of 4 disjoint segments in  $\mathbb{R}^3$  that do not admit a polyhedralization: perspective view (b) and view from above (c).

For a set  $S$  of disjoint line segments in  $\mathbb{R}^3$ , we define a **polyhedralization** as a polyhedron homotopic to a ball whose vertices are the segment endpoints, and every segment in  $S$  is either an edge or an (external or internal) diagonal. A polyhedralization **circumscribes**  $S$  if every segment in  $S$  is an edge or an internal diagonal. It is not difficult to see that an arrangement of disjoint segments in general position in  $\mathbb{R}^3$  need not admit a polyhedralization.

► **Proposition 16.** *For every  $n \geq 4$ , there is a set of  $n$  disjoint segments in  $\mathbb{R}^3$  that do not admit a polyhedralization.*

Our construction is depicted in Figure 14(b–c).

We suspect that it is NP-hard to decide whether a set of  $n$  segments in  $\mathbb{R}^3$  admit a polyhedralization (or even a circumscribing polyhedralization). However, our proof techniques do not seem to extend to higher dimensions.

### 6.3 Open Problems

We conclude with a collection of open problems.

1. In Section 5 we established NP-hardness for the polygonization problem, even when the input consists of disjoint segments with four distinct slopes. Is it NP-hard to decide whether  $n$  disjoint *axis-parallel* segments in the plane admit a polygonization? Is it NP-hard for segments of 3 possible directions?
2. In Section 4 we proved that it is NP-complete to decide whether a 2-regular PSLG admits a circumscribing polygon. We do not know whether the problem remains hard for 1-regular PSLGs (i.e., disjoint line segments). The connection gadgets we designed for the polygonization problem (Section 4) do not seem to work for circumscribing polygons. Is it NP-hard to decide whether  $n$  disjoint segments admit a circumscribing polygon?
3. Does every arrangement of disjoint axis-parallel segments in  $\mathbb{R}^2$ , not all in a line, admit a circumscribing polygon?
4. Does every arrangement of disjoint line segments in  $\mathbb{R}^3$ , not all in a plane, admit a circumscribing polyhedron?
5. We can decide in  $O(n \log n)$  time whether  $n$  disjoint segments are extendible to disjoint rays (Section 3). Can we decide efficiently whether they admit escape routes?
6. Let  $f(n)$  be the maximum integer such that every set of  $n$  disjoint segments contains  $f(n)$  segments that admit a circumscribing polygon. In Section 2, we prove a lower bound of  $f(n) = \Omega(\sqrt{n})$ . Is it possible that  $f(n) = \Omega(n)$ ? Is there a nontrivial upper bound?

7. Let  $g(n)$  be the maximum integer such that every set of  $n$  disjoint segments contains  $g(n)$  segments that are extensible to disjoint rays. Theorem 11 implies  $g(n) \leq f(n)$ . PA Ramsey-type result on the intersection graph of rays [12, Remark 2] yields  $g(n) = \Omega(n^{1/3})$ , and Lemma 12 gives  $g(n) = O(\sqrt{n})$ . What is the asymptotic growth rate of  $g(n)$ ?
8. Let  $h(n)$  be the maximum integer such that every set of  $n$  disjoint segments contains  $h(n)$  segments that admit an escape route. Theorem 11 implies  $g(n) \leq h(n) \leq f(n)$ . We have  $h(n) = \Omega(n^{1/3})$  and  $h(n) = O(\sqrt{n})$ . What is the asymptotic growth rate of  $h(n)$ ?

---

### References

- 1 Pankaj K. Agarwal, Ferran Hurtado, Godfried T. Toussaint, and Joan Trias. On polyhedra induced by point sets in space. *Discrete Applied Mathematics*, 156(1):42–54, 2008. doi:10.1016/j.dam.2007.08.033.
- 2 Hugo A. Akitaya, Matias Korman, Mikhail Rudoy, Diane L. Souvaine, and Csaba D. Tóth. Circumscribing Polygons and Polygonizations. *CoRR*, abs/1903.07019, 2019. arXiv:1903.07019.
- 3 Gill Barequet, Nadia Benbernou, David Charlton, Erik D. Demaine, Martin L. Demaine, Mashhood Ishaque, Anna Lubiw, André Schulz, Diane L. Souvaine, Godfried T. Toussaint, and Andrew Winslow. Bounded-degree polyhedronization of point sets. *Comput. Geom.*, 46(2):148–153, 2013. doi:10.1016/j.comgeo.2012.02.008.
- 4 Norishige Chiba and Takao Nishizeki. The Hamiltonian cycle problem is linear-time solvable for 4-connected planar graphs. *Journal of Algorithms*, 10(2):187–211, 1989. doi:10.1016/0196-6774(89)90012-6.
- 5 Emilio Di Giacomo and Giuseppe Liotta. The Hamiltonian Augmentation Problem and Its Applications to Graph Drawing. In Md. Saidur Rahman and Satoshi Fujita, editors, *Proc. 4th International Workshop Algorithms and Computation (WALCOM)*, volume 5942 of *LNCS*, pages 35–46, Berling, 2010. Springer. doi:10.1007/978-3-642-11440-3\_4.
- 6 Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of  $K_N$ . *Comput. Geom.*, 16(4):211–221, 2000. doi:10.1016/S0925-7721(00)00010-9.
- 7 Michael R. Garey, David S. Johnson, and Robert E. Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM J. Comput.*, 5(4):704–714, 1976. doi:10.1137/0205049.
- 8 Michael Hoffmann and Csaba D. Tóth. Segment endpoint visibility graphs are Hamiltonian. *Comput. Geom.*, 26(1):47–68, 2003. doi:10.1016/S0925-7721(02)00172-4.
- 9 Ferran Hurtado and Csaba D. Tóth. Plane Geometric Graph Augmentation: A Generic Perspective. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 327–354. Springer, New York, 2013. doi:10.1007/978-1-4614-0110-0\_17.
- 10 Mashhood Ishaque, Diane L. Souvaine, and Csaba D. Tóth. Disjoint Compatible Geometric Matchings. *Discrete & Computational Geometry*, 49(1):89–131, 2013. doi:10.1007/s00454-012-9466-9.
- 11 Mashhood Ishaque, Bettina Speckmann, and Csaba D. Tóth. Shooting Permanent Rays among Disjoint Polygons in the Plane. *SIAM J. Comput.*, 41(4):1005–1027, 2012. doi:10.1137/100804310.
- 12 David Larman, Jiří Matoušek, János Pach, and Jenő Töröcsik. A Ramsey-Type Result for Convex Sets. *Bull. London Math. Soc.*, 26(2):132–136, 1994. doi:10.1112/blms/26.2.132.
- 13 Andranik Mirzaian. Hamiltonian Triangulations and Circumscribing Polygons of Disjoint Line Segments. *Comput. Geom.*, 2:15–30, 1992. doi:10.1016/0925-7721(92)90018-N.
- 14 Joseph O’Rourke and Jennifer Rippel. Two Segment Classes with Hamiltonian Visibility Graphs. *Comput. Geom.*, 4:209–218, 1994. doi:10.1016/0925-7721(94)90019-1.
- 15 Kenta Ozeki, Nico Van Cleemput, and Carol T. Zamfirescu. Hamiltonian properties of polyhedra with few 3-cuts—a survey. *Discrete Mathematics*, 341(9):2646–2660, 2018. doi:10.1016/j.disc.2018.06.015.

- 16 János Pach and Eduardo Rivera-Campo. On circumscribing polygons for line segments. *Comput. Geom.*, 10(2):121–124, 1998. doi:10.1016/S0925-7721(97)00023-0.
- 17 János Pach and Rephael Wenger. Embedding Planar Graphs at Fixed Vertex Locations. *Graphs and Combinatorics*, 17(4):717–728, 2001. doi:10.1007/PL00007258.
- 18 David Rappaport. Computing simple circuits from a set of line segments is NP-complete. *SIAM Journal on Computing*, 18(6):1128–1139, 1989. doi:10.1137/0218075.
- 19 David Rappaport, Hiroshi Imai, and Godfried T. Toussaint. Computing simple circuits from a set of line segments. *Discrete & Computational Geometry*, 5(3):289–304, 1990. doi:10.1007/BF02187791.
- 20 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: Perfect matchings, spanning cycles, and Kasteleyn’s technique. *J. Comb. Theory, Ser. A*, 120(4):777–794, 2013. doi:10.1016/j.jcta.2013.01.002.
- 21 Masatsugu Urabe and Mamoru Watanabe. On a Counterexample to a Conjecture of Mirzaian. *Comput. Geom.*, 2:51–53, 1992. doi:10.1016/0925-7721(92)90020-S.
- 22 Hassler Whitney. A Theorem on Graphs. *Annals of Mathematics, 2nd Ser.*, 32(2):378–390, 1931.



# Morphing Contact Representations of Graphs

**Patrizio Angelini**

University of Tübingen, Tübingen, Germany  
angelini@informatik.uni-tuebingen.de

**Steven Chaplick**

University of Würzburg, Würzburg, Germany  
steven.chaplick@uni-wuerzburg.de

**Sabine Cornelsen**

University of Konstanz, Konstanz, Germany  
sabine.cornelsen@uni-konstanz.de

**Giordano Da Lozzo**

Roma Tre University, Rome, Italy  
giordano.dalozzo@uniroma3.it

**Vincenzo Roselli**

Roma Tre University, Rome, Italy  
vincenzo.roselli@uniroma3.it

---

## Abstract

We consider the problem of morphing between contact representations of a plane graph. In a contact representation of a plane graph, vertices are realized by internally disjoint elements from a family of connected geometric objects. Two such elements touch if and only if their corresponding vertices are adjacent. These touchings also induce the same embedding as in the graph. In a morph between two contact representations we insist that at each time step (continuously throughout the morph) we have a contact representation of the same type.

We focus on the case when the geometric objects are triangles that are the lower-right half of axis-parallel rectangles. Such RT-representations exist for every plane graph and right triangles are one of the simplest families of shapes supporting this property. Thus, they provide a natural case to study regarding morphs of contact representations of plane graphs.

We study piecewise linear morphs, where each step is a linear morph moving the endpoints of each triangle at constant speed along straight-line trajectories. We provide a polynomial-time algorithm that decides whether there is a piecewise linear morph between two RT-representations of a plane triangulation, and, if so, computes a morph with a quadratic number of linear morphs. As a direct consequence, we obtain that for 4-connected plane triangulations there is a morph between every pair of RT-representations where the “top-most” triangle in both representations corresponds to the same vertex. This shows that the realization space of such RT-representations of any 4-connected plane triangulation forms a connected set.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Mathematics of computing → Graph algorithms

**Keywords and phrases** Contact representations, Triangulations, Planar morphs, Schnyder woods

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.10

**Related Version** A full version of the paper [5] is available at <https://arxiv.org/abs/1903.07595>.

**Funding** Work supported by DFG grant Ka812/17-1 and by MIUR-DAAD Joint Mobility Program n.57397196 (Angelini), by DFG grant WO 758/11-1 (Chaplick), and by MIUR Project “MODE” under PRIN 20157EFM5C, by MIUR Project “AHeAD” under PRIN 20174LF3T8, by MIUR-DAAD JMP N° 34120, and by H2020-MSCA-RISE project 734922 – “CONNECT” (Da Lozzo and Roselli).

**Acknowledgements** This work began at the Graph and Network Visualization Workshop (GNV’18) in Heiligkreuztal. We thank S. Felsner, N. Heinsohn, and A. Lubiw for interesting discussions.



© Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, and Vincenzo Roselli;

licensed under Creative Commons License CC-BY

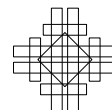
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 10; pp. 10:1–10:16

Leibniz International Proceedings in Informatics



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



## 1 Introduction

We consider the morphing problem from the perspective of geometric representations of graphs. While a lot of work has been done to understand how to planarly morph the standard *node-link* diagrams<sup>1</sup> of plane graphs<sup>2</sup> and to “rigidly” morph<sup>3</sup> configurations of geometric objects, comparatively little has been explicitly done regarding (non-rigid) morphing of alternative representations of planar graphs, e.g., contact systems of geometric objects such as disks or triangles. In this case, the planarity constraint translates into the requirement of continuously maintaining a representation of the appropriate type throughout the morph.

More formally, let  $\mathcal{F}$  be a family of geometric objects homeomorphic to a disk. An  $\mathcal{F}$ -*contact representation* of a plane graph  $G$  maps vertices to internally disjoint elements of  $\mathcal{F}$ . We denote the geometric object representing a vertex  $v$  by  $\Delta(v)$ . Objects  $\Delta(v)$  and  $\Delta(w)$  touch if and only if  $\{v, w\}$  is an edge. The contact system of the objects must induce the same faces and outer face as in  $G$ . A *morph* between two  $\mathcal{F}$ -contact representations  $R_0$  and  $R_1$  of a plane graph  $G$  is a continuously changing family of  $\mathcal{F}$ -contact representations  $R_t$  of  $G$  indexed by time  $t \in [0, 1]$ . An implication of the existence of morphs between any two representations of the same type is that the topological space defined by such representations is connected. We are interested in elementary morphs, and in particular in *linear morphs*, where the boundary points of the geometric objects move at constant speed along straight-line trajectories from their starting to their ending position. A *piecewise linear morph of length  $\ell$*  between two  $\mathcal{F}$ -contact representations  $R_1$  and  $R_{\ell+1}$  of a plane graph  $G$  is a sequence  $\langle R_1, \dots, R_{\ell+1} \rangle$  of  $\mathcal{F}$ -contact representations of  $G$  such that  $\langle R_i, R_{i+1} \rangle$  is a linear morph, for  $i = 1, \dots, \ell$ . For a background on the mathematical aspects of morphing, see, e.g., [3].

**Morphs of Node-Link Diagrams.** Fáry’s theorem tells us that every plane graph has a node-link diagram where the edges are mapped to line segments. Of course, for a given plane graph  $G$ , there can be many such node-link diagrams of  $G$ , and the goal of the work in planar morphing is to study how (efficiently) one can create a smooth (animated) transition from one such node-link diagram to another while maintaining planarity. Already in the 1940’s Cairns [17] proved that, for plane triangulations, planar morphs exist between any pair of such node-link diagrams. However, the construction involved exponentially-many morphing steps. Floater and Gotsman [29], and Gotsman and Surazhsky [31, 39] gave a different approach via Tutte’s graph drawing algorithm [42], but this involves non-linear trajectories of unbounded complexity. Thomassen [40] and Aronov et al. [10] independently showed that two node-link diagrams of the same plane graph have a *compatible triangulation*<sup>4</sup> thereby lifting Cairns’ result to plane graphs. Of particular interest is the study of *linear morphs*, where each vertex moves at a uniform speed along a straight-line. After several intermediate results to improve the complexity of the morphs [2, 6] and to remove the necessity of computing compatible triangulations [8], the current state of the art [1] is that there is a planar morph between any pair of node-link diagrams of any  $n$ -vertex plane graph using  $\theta(n)$  linear steps. Planar morphs of other specialized plane node-link diagrams have also been considered, e.g., planar orthogonal drawings [13, 43], convex drawings [7], upward planar drawings [21], and so-called *Schnyder drawings* [12]. In this latter result the lattice structure of all Schnyder

<sup>1</sup> where vertices are represented as points and edges as non-crossing curves

<sup>2</sup> the set of faces and the outer face are fixed

<sup>3</sup> scaling the objects is not allowed, e.g., as in *bar-joint* systems [35] or in *body-hinge* systems [14, 19, 24]

<sup>4</sup> i.e., a way to triangulate both diagrams to produce the same plane triangulation

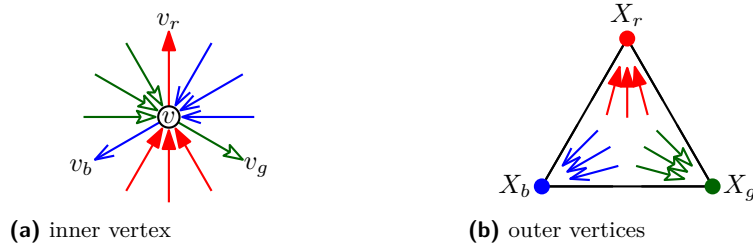
woods of a plane triangulation [15, 25] is exploited in order to obtain a sequence of linear morphs within a grid of quadratic size. Finally, planar morphs on the surface of a sphere [33] and in three dimensions have been investigated [11].

**Morphs of Contact Representations.** Similar to Fáry's theorem, the well-known Koebe-Andreev-Thurston theorem [4, 34] states that every plane graph  $G$  has a *coin* representation, i.e. an  $\mathcal{F}$ -contact representation where  $\mathcal{F}$  is the set of all disks. Additionally, for the case of 3-connected plane graphs, such coin representations of  $G$  are unique up to *Möbius transformations* [16] – see [27] for a modern treatment. There has been a lot of work on how to intuitively understand and animate such transformations (see, e.g., the work of Arnold and Rogness [9]), i.e., for our context, how to morph between two coin representations. Of course, ambiguity remains regarding how to formalize the complexity of such morphs. In particular, this connection to Möbius transformations appears to indicate that a theory of piecewise linear morphing for coin representations would be quite limited.

For this reason, we instead focus on contact representations of convex polygons. These shapes still allow for representing all plane triangulations, as a direct consequence of the Koebe-Andreev-Thurston theorem, but are more amenable to piecewise linear morphs, where the linearity is defined on the trajectories of the corners. De Fraysseix et al. [23] showed that every plane graph  $G$  has a contact representation by triangles, and observed that these triangle-contact representations correspond to the 3-orientations (i.e., the *Schnyder woods*) of  $G$ . Schrenzenmaier [38] used Schnyder woods to show that each 4-connected triangulation has a contact representation with homothetic triangles. Gonçalves et al. [30] extended the triangle-contact results from triangulations [23] to 3-connected plane graphs, by showing that Felsner's generalized Schnyder woods [25] correspond to *primal-dual* triangle-contact representations. Note that triangles and coins are not the only families of shapes that have been studied from the perspective of contact representations. Some further examples include boxes in  $\mathbb{R}^3$  [18, 26, 41], line segments [22, 32], and homothetic polygons [20, 28, 37].

The construction of triangle-contact representations [23] (and the correspondence to 3-orientations) can be adjusted so that each triangle is the lower-right half of an axis-parallel rectangle. These *right-triangle representations* (*RT-representations*) are our focus; see Fig. 3.

**Our Contribution and Outline.** The paper is organized as follows. We start with some definitions in Section 2 and describe the relationship between (degenerate) RT-representations and Schnyder woods of plane triangulations in Section 3. In Section 4, we provide necessary and sufficient conditions for a linear morph between two RT-representations. The first condition is that each corner  $c$  of a triangle touches the same side  $s$  of another triangle in the two representations, that is, the morph happens within the same Schnyder wood. The contact between  $c$  and  $s$  is always maintained when  $s$  has the same slope in the two RT-representations. Otherwise, we require the point of  $s$  hosting  $c$  to be defined by the same convex combination of the end-points of  $s$  in both representations. In Section 5, we present our morphing algorithm. If the two input RT-representations correspond to different Schnyder woods, we consider a path between them in the lattice structure of all Schnyder woods, similar to [12], that satisfies some properties (if it exists). When moving along this path, from a Schnyder wood to another, we construct intermediate RT-representations that simultaneously correspond to both woods. We provide an algorithm to construct such intermediate RT-representations that result in a linear morph at each step. Finally, in Section 6, we show how to decide whether there is a path in the lattice structure that satisfies the required properties. This results in an efficient testing algorithm for the existence of a piecewise linear morph



■ **Figure 1** The two conditions for a Schnyder wood.

between two RT-representations of a plane triangulation; in the positive case, the computed piecewise linear morph has at most quadratic length. Consequently, for 4-connected plane triangulations, under a natural condition on the outer face of their RT-representations, the topological space defined by such RT-representations is connected.

## 2 Definitions and Preliminaries

**Basics.** A *plane triangulation* is a maximal planar graph with a distinguished outer face. A *directed acyclic graph (DAG)* is an oriented graph with no directed cycles. A *topological ordering* of an  $n$ -vertex DAG  $G = (V, E)$  is a one-to-one map  $\tau : V \rightarrow \{1, \dots, n\}$  such that  $\tau(v) < \tau(w)$  for  $(v, w) \in E$ . Let  $p$  and  $q$  be two points in the plane. The *line segment  $\overline{pq}$*  is the set  $\{(1 - \lambda)p + \lambda q; 0 \leq \lambda \leq 1\}$  of convex combinations of  $p$  and  $q$ . Considering  $\overline{pq}$  oriented from  $p$  to  $q$ , we say that  $x$  *cuts  $\overline{pq}$  with the ratio  $\lambda$*  if  $x = (1 - \lambda)p + \lambda q$ .

In the case of polygons, a linear morph is completely specified by the initial and final positions of the corners of each polygon. If a corner  $p$  is at position  $p_0$  in the initial representation (at time  $t = 0$ ) and at position  $p_1$  in the final representation (at time  $t = 1$ ), then its position at time  $t$  during a linear morph is  $(1 - t)p_0 + tp_1$  for any  $0 \leq t \leq 1$ .

**Schnyder Woods.** A *3-orientation* [15, 25] of a plane triangulation is an orientation of the inner edges such that each inner vertex has out-degree 3 and the three outer vertices have out-degree 0. A *Schnyder wood*  $T$  [36] of a plane triangulation  $G$  is a 3-orientation together with a partition of the inner edges into three color classes, such that the three outgoing edges of an inner vertex have distinct colors and all the incoming edges of an outer vertex have the same color. Moreover, the color assignment around the vertices must be as indicated in Fig. 1. We say that a cycle in a Schnyder wood is *oriented* if it is a directed cycle.

The following well-known properties of Schnyder woods can directly be deduced from the work of Schnyder [36]. **1.** Every plane triangulation has a 3-orientation. **2.** For each 3-orientation of a plane triangulation there is exactly one partition of the inner edges into three color classes such that the pair yields a Schnyder wood. **3.** Each color class of a Schnyder wood induces a directed spanning tree rooted at an outer vertex. **4.** Reversing the edges of two color classes and maintaining the orientation of the third color class yields a directed acyclic graph. **5.** The edges of an oriented triangle in a Schnyder wood have three distinct colors and every triangle composed of edges of three different colors is oriented.

We call the color classes red ( $r, \rightarrow$ ), blue ( $b, \rightarrow$ ), and green ( $g, \rightarrow$ ). The symbols  $X_r$ ,  $X_b$ , and  $X_g$  denote the *red*, *blue*, and *green outer vertex* of  $G$ , i.e., the outer vertices with incoming red, blue, and green edges, respectively. For an inner vertex  $v$ , let  $v_r$ ,  $v_b$ , and  $v_g$  be the respective neighbors of  $v$  such that  $(v, v_r)$  is red,  $(v, v_b)$  is blue, and  $(v, v_g)$  is green. Finally, let  $\text{DAG}_r(T)$  ( $\text{DAG}_b(T)$ ) be the directed acyclic graph obtained from  $G$  by orienting all red (blue) edges as in  $T$  while all blue (red) and green edges are reversed.



Let  $C$  be an oriented triangle of a Schnyder wood  $T$ . Reversing  $C$  yields another 3-orientation with its unique Schnyder wood  $T_C$ . If  $C$  is a facial cycle, then  $T$  differs from  $T_C$  by recoloring the edges on  $C$  only. More precisely, the former outgoing edge of a vertex gets the color of the former incoming edge of the same vertex. This procedure of reversing and recoloring is called *flipping*<sup>5</sup> an oriented triangle of a Schnyder wood. Any Schnyder wood can be converted into any other Schnyder wood of the same plane triangulation by flipping  $\mathcal{O}(n^2)$  oriented triangles [12, 15]. For two Schnyder woods  $T_0$  and  $T_\ell$ ,  $\langle C_1, \dots, C_\ell \rangle$  is a *flip sequence between  $T_0$  and  $T_\ell$*  if there are Schnyder woods  $T_1, \dots, T_{\ell-1}$  such that  $C_i$ ,  $i = 1, \dots, \ell$ , is an oriented triangle in  $T_{i-1}$  and  $T_i$  is obtained from  $T_{i-1}$  by flipping  $C_i$ . We say that a Schnyder wood  $T'$  can be obtained from a Schnyder wood  $T$  by a sequence of facial flips if there is a flip sequence between  $T$  and  $T'$  that contains only facial cycles.

### 3 RT-Representations of Plane Triangulations

Let  $R$  be an RT-representation of a plane triangulation  $G$  and let  $u$  be a vertex of  $G$ . Recall that  $\Delta(u)$  is the triangle representing  $u$  in  $R$ . We denote by  $\perp(u)$ ,  $\lrcorner(u)$ , and  $\swarrow(u)$  the horizontal, vertical, and diagonal side of  $\Delta(u)$ . Further, we denote by  $\ulcorner(u)$ ,  $\lrcorner(u)$ , and  $\lrcorner(u)$ , the left, right, and top corner of  $\Delta(u)$ , respectively. If two triangles touch each other in their corners, we say that these two corners *coincide*. If there exist no two triangles whose corners coincide, then  $R$  is *non-degenerate*; otherwise, it is *degenerate*. Let  $(c, s)$  be a pair with  $c \in \{\ulcorner, \lrcorner, \lrcorner\}$  and  $s \in \{\lrcorner, \lrcorner, \swarrow\}$ , we say that  $(c, s)$  is a *compatible pair* if it belongs to the set  $\{(\lrcorner, \swarrow), (\ulcorner, \lrcorner), (\lrcorner, \lrcorner)\}$ . Observe that, in any RT-representation of  $G$ , if a corner  $c$  of a triangle  $\Delta(u)$  touches the side  $s$  of a triangle  $\Delta(v)$ , with  $(u, v) \in E(G)$ , then  $(c, s)$  is a compatible pair. We formally require this also in the case of a degeneracy. E.g., if  $\lrcorner(v)$  coincides with  $\ulcorner(u)$  for two vertices  $u$  and  $v$ , then the respective compatible pair is either  $(\ulcorner, \lrcorner)$  or  $(\lrcorner, \swarrow)$  – even though  $\lrcorner(v)$  also touches  $\lrcorner(u)$ , and  $\ulcorner(u)$  touches  $\lrcorner(v)$ .

In the next two subsections, we describe the relationship between RT-representations and Schnyder woods [23] and extend it to the case of degenerate RT-representations.

#### 3.1 From RT-Representations to Schnyder Woods

Let  $G = (V, E)$  be a plane triangulation with a given RT-representation  $R$ . It is possible to orient and color the edges of  $G$  in order to obtain a Schnyder wood by considering the types of contacts between triangles in  $R$  as follows.

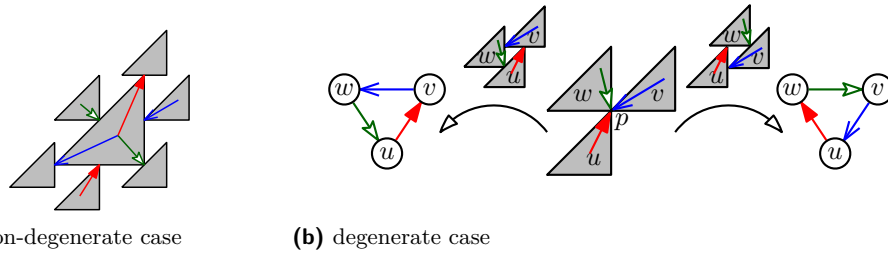
First, consider the non-degenerate case; refer to Fig. 2a. Let  $e = \{u, v\} \in E$  be an inner edge such that a corner  $c$  of  $\Delta(u)$  touches a side  $s$  of  $\Delta(v)$ . We use the following rules: We orient  $e$  from  $u$  to  $v$ , and color  $e$  **blue** if  $c$  is  $\ulcorner(u)$ , **green** if  $c$  is  $\lrcorner(u)$ , **red** if  $c$  is  $\lrcorner(u)$ .

► **Lemma 1** ([23], Theorem 2.2). *The above assignment yields a Schnyder wood.*

Assume now that there exist two triangles  $\Delta(u)$  and  $\Delta(v)$  whose corners coincide. Observe that the assignment of colors and directions to the edge  $\{u, v\}$  determined by the procedure above would be ambiguous. The next observation will be useful to resolve this ambiguity.

► **Observation 2.** *In an RT-representation of a plane triangulation, if the corner of a triangle  $\Delta(u)$  coincides with the corner of a triangle  $\Delta(v)$  in a point  $p$ , then there exists a triangle  $\Delta(w)$ ,  $w \neq u, v$ , with a corner on  $p$ , unless  $\{u, v\}$  is an edge of the outer face.*

<sup>5</sup> Brehm [15] called flipping a counter clockwise triangle a flip, and flipping a clockwise triangle a flop.



■ **Figure 2** From an RT-representation to a Schnyder wood.

By Observation 2, in a degenerate RT-representation there exist three vertices  $u$ ,  $v$ , and  $w$  such that  $\nearrow(u)$ ,  $\swarrow(v)$ ,  $\perp(w)$  lie on a point, see Fig. 2b. For each of the three edges, a choice of coloring and orientation corresponds to deciding which of the two triangles participates to the touching with its corner and which triangle with an extremal point of one of its sides. This yields two options as indicated in Fig. 2b, both resulting in a Schnyder wood. Note that the face  $f = \langle u, v, w \rangle$  is cyclic in both these Schnyder woods, and each of them can be obtained from the other by flipping  $f$ . Summarizing, we get the following.

► **Observation 3.** *Given an RT-representation  $R$  of a plane triangulation  $G$ , let  $P$  be the set of points where three triangles meet. Then,  $R$  corresponds to a set  $\mathcal{T}_R$  of  $2^{|P|}$  different Schnyder woods on  $G$ , the points of  $P$  correspond to  $|P|$  edge-disjoint oriented triangles, and the Schnyder woods in  $\mathcal{T}_R$  differ in flipping some of them.*

### 3.2 From Schnyder Woods to RT-Representations

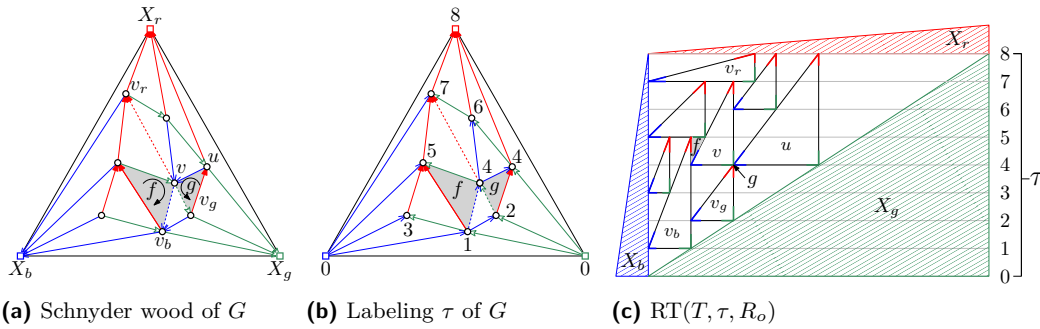
Assume now that we are given a Schnyder wood  $T$  of a plane triangulation  $G = (V, E)$ . We describe a technique for constructing an RT-representation of  $G$  corresponding to  $T$  in which the y-coordinate of the horizontal side of each triangle is prescribed by a function  $\tau : V \rightarrow \mathbb{R}$  satisfying some constraints; observe that in the non-degenerate case in [23]  $\tau$  is a topological labeling of  $\text{DAG}_r(T)$ , i.e., a canonical ordering of  $G$ .

We call  $\tau : V \rightarrow \mathbb{R}$  an *Admissible Degenerate Topological* labeling of the graph  $\text{DAG}_r(T)$ , for short *ADT-labeling*, if for each directed edge  $(u, v)$  of  $\text{DAG}_r(T)$ , we have **1.**  $\tau(u) \leq \tau(v)$  and **2.**  $\tau(u) = \tau(v)$  only if **a.** is green and belongs to a clockwise oriented facial cycle, or **b.** is blue and belongs to a counter-clockwise oriented facial cycle, and **3.** if  $\tau(u_b) = \tau(u) = \tau(u_g)$  for a vertex  $u$ , and  $u_1$  and  $u_2$  are vertices such that  $\langle u, u_g, u_1 \rangle$  is a clockwise facial cycle and  $\langle u, u_b, u_2 \rangle$  is a counter-clockwise facial cycle, then  $u_1 \neq u_2$ .

► **Lemma 4.** *Let  $R$  be an RT-representation of a plane triangulation  $G = (V, E)$ , let  $T$  be a Schnyder wood corresponding to  $R$ , and let  $\tau(v)$ ,  $v \in V$ , be the y-coordinate of  $\perp(v)$ . Then,  $\tau$  is an ADT-labeling of  $\text{DAG}_r(T)$ .*

**Proof.** Let  $(u, v)$  be a directed edge of  $\text{DAG}_r(T)$ . By the definition of  $T$ , we get immediately that  $\tau(u) \leq \tau(v)$  independently of whether  $(u, v)$  is red, green, or blue. In fact, if  $(u, v)$  is red, then it is oriented from  $u$  to  $v$  in  $T_r$ . Thus, the compatible pair corresponding to such an edge in  $R$  is  $(\nearrow(u), \perp(v))$ . Hence,  $\perp(u)$  lies strictly below  $\perp(v)$ . If  $(u, v)$  is green (resp., blue), then it is oriented from  $v$  to  $u$  in  $T$ . Thus, the compatible pair corresponding to such an edge in  $R$  is  $(\perp(v), \nearrow(u))$  (resp.,  $(\swarrow(v), \nearrow(u))$ ). Hence,  $\perp(u)$  does not lie above  $\perp(v)$ .

Assume that  $\tau(u) = \tau(v)$ , which implies that  $(u, v)$  is not red, as observed above. Suppose that  $(v, u)$  is a green edge. Then,  $\perp(v)$  and  $\swarrow(u)$  coincide. By Observation 2, there exists a vertex  $z$  such that  $\nearrow(z)$  coincides with  $\perp(v)$  and  $\swarrow(u)$ . Thus,  $\langle v, u, z \rangle$  is a clockwise



**Figure 3** (a) A Schnyder wood  $T$  of a plane triangulation  $G$ ; the edges connecting  $v$  with vertices  $v_r, v_g,$  and  $v_b$  are dashed. (b) Graph  $DAG_r(T)$  with ADT-labeling  $\tau$ . (c) An RT-representation of  $G$  constructed from  $T, \tau,$  and an RT-representation  $R_o$  of the outer face.

oriented facial cycle. Similarly, when  $(v, u)$  is a blue edge, there is a vertex  $z$  such that  $\angle(v), \lrcorner(u),$  and  $\nearrow(z)$  coincide. Therefore,  $\langle v, u, z \rangle$  is a counter-clockwise oriented facial cycle.

Finally, if  $\tau(u_b) = \tau(u) = \tau(u_g)$  and  $u_1$  and  $u_2$  are the vertices such that  $\langle u, u_g, u_1 \rangle$  is a clockwise facial cycle and  $\langle u, u_b, u_2 \rangle$  is a counter-clockwise facial cycle, then  $\nearrow(u_1)$  touches  $\lrcorner(u)$  and  $\nearrow(u_2)$  touches  $\angle(u)$ . Thus,  $u_1 \neq u_2$ .  $\blacktriangleleft$

**Lemma 5.** *Let  $T$  be a Schnyder wood of an  $n$ -vertex plane triangulation  $G$ , let  $\tau$  be an ADT-labeling of  $DAG_r(T)$ , and let  $R_o = \Delta(X_r) \cup \Delta(X_g) \cup \Delta(X_b)$  be an RT-representation of the outer face of  $G$  such that  $\triangle(X_i)$  has  $y$ -coordinate  $\tau(X_i)$ , with  $i \in \{r, g, b\}$ . Then, there exists a unique RT-representation  $RT(T, \tau, R_o)$  of  $G$  corresponding to  $T$  in which  $\triangle(v)$  has  $y$ -coordinate  $\tau(v)$ , for each vertex  $v$  of  $G$ , and in which the outer face is drawn as in  $R_o$ .*

**Outline of the Proof.** We process the vertices of  $G$  according to a topological ordering  $\tau'$  of  $DAG_r(T)$ . In the first two steps, we draw triangles  $\Delta(X_b)$  and  $\Delta(X_g)$  as in  $R_o$ ; see Fig. 3c. At each of the following steps, we consider a vertex  $v$ , with  $2 < \tau'(v) = i < n$ .

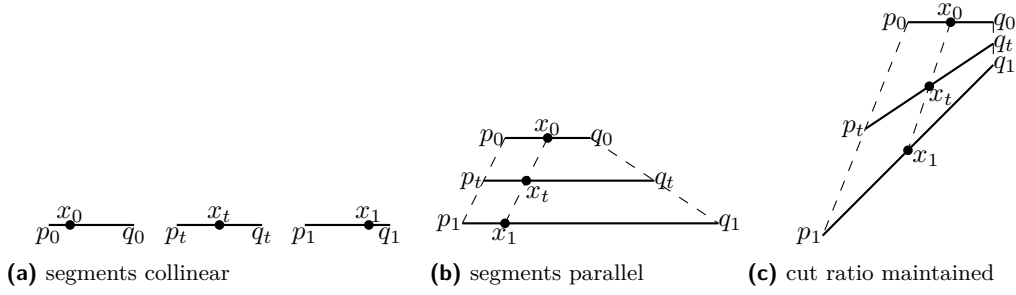
We draw  $\Delta(v)$  with its horizontal side on  $y = \tau(v)$  and with its top corner at  $y = \tau(v_r)$ , as follows. Since the blue edge  $(v_b, v)$  and the green edge  $(v_g, v)$  are entering  $v$  in  $DAG_r(T)$ , the triangles  $\Delta(v_b)$  and  $\Delta(v_g)$  have already been drawn. Also, by Property 1 of ADT-labeling, we have that  $\tau(v_b) \leq \tau(v)$  and  $\tau(v_g) \leq \tau(v)$ . Further, it can be shown that  $\nearrow(v_b)$  and  $\nearrow(v_g)$  have  $y$ -coordinate larger than or equal to  $\tau(v)$ , and that if a triangle  $\Delta(u)$  intersects the line  $y = \tau(v)$  between  $\Delta(v_b)$  and  $\Delta(v_g)$ , then  $u$  is a neighbor of  $v$  such that  $v = u_r$ . By construction,  $\nearrow(u)$  has  $y$ -coordinate equal to  $\tau(v)$ . Thus, we draw the horizontal side of  $\Delta(v)$  on  $y = \tau(v)$  between  $\Delta(v_b)$  and  $\Delta(v_g)$ . The conditions of ADT-labelings guarantee that  $\triangle(v)$  has positive length. If  $i = n$ , and hence  $v = X_r$ , we draw  $\Delta(X_r)$  as in  $R_o$ .  $\blacktriangleleft$

## 4 Geometric Tools

In this section, we provide geometric lemmata that will be exploited in the subsequent sections. We first show that the incidence of a point and a line segment is maintained during a linear morph if the line segment is moved in parallel (with a possible stretch, but keeping the orientation) or the ratio with which the point cuts the segment is maintained; see Fig. 4.

**Lemma 6.** *For  $i = 0, 1$  let  $p_i, q_i$  be two points in the plane and let  $x_i \in \overline{p_i q_i}$ . For  $0 < t < 1$ , further let  $p_t = (1 - t)p_0 + tp_1$  and  $q_t = (1 - t)q_0 + tq_1$ . Then,  $x_t = (1 - t)x_0 + tx_1 \in \overline{p_t q_t}$  if*

1.  $\overline{p_0 q_0}$  and  $\overline{p_1 q_1}$  are parallel with the same direction, or
2.  $x_0$  cuts  $\overline{p_0 q_0}$  with the same ratio as  $x_1$  cuts  $\overline{p_1 q_1}$



■ **Figure 4** Morphing a segment and a point.

**Proof.** Assume first that  $\overline{p_0q_0}$  and  $\overline{p_1q_1}$  are parallel. If  $\overline{p_0q_0}$  and  $\overline{p_1q_1}$  are collinear, we may assume that they are both contained in the x-axis, that  $p_i, q_i, i = 0, 1$ , are real numbers, and that  $p_0 < q_0$ . Since  $\overline{p_0q_0}$  and  $\overline{p_1q_1}$  have the same direction, this implies that  $p_1 < q_1$ . Since  $x_i, i = 0, 1$ , is a point in  $\overline{p_iq_i}$ , it follows that  $p_i \leq x_i \leq q_i$ . Hence, we get for  $t \in [0, 1]$  that

$$\underbrace{(1-t)p_0 + tp_1}_{p_t} \leq \underbrace{(1-t)x_0 + tx_1}_{x_t} \leq \underbrace{(1-t)q_0 + tq_1}_{q_t}.$$

If  $\overline{p_0q_0}$  and  $\overline{p_1q_1}$  are parallel with the same direction but not collinear, then the polygon  $\langle p_0, q_0, q_1, p_1 \rangle$  is convex. Thus,  $\overline{x_0x_1}$  must intersect  $\overline{p_tq_t}$ , for any  $t$ . Also,  $\overline{p_tq_t}$  and  $x_t$  both lie on the same line  $\ell_t$ . More precisely, let  $d$  be the distance between the lines through segments  $\overline{p_0q_0}$  and  $\overline{p_1q_1}$ . Then,  $\ell_t$  is the line with distance  $td$  from  $\overline{p_0q_0}$ .

Finally, if  $x_0$  cuts  $\overline{p_0q_0}$  with the same ratio  $\lambda$  as  $x_1$  cuts  $\overline{p_1q_1}$ , then  $x_t = (1-t)((1-\lambda)p_0 + \lambda q_0) + t((1-\lambda)p_1 + \lambda q_1) = (1-\lambda)p_t + \lambda q_t \in \overline{p_tq_t}$ . ◀

Lemma 6 implies the following sufficient criterion for a linear morph.

► **Lemma 7.** *Let  $R_0$  and  $R_1$  be two RT-representations of a triangulation  $G$  corresponding to the same Schnyder wood such that the triangles of the outer face pairwise touch in their corners. The pair  $\langle R_0, R_1 \rangle$  defines a linear morph if, for any two adjacent vertices  $u$  and  $v$  such that a corner  $c_i(v)$  of  $v$  touches a side  $s_i(u)$  of  $u$ , where  $c \in \{\nearrow, \leftarrow, \perp\}$  and  $s \in \{\triangleleft, \blacktriangleleft, \blacktriangleright\}$ , one of the following holds:*

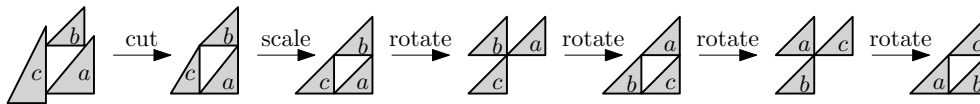
1.  $s_1(u)$  and  $s_2(u)$  are parallel.
2.  $c_1(v)$  cuts  $s_1(u)$  with the same ratio as  $c_2(v)$  cuts  $s_2(u)$ .

By Observation 3, an RT-representation  $R$  of a plane triangulation  $G$  corresponds to a set  $\mathcal{T}_R$  of Schnyder woods that differ from each other by flipping a set of edge disjoint triangles. The *topmost* vertex of  $R$  is the vertex  $v$  of  $G$  maximizing the y-coordinate of  $\triangleleft(v)$ .

► **Lemma 8.** *Let  $R_0$  and  $R_1$  be two RT-representations of the same plane triangulation  $G = (V, E)$  such that  $\langle R_0, R_1 \rangle$  is a linear morph. Then  $\mathcal{T}_{R_0} \cap \mathcal{T}_{R_1} \neq \emptyset$ .*

► **Theorem 9 (Necessary Condition).** *If there is a piecewise linear morph between two RT-representations of a plane triangulation  $G$ , then the corresponding Schnyder woods can be obtained from each other by a sequence of facial flips. In particular the topmost vertex is the same in both representations if  $G$  has more than three vertices.*

**Proof.** Let  $\langle R_1, \dots, R_\ell \rangle$  be a sequence of linear morphs. Lemma 8 implies  $\mathcal{T}_{R_i} \cap \mathcal{T}_{R_{i+1}} \neq \emptyset$  for  $i = 1, \dots, \ell - 1$ . Let  $T_i \in \mathcal{T}_{R_i} \cap \mathcal{T}_{R_{i+1}}$  for  $i = 1, \dots, \ell - 1$ . Then  $T_{i+1}, i = 1, \dots, \ell - 2$  can be obtained from  $T_i$  by a sequence of edge-disjoint facial flips. Hence, the Schnyder wood  $T_{\ell-1}$  of  $R_\ell$  can be obtained from the Schnyder wood  $T_1$  of  $R_1$  by a sequence of facial flips. ◀



■ **Figure 5** Morphing an RT-representation of a triangle to a labeled canonical form: First, cut the extruding parts of the triangles, maintaining the slopes of the diagonal sides. Then, scale the triangles such that the horizontal and vertical sides have length one. Finally, keep rotating the triangles until the topmost vertex is as desired.

## 5 A Morphing Algorithm

In this section, we prove the following theorem.

► **Theorem 10** (Sufficient Condition). *Let  $R_1$  and  $R_2$  be two RT-representations of an  $n$ -vertex plane triangulation  $G$  corresponding to the Schnyder woods  $T_1$  and  $T_2$ , respectively. If  $T_2$  can be obtained from  $T_1$  by a sequence of  $\ell$  facial flips, then there exists a piecewise linear morph between  $R_1$  and  $R_2$  of length  $\mathcal{O}(n + \ell)$ . Such a morph can be computed in  $\mathcal{O}(n(n + \ell))$  time, provided that the respective sequence of  $\ell$  facial flips is given.*

Since there is always a piecewise linear morph between two RT-representations of a plane triangle (see Fig. 5), we will assume that  $G$  has at least four vertices. This implies especially that the topmost vertex, which always coincides with  $X_r$ , is the same in  $R_1$  and  $R_2$ .

In Section 5.1, we introduce our main procedure ADJUST, which moves a triangle in an RT-representation along an incident diagonal and adjusts the remaining triangles so that the result is a linear morph. Repeatedly applying ADJUST, we first morph  $R_1$  to a non-degenerate RT-representation that still corresponds to  $T_1$  (Section 5.3); then, we perform a sequence of linear morphs to realize the  $\ell$  facial flips geometrically (Section 5.2), hence obtaining an RT-representation corresponding to  $T_2$ , which we finally morph to  $R_2$  (Section 5.3).

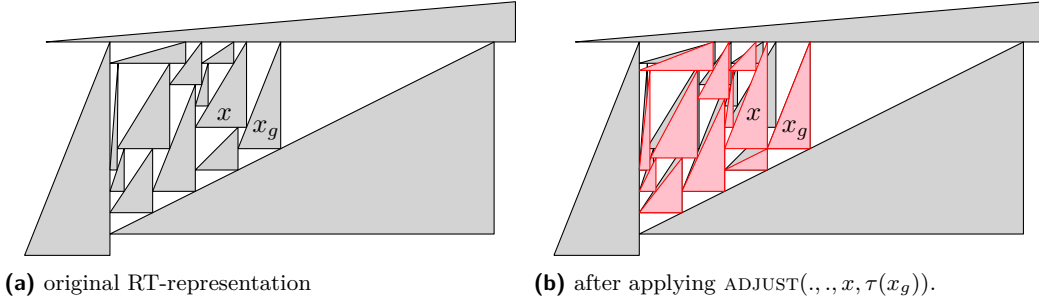
### 5.1 Moving a Triangle Along a Diagonal

Let  $G = (V, E)$  be a plane triangulation and let  $R$  be an RT-representation of  $G$  corresponding to a Schnyder wood  $T$  of  $G$ . Given an inner vertex  $x$  of  $G$  and a real value  $y$  with some properties, ADJUST computes a new RT-representation  $R'$  of  $G$  corresponding to  $T$  in which  $\triangleleft(x)$  has  $y$ -coordinate  $y$  and  $\Delta(x_g)$  remains unchanged, such that  $\langle R, R' \rangle$  is a linear morph.

To achieve this goal, the  $y$ -coordinate of  $\triangleleft(v)$ , for some vertex  $v \neq x$ , may also change; however, the ratio with which  $\triangleleft(v)$  cuts  $\nearrow(v_g)$  does not change, thus satisfying Property 2 of Lemma 7. The  $y$ -coordinates of the horizontal sides are encoded by a new ADT-labeling  $\tau$  of  $G$ , and  $R'$  is the unique RT-representation  $\text{RT}(T, \tau, R_o)$  of  $G$  that is obtained by applying Lemma 5 with input  $G, T, \tau$ , and the representation  $R_o$  of the outer face of  $G$  in  $R$ .

For a vertex  $w \in V$ , we denote by  $\text{top}(w)$  the  $y$ -coordinate of  $\nearrow(w)$ ; recall that, in our construction, we have  $\text{top}(w) = \tau(w_r)$ , if  $w$  is an inner vertex. Also, let  $v_1, \dots, v_\ell$  be the neighbors of  $w$  such that  $\triangleleft(v_1), \dots, \triangleleft(v_\ell)$  appear in this order from  $\swarrow(w)$  to  $\nearrow(w)$  along  $\nearrow(w)$ . For a fixed  $i \in \{1, \dots, \ell\}$ , we say that *moving  $v_i$  to  $y \in \mathbb{R}$  respects the order along  $\nearrow(w)$*  if (i)  $i = 1$  and  $\tau(w) \leq y < \tau(v_2)$  (where equality is only allowed if  $\swarrow(v_1)$  does not lie on  $\triangleleft(w_b)$ ), (ii)  $i = 2, \dots, \ell - 1$  and  $\tau(v_{i-1}) < y < \tau(v_{i+1})$ , or (iii)  $i = \ell$  and  $\tau(v_{\ell-1}) < y \leq \text{top}(w)$  and  $y < \text{top}(v_\ell)$ . Further, for a vertex  $v$ , we consider the ratio  $\lambda(v)$  with which  $\triangleleft(v)$  cuts the incident diagonal side, i.e.,  $\lambda(v) = \frac{\tau(v) - \tau(v_g)}{\text{top}(v_g) - \tau(v_g)}$ , if either  $v$  is an inner vertex or  $v \in \{X_b, X_r\}$ ,  $\triangleleft(v)$  is on  $\nearrow(X_g)$ , and  $v_g := X_g$ .

## 10:10 Morphing Contact Representations of Graphs



■ **Figure 6** Moving  $\Delta(x)$  down along  $\nearrow(x_g)$ .

For the vertex  $x$  and the  $y$ -coordinate  $y$  that are part of the input of  $\text{ADJUST}$ , we assume that moving  $x$  to  $y$  respects the order along  $\nearrow(x_g)$ . Setting  $\tau(x) \leftarrow y$  may have implications on the neighbors of  $x$  of the following type. **A.** For every vertex  $v$  such that  $x = v_g$ , the value of  $\tau(v)$  has to be modified to ensure that the ratio  $\lambda(v)$  with which  $\perp(v)$  cuts  $\nearrow(x)$  is maintained; **B.** for every vertex  $u$  such that  $x = u_r$ , we have to set  $\text{top}(u) = y$  to maintain the contact between  $\Delta(u)$  and  $\Delta(x)$ . Since these modifications may change the diagonal side of  $\Delta(u)$  and  $\Delta(v)$ , they may trigger analogous implications for the neighbors of  $u$  and  $v$ .

Since the  $y$ -coordinate of  $\triangleleft(u)$  is not changed, only a type-A implication may be triggered for the neighbors of  $u$ . Further, the two implications correspond to following either a red or a green edge, respectively, in reverse direction with respect to the one in  $T$ . Hence, the vertices whose triangles may need to be adjusted are those that can be reached from the vertex  $x$  by a reversed directed path in  $T$  using only red and green edges, but no two consecutive red edges; see Fig. 7. Note that, since the green and the red edges have opposite orientation in  $T$  and in  $\text{DAG}_b(T)$ , which is acyclic, this implies that  $\text{ADJUST}$  terminates.

The procedure  $\text{ADJUST}$  (see Fig. 6 for an illustration) first finds all the triangles that may need to be adjusted, by performing a simple graph search from  $x$  following the above described paths of red and green edges. In a second pass, it performs the adjustment of each triangle  $\Delta(w)$ , by modifying  $\tau(w)$  so that  $\lambda(w)$  is maintained. We ensure that the new value of  $\tau(w)$  is computed only after the triangle  $\Delta(w_g)$  has already been adjusted.

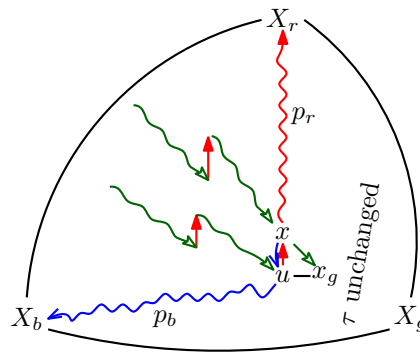
► **Lemma 11.** *Let  $R_1$  be an RT-representation of a plane triangulation  $G = (V, E)$  corresponding to the Schnyder wood  $T$  and let the  $y$ -coordinate of  $\triangleleft_1(v)$  be  $\tau_1(v)$ ,  $v \in V$ . Let  $x \in V$  be an inner vertex and let  $y \in \mathbb{R}$  be such that moving  $x$  to  $y$  respects the order along  $\nearrow(x_g)$ . Let  $\tau_2$  be the output of  $\text{ADJUST}(\tau_1, T, x, y)$ .*

*Then, we have that (i)  $\tau_2(x) = y$ , (ii)  $\lambda(v)$  is maintained for any vertex  $v \neq x$ , (iii)  $\tau_2$  is an ADT-labeling of  $\text{DAG}_r(T)$ , and (iv) the morph between  $R_1$  and  $R_2 = \text{RT}(T, \tau_2, R_o)$  is linear, where  $R_o = \Delta_1(X_b) \cup \Delta_1(X_g) \cup \Delta_1(X_r)$ .*

**Outline of the Proof.** Properties i and ii are clear from the construction. We establish a cycle  $C'$  (see Fig. 7) that encloses all vertices for which  $\tau$  might be changed. Distinguishing the cases  $y < \tau_1(x)$  and  $y > \tau_1(x)$ , Property iii can be shown by induction on a suitable ordering of the edges in  $\text{DAG}_r(T)$ . Since all predecessors of  $x_g$  in  $\text{DAG}_b(T)$  are outside or on  $C'$ , we have that  $\Delta_1(x_g) = \Delta_2(x_g)$ . Now, by Lemma 7,  $\langle R_1, R_2 \rangle$  is a linear morph. ◀

## 5.2 A Flipping Algorithm

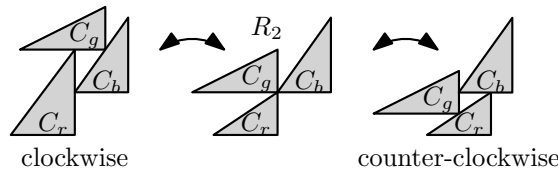
Recall that, given a Schnyder wood  $T$  and an oriented cycle  $C$  in  $T$ , the Schnyder wood  $T_C$  is obtained from  $T$  by flipping  $C$ . In the following theorem we show how to realize this flip geometrically with two linear morphs in the case in which  $C$  is a facial cycle.



**Figure 7** Let  $u$  be the neighbor of  $x$  following  $x_g$  in clockwise order. Then, the cycle  $C'$  composed of the blue  $u$ - $X_b$ -path  $p_b$ , the edge  $\{u, x\}$ , the red  $x$ - $X_r$ -path  $p_r$ , and the edge  $\{X_r, X_b\}$  encloses all vertices for which  $\tau$  is changed. Vertex  $x$  is the only vertex on  $C'$  for which  $\tau$  is changed.

**Theorem 12.** Let  $R_1$  be a non-degenerate RT-representation of a plane triangulation  $G$  corresponding to a Schnyder wood  $T$ . Let  $C$  be an oriented facial cycle in  $T$ . We can construct a sequence of two linear morphs  $\langle R_1, R_2, R_3 \rangle$  such that  $R_3$  is a non-degenerate RT-representation of  $G$  corresponding to a Schnyder wood  $T_C$ .

**Proof.** For the oriented facial cycle  $C$ , let  $C_r$ ,  $C_g$ , and  $C_b$  be the vertices with outgoing red, green, and blue edge, respectively, in  $C$ . In order to flip  $C$ , we move  $C_g$  along the respective incident diagonal sides as sketched in the following figure.



More precisely, let  $\tau_1$  be the y-coordinates of the horizontal sides in  $R_1$ . We first compute  $\tau_2 \leftarrow \text{ADJUST}(\tau_1, T, C_g, \tau_1(C_b))$ . If  $C$  is clockwise oriented, we then compute

$$\tau_3 \leftarrow \text{ADJUST}(\tau_2, T_C, C_g, (\tau_2(C_g) + \max\{\tau_2(u); u = C_r \text{ or } u_g = C_r\})/2).$$

If  $C$  is counter-clockwise oriented, we proceed as follows.

$$\tau_3 \leftarrow \text{ADJUST}(\tau_2, T_C, C_g, (\tau_2(C_g) + \min\{\tau_2(u); u = (C_b)_r \text{ or } u_g = C_b\})/2).$$

In each case the new y-coordinates  $y$  for  $C_g$  are chosen such that moving  $C_g$  to  $y$  respects the order along the respective incident diagonal. Thus,  $\text{ADJUST}$  can be applied. Also,  $\tau_2$  is an ADT-labeling of both,  $\text{DAG}_r(T)$  and  $\text{DAG}_r(T_C)$ , and  $\tau_3$  is an ADT-labeling of  $\text{DAG}_r(T_C)$ . Let  $R_2 = \text{RT}(T, \tau_2, R_o) = \text{RT}(T_C, \tau_2, R_o)$  and let  $R_3 = \text{RT}(T_C, \tau_3, R_o)$ . Since  $\tau_2$  and  $\tau_3$  are produced by  $\text{ADJUST}$ , by Lemma 11, both  $\langle R_1, R_2 \rangle$  and  $\langle R_2, R_3 \rangle$  are linear morphs. ◀

### 5.3 Morphing Representations with the same Schnyder Wood

In this section, we consider RT-representations corresponding to the same Schnyder Wood.

**Theorem 13.** Let  $R_1$  and  $R_2$  be two RT-representations of an  $n$ -vertex plane triangulation corresponding to the same Schnyder wood  $T$ . Then, there is a piecewise linear morph between  $R_1$  and  $R_2$  of length at most  $2n$ .



## 10:12 Morphing Contact Representations of Graphs

The idea is to first transform the outer face to a canonical form, and then to move one vertex  $v$  per step to a new  $y$ -coordinate  $y$  such that the ratio  $\lambda(v)$  is set to how it should be in  $R_2$ . The order in which we process the vertices is such that ADJUST can be applied to the vertex  $v$  and the  $y$ -coordinate  $y$ . Recall that ADJUST does not alter the ratio  $\lambda$ , except for the currently processed vertex  $v$ . The following lemma can be proven by induction on  $n$ .

► **Lemma 14.** *Let  $P = \{p_1 < \dots < p_n\}$  and  $Q = \{q_1 < \dots < q_n\}$  be two sets of  $n$  reals each. If  $P \neq Q$  then there is an  $i$  such that  $p_i \neq q_i$  and  $P$  has no element between  $p_i$  and  $q_i$ .*

► **Corollary 15.** *Let  $P$  and  $Q$  each be a set of  $n$  points on a segment  $s$ . We can move  $P$  to  $Q$  in  $n$  steps by moving one point per step and by maintaining the ordering of the points on  $s$ .*

**Proof of Theorem 13.** Let  $\tau'$  be a topological ordering of the inner vertices of  $\text{DAG}_r(T)$ . We extend  $\tau'$  to an ADT-labeling of  $\text{DAG}_r(T)$  by setting  $\tau'(X_b) = 0 = \tau'(X_g)$ , and  $\tau'(X_r) = n - 2$ . With a sequence of at most  $n$  linear morphs we transform  $R_i$ ,  $i = 1, 2$ , into an RT-representation  $R' = \text{RT}(T, \tau', R_o)$ , where  $R_o$  has the following *canonical form*:  $\lrcorner(X_b) = \llcorner(X_g) = (0, 0)$ ,  $\nearrow(X_b) = \llcorner(X_r) = (0, n - 2)$ ,  $\nearrow(X_g) = \lrcorner(X_r) = (n - 2, n - 2)$ , and the lengths of  $\lrcorner(X_r)$  and  $\llcorner(X_b)$  are one. In the first morph, we cut the extruding parts of the outer triangles. In the second morph, we independently scale the  $x$ - and  $y$ -coordinates of the corners and translate the drawing, to fit the corners as indicated. In a third step, we adjust the lengths of  $\lrcorner(X_r)$  and  $\llcorner(X_b)$ . In the first morph the slope of no side is changed, in the second morph no ratio is changed, and in the third morph there are only four sides that are changed, which are not incident to any other triangle. Thus, the three morphs are linear. Let the resulting RT-representation be  $R'_i$ .

We now process the vertices in a reversed topological ordering on  $\text{DAG}_b(T)$ . We process a vertex  $w$  as follows. Let  $\tau$  be the current  $y$ -coordinates of the horizontal sides. Let  $\mathcal{G}(w) = \{v \in V; w = v_g\}$  and let  $P = \{\tau(v); v \in \mathcal{G}(w)\}$ . For  $v \in \mathcal{G}(w)$  let  $y(v)$  be such that

$$\frac{y(v) - \tau(w)}{\tau(w_r) - \tau(w)} = \frac{\tau'(v) - \tau'(w)}{\tau'(w_r) - \tau'(w)},$$

i.e., placing  $\llcorner(v)$ ,  $v \in \mathcal{G}(w)$  on the  $y$ -coordinate  $y(v)$  cuts  $\nearrow(w)$  in the the same ratio as in  $R'$ . Let  $Q = \{y(v); v \in \mathcal{G}(w)\}$ . By the above corollary, we can order  $\mathcal{G}(w) = \{v_1, \dots, v_k\}$  such that replacing in the  $i$ th step  $\tau(v_i)$  by  $y(v_i)$  maintains the ordering of  $\{\tau(v); v \in \mathcal{G}(w)\}$ . Since  $\tau'$  is a topological ordering, we will not move  $\lrcorner(v_i)$  to an end vertex of  $\nearrow(w)$ . For  $i = 1, \dots, k$  we now call  $\tau \leftarrow \text{ADJUST}(\tau, T, v_i, y(v_i))$ . This yields one linear morphing step.

After processing all vertices  $w$  in a reversed topological ordering of  $\text{DAG}_b(T)$  and all vertices in  $\mathcal{G}(w)$  in the order given above, we have obtained an RT-representation  $R$  in which any right corner cuts its incident diagonal in the same ratio as in  $R'$ . Since the outer face is fixed, this implies that  $R = R'$ . Observe that  $\mathcal{G}(w)$ ,  $w \in V$ , is a partition of the set of inner vertices. Hence, we get at most one morphing step for each of the  $n - 3$  inner vertices. ◀

Combining the results of Sections 5.1 to 5.3 yields the main result of the section.

**Proof of Theorem 10.** First, we transform  $R_1$  into a non-degenerate RT-representation  $R$  with Schnyder wood  $T_1$  and a canonical representation of the outer face in  $\mathcal{O}(n)$  linear morphing steps, by Theorem 13. Then, we perform the  $\ell$  facial flips as described in the proof of Theorem 12, using two linear morphs for each flip. This yields an RT-representation  $R'$  with Schnyder wood  $T_2$ . Finally, we transform  $R'$  into  $R_2$  in  $\mathcal{O}(n)$  linear morphing steps, by Theorem 13. This yields a total of  $\mathcal{O}(n + \ell)$  linear morphs. Each linear morph can be computed by one application of ADJUST, which runs in linear time. ◀



**6 A Decision Algorithm**

It follows from Theorem 9 and Theorem 10 that there is a piecewise linear morph between two RT-representations of a plane triangulation if and only if the respective Schnyder woods can be obtained from each other by flipping faces only. Note that this condition is always satisfied if the triangulation is 4-connected and the topmost vertex is the same in both RT-representations. On the other hand, if the graph contains separating triangles, we have to decide whether there is such a sequence of facial flips. We will show that this can be decided efficiently and that, in the positive case, there exists one sequence whose length is at most quadratic in the number of vertices. This establishes our final result.

► **Theorem 16.** *Let  $R_1$  and  $R_2$  be two RT-representations of an  $n$ -vertex plane triangulation. We can decide in  $\mathcal{O}(n^2)$  time whether there is a piecewise linear morph between  $R_1$  and  $R_2$  and, if so, a morph with  $\mathcal{O}(n^2)$  linear morphing steps can be computed in  $\mathcal{O}(n^3)$  time.*

Since there is a one-to-one correspondence between Schnyder woods of a plane triangulation and its 3-orientations, we will omit the colors in the following. A careful reading of Brehm [15] and Felsner [25] reveals the subsequent properties of 3-orientations. The set of 3-orientations of a triangulation forms a distributive lattice with respect to the following ordering.  $T_1 \leq T_2$  if and only if  $T_1$  can be obtained from  $T_2$  by a sequence of flips on some counter-clockwise triangles. The minimum element is the unique 3-orientation without counter-clockwise cycles. Moreover, given a 3-orientation  $T$  and a triangle  $t$ , the number of occurrences of  $t$  in any flip-sequence between  $T$  and the minimum 3-orientation is the same – provided that the flip sequence contains only counter-clockwise triangles. Let this number be the potential  $\pi_T(t)$ .

Observe that  $\pi_T$  is distinct for distinct  $T$ . Moreover,  $\min(\pi_{T_1}(t), \pi_{T_2}(t))$ ,  $t$  triangle, is the potential of the *meet*  $T_1 \wedge T_2$  (i.e., the infimum) of two 3-orientations  $T_1$  and  $T_2$ , while  $\max(\pi_{T_1}(t), \pi_{T_2}(t))$ ,  $t$  triangle, is the potential of the *join*  $T_1 \vee T_2$  (i.e., the supremum) of  $T_1$  and  $T_2$ . The potential  $\pi_T$  can be computed in quadratic time for a fixed 3-orientation  $T$  of an  $n$ -vertex triangulation: At most  $\mathcal{O}(n^2)$  flips have to be performed in order to reach the minimum 3-orientation. With a linear-time preprocessing, we can store all initial counter-clockwise triangles in a list. After each flip, the list can be updated in constant time.

► **Lemma 17.** *Let  $T_1$  and  $T_2$  be two 3-orientations of an  $n$ -vertex triangulation.  $T_1$  can be obtained from  $T_2$  by a sequence of facial flips if and only if  $\pi_{T_1}(t) - \pi_{T_2}(t) = 0$  for all separating triangles  $t$ . Moreover, if  $T_1$  can be obtained from  $T_2$  by a sequence of facial flips, then it can be obtained by  $\mathcal{O}(n^2)$  facial flips.*

**Proof.** Observe that going from  $T_1$  to the meet  $T_1 \wedge T_2$  involves

$$\pi_{T_1}(t) - \min(\pi_{T_1}(t), \pi_{T_2}(t)) \in \{0, \pi_{T_1}(t) - \pi_{T_2}(t)\}$$

counter-clockwise flips on triangle  $t$ , and going from the meet  $T_1 \wedge T_2$  to  $T_2$  involves

$$\pi_{T_2}(t) - \min(\pi_{T_1}(t), \pi_{T_2}(t)) \in \{0, \pi_{T_2}(t) - \pi_{T_1}(t)\}$$

clockwise flips on triangle  $t$ . Thus, if  $\pi_{T_1}(t) - \pi_{T_2}(t) = 0$  for all separating triangles  $t$ , then no flip must be performed on a separating triangle. Then, the total number of flips is bounded by  $\sum_{t \text{ face}} (\pi_{T_1}(t) + \pi_{T_2}(t)) \in \mathcal{O}(n^2)$ .

Assume now that there is a sequence  $T_1 = T'_0, T'_1, \dots, T'_\ell, T'_{\ell+1} = T_2$  of 3-orientations such that  $T'_{i+1}$ ,  $i = 0, \dots, \ell$ , is obtained from  $T'_i$  by a (clockwise or counter-clockwise) facial flip. We show by induction on  $\ell$  that  $\pi_{T_1}(t) - \pi_{T_2}(t) = 0$  for all separating triangles  $t$ . If  $\ell = 0$ , let

$t_0$  be the triangle that has to be flipped in order to go from  $T_1$  to  $T_2$ . Then,  $t_0$  is a face and  $\pi_{T_1}(t) - \pi_{T_2}(t) = 0$  for  $t \neq t_0$ . Assume now that  $\ell \geq 1$ . Let  $t$  be a separating triangle. Then

$$\pi_{T_1}(t) - \pi_{T_2}(t) = \underbrace{\pi_{T_1}(t) - \pi_{T'_\ell}(t)}_{=0 \text{ by IH}} + \underbrace{\pi_{T'_\ell}(t) - \pi_{T_2}(t)}_{=0 \text{ by IH}} = 0. \quad \blacktriangleleft$$

## 7 Conclusions and Open Problems

We have studied piecewise linear morphs between RT-representations of plane triangulations, and shown that when such a morph exists, there is one of length  $\mathcal{O}(n^2)$ . It would be interesting to explore lower bounds on this length. Observe that the minimum length of a flip-sequence containing only facial cycles does not immediately imply such bound, since some flips could be parallelized. Additionally, bounds on the resolution throughout our morphs would be worth investigating; however, it is unclear whether the “ratio fixing” we use would allow nice bounds. For this, it may help to return to integer y-coordinates between any two flips; however, this would result in a cubic number of linear morphing steps. A major open direction is whether our results can be lifted to general plane graphs, e.g., through the use of compatible triangulations. Note that such a compatible triangulation would need to be formed while preserving the conditions for the existence of a linear morph, i.e., without introducing the need to flip a separating triangle.

Finally, beyond the context of RT-representations, many other families of geometric objects could be considered. For example, morphing degenerate contact representations of line segments generalizes planar morphing, by treating contact points as vertices.

---

### References

- 1 Soroush Alamdari, Patrizio Angelini, Fidel Barrera-Cruz, Timothy M. Chan, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Penny Haxell, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. How to Morph Planar Graph Drawings. *SIAM Journal on Computing*, 46(2):824–852, 2017.
- 2 Soroush Alamdari, Patrizio Angelini, Timothy M. Chan, Giuseppe Di Battista, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, Vincenzo Roselli, Sahil Singla, and Bryan T. Wilkinson. Morphing Planar Graph Drawings with a Polynomial Number of Steps. In Sanjeev Khanna, editor, *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2013)*, pages 1656–1667. SIAM, 2013.
- 3 Helmut Alt and Leonidas J. Guibas. Chapter 3 - Discrete Geometric Shapes: Matching, Interpolation, and Approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. North-Holland, Amsterdam, 2000.
- 4 E. M. Andreev. On convex polyhedra in Lobachevskij spaces. *Math. USSR, Sb.*, 10:413–440, 1971.
- 5 Patrizio Angelini, Steven Chaplick, Sabine Cornelsen, Giordano Da Lozzo, and Vincenzo Roselli. Morphing Contact Representations of Graphs. *CoRR*, abs/1903.07595, 2019. [arXiv:1903.07595](https://arxiv.org/abs/1903.07595).
- 6 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing Planar Graph Drawings Optimally. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming - 41st International Colloquium (ICALP 2014)*, volume 8572 of *LNCS*, pages 126–137. Springer, 2014.
- 7 Patrizio Angelini, Giordano Da Lozzo, Fabrizio Frati, Anna Lubiw, Maurizio Patrignani, and Vincenzo Roselli. Optimal Morphs of Convex Drawings. In Lars Arge and János Pach, editors,

- Proceedings of the 31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *LIPICs*, pages 126–140. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.
- 8 Patrizio Angelini, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Morphing Planar Graph Drawings Efficiently. In Stephen K. Wismath and Alexander Wolff, editors, *Proceedings of the 21st International Symposium on Graph Drawing (GD 2013)*, volume 8242 of *LNCS*, pages 49–60. Springer, 2013.
  - 9 Douglas N. Arnold and Jonathan Rogness. Möbius Transformations Revealed. *Notices of the AMS*, 55(10), 2008.
  - 10 Boris Aronov, Raimund Seidel, and Diane Souvaine. On compatible triangulations of simple polygons. *Computational Geometry*, 3(1):27–35, 1993.
  - 11 Elena Arseneva, Prosenjit Bose, Pilar Cano, Anthony D’Angelo, Vida Dujmovic, Fabrizio Frati, Stefan Langerman, and Alessandra Tappini. Pole Dancing: 3D Morphs for Tree Drawings. In Therese Biedl and Andreas Kerren, editors, *Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD 2018)*, volume 11282 of *LNCS*, pages 371–384. Springer, 2018.
  - 12 Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. Morphing Schnyder Drawings of Planar Triangulations. *Discrete and Computational Geometry*, pages 1–24, 2018.
  - 13 Therese Biedl, Anna Lubiw, Mark Petrick, and Michael Spriggs. Morphing Orthogonal Planar Graph Drawings. *ACM Transactions on Algorithms*, 9(4):29:1–29:24, 2013.
  - 14 Clinton Bowen, Stephane Durocher, Maarten Löffler, Anika Rounds, André Schulz, and Csaba D. Tóth. Realization of Simply Connected Polygonal Linkages and Recognition of Unit Disk Contact Trees. In Emilio Di Giacomo and Anna Lubiw, editors, *Proceedings of the 23rd International Symposium on Graph Drawing (GD 2015)*, volume 9411 of *LNCS*, pages 447–459. Springer, 2015.
  - 15 Enno Brehm. 3-Orientations and Schnyder 3-Tree-Decompositions. Master’s thesis, Freie Universität Berlin, FB Mathematik und Informatik, 2000. Diploma thesis.
  - 16 Graham R. Brightwell and Edward R. Scheinerman. Representations of Planar Graphs. *SIAM Journal on Discrete Mathematics*, 6(2):214–229, 1993.
  - 17 S. S. Cairns. Deformations of Plane Rectilinear Complexes. *The American Mathematical Monthly*, 51(5):247–252, 1944.
  - 18 Steven Chaplick, Stephen G. Kobourov, and Torsten Ueckerdt. Equilateral L-Contact Graphs. In Andreas Brandstädt, Klaus Jansen, and Rüdiger Reischuk, editors, *Proceedings of the 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013)*, volume 8165 of *LNCS*, pages 139–151. Springer, 2013.
  - 19 Robert Connelly, Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Stefan Langerman, Joseph S. B. Mitchell, Ares Ribó, and Günter Rote. Locked and Unlocked Chains of Planar Shapes. *Discrete & Computational Geometry*, 44(2):439–462, 2010.
  - 20 Giordano Da Lozzo, William E. Devanny, David Eppstein, and Timothy Johnson. Square-Contact Representations of Partial 2-Trees and Triconnected Simply-Nested Graphs. In Yoshio Okamoto and Takeshi Tokuyama, editors, *28th International Symposium on Algorithms and Computation, (ISAAC 2017)*, volume 92 of *LIPICs*, pages 24:1–24:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
  - 21 Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Upward Planar Morphs. In Therese Biedl and Andreas Kerren, editors, *Proceedings of the 26th International Symposium on Graph Drawing and Network Visualization (GD 2018)*, volume 11282 of *LNCS*, pages 92–105. Springer, 2018.
  - 22 Hubert de Fraysseix and Patrice Ossona de Mendez. Representations by Contact and Intersection of Segments. *Algorithmica*, 47(4):453–463, 2007.
  - 23 Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl. On Triangle Contact Graphs. *Combinatorics, Probability, and Computing*, 3(2):233–246, 1994.
  - 24 Erik D. Demaine and Joseph O’Rourke. *Geometric folding algorithms - linkages, origami, polyhedra*. Cambridge University Press, 2007.

- 25 Stefan Felsner. Lattice Structures from Planar Graphs. *The Electronic Journal of Combinatorics*, 11(1), 2004.
- 26 Stefan Felsner and Mathew C. Francis. Contact Representations of Planar Graphs with Cubes. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG 2011)*, pages 315–320. ACM, 2011.
- 27 Stefan Felsner and Günter Rote. On Primal-Dual Circle Representations. In Jeremy T. Fineman and Michael Mitzenmacher, editors, *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69 of *OpenAccess Series in Informatics (OASICS)*, pages 8:1–8:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASICS.SOSA.2019.8.
- 28 Stefan Felsner, Hendrik Schrezenmaier, and Raphael Steiner. Equiangular Polygon Contact Representations. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Proceedings of the 44th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2018)*, volume 11159 of *LNCS*, pages 203–215. Springer, 2018.
- 29 Michael S. Floater and Craig Gotsman. How to morph tilings injectively. *Journal of Computational and Applied Mathematics*, 101(1):117–129, 1999.
- 30 Daniel Gonçalves, Benjamin Lévêque, and Alexandre Pinlou. Triangle Contact Representations and Duality. *Discrete & Computational Geometry*, 48(1):239–254, 2012.
- 31 Craig Gotsman and Vitaly Surazhsky. Guaranteed intersection-free polygon morphing. *Computers & Graphics*, 25(1):67–75, 2001.
- 32 Stephen Kobourov, Torsten Ueckerdt, and Kevin Verbeek. Combinatorial and Geometric Properties of Planar Laman Graphs. In *Proceedings of the 24th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 1668–1678. SIAM, 2013.
- 33 Stephen G. Kobourov and Matthew Landis. Morphing Planar Graphs in Spherical Space. *Journal of Graph Algorithms and Applications*, 12(1):113–127, 2008.
- 34 Paul Koebe. Kontaktprobleme der Konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physikalische Klasse*, 88:141–164, 1936.
- 35 G. Laman. On Graphs and Rigidity of Plane Skeletal Structures. *Journal of Engineering Mathematics*, 4(4):331–340, 1970.
- 36 Walter Schnyder. Embedding Planar Graphs on the Grid. In *Proceedings of the 1st ACM-SIAM Symposium on Discrete Algorithms (SODA '90)*, pages 138–148, 1990.
- 37 Oded Schramm. *Combinatorially Prescribed Packings and Applications to Conformal and Quasiconformal Maps*. PhD thesis, Princeton University, 1990. Modified version: [arXiv:0709.0710v1](https://arxiv.org/abs/0709.0710v1).
- 38 Hendrik Schrezenmaier. Homothetic Triangle Contact Representations. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Proceedings of the 43rd International Workshop on Graph Theoretic Concepts in Computer Science (WG 2017)*, number 10520 in *LNCS*, pages 425–437, 2017.
- 39 Vitaly Surazhsky and Craig Gotsman. Controllable morphing of compatible planar triangulations. *ACM Transactions on Graphics*, 20(4):203–231, 2001.
- 40 Carsten Thomassen. Deformations of Plane Graphs. *Journal of Combinatorial Theory, Series B*, 34(3):244–257, 1983.
- 41 Carsten Thomassen. Interval representations of planar graphs. *Journal of Combinatorial Theory, Series B*, 40(1):9–20, 1986.
- 42 W. T. Tutte. How to Draw a Graph. *Proceedings of the London Mathematical Society*, s3-13(1):743–767, 1963.
- 43 Arthur van Goethem and Kevin Verbeek. Optimal Morphs of Planar Orthogonal Drawings. In Bettina Speckmann and Csaba D. Tóth, editors, *Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPICs*, pages 42:1–42:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

# When Convexity Helps Collapsing Complexes

**Dominique Attali**

Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, Grenoble, France  
Dominique.Attali@grenoble-inp.fr

**André Lieutier**

Dassault systèmes, Aix-en-Provence, France  
andre.lieutier@3ds.com

**David Salinas**

Amazon research, Berlin, Germany  
dsalina@amazon.com

---

## Abstract

This paper illustrates how convexity hypotheses help collapsing simplicial complexes. We first consider a collection of compact convex sets and show that the nerve of the collection is collapsible whenever the union of sets in the collection is convex. We apply this result to prove that the Delaunay complex of a finite point set is collapsible. We then consider a convex domain defined as the convex hull of a finite point set. We show that if the point set samples sufficiently densely the domain, then both the Čech complex and the Rips complex of the point set are collapsible for a well-chosen scale parameter. A key ingredient in our proofs consists in building a filtration by sweeping space with a growing sphere whose center has been fixed and studying events occurring through the filtration. Since the filtration mimics the sublevel sets of a Morse function with a single critical point, we anticipate this work to lay the foundations for a non-smooth, discrete Morse Theory.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

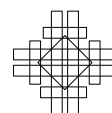
**Keywords and phrases** collapsibility, convexity, collection of compact convex sets, nerve, filtration, Delaunay complex, Čech complex, Rips complex

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.11

## 1 Introduction

**Contractibility and collapsibility.** In the realm of point set topology in Euclidean spaces, a set is said to be *contractible* if it has the homotopy type of a point. Examples of contractible sets are bounded convex sets which can each be continuously retracted to a point. Consider a finite collection of convex sets  $\mathcal{C}$  and the domain defined by their union  $\bigcup \mathcal{C}$ . We know that each convex set in the collection is contractible. Moreover, any non-empty intersection of two or more convex sets being itself convex is again contractible. In this situation, the topology of the domain  $\bigcup \mathcal{C}$  is determined by the pattern in which convex sets in  $\mathcal{C}$  intersect. This is asserted by the Nerve Lemma [6] also known as Leray's Theorem. Recall that the *nerve* of a collection of sets is the abstract simplicial complex whose vertices correspond to sets in the collection and whose simplices correspond to sub-collections with non-empty intersection. The nerve provides a way to record the intersection pattern of sets in a collection. The Nerve Lemma implies that the nerve inherits the homotopy type of  $\bigcup \mathcal{C}$ . In particular, if  $\bigcup \mathcal{C}$  is convex, then it is contractible and so is the nerve.

*Simplicial collapses* are unitary operations on simplicial complexes that preserve the homotopy type while removing a few simplices. A simplicial complex is said to be *collapsible* if it can be reduced to a single vertex by a sequence of collapses. Collapsibility can be interpreted as a combinatorial version of contractibility. Indeed, each sequence of collapses



can be interpreted as a combinatorial analog of a deformation retract. Whereas contractibility involves continuous processes, the notion of collapsibility, thanks to its combinatorial and finitary nature, can be handled directly in a computational context [4, 2, 3].

If a simplicial complex is collapsible, then it is also contractible. However, the converse does not hold – the triangulated Bing’s house is a famous example of simplicial complex which is contractible without being collapsible [9]. A natural question to ask is: Under which condition a simplicial complex is collapsible? This question has been studied in different context. For instance, it is known that all triangulation of the 2-dimensional ball are collapsible. Whether linear subdivisions of convex  $d$ -balls, or at least their subdivision, are collapsible has been a long-standing open problem [5].

**Contributions.** In this paper, we establish collapsibility of certain simplicial complexes. First, we consider a finite collection of compact convex sets whose union is convex and prove that the nerve of the collection is collapsible. This is a stronger result than what we obtain when applying the Nerve Lemma which only entails contractibility of the nerve. Then, we focus on two simplicial complexes built upon a finite point set  $S$ . The first one is the Čech complex  $\mathcal{C}(S, r)$  of  $S$  with scale parameter  $r$  defined as the nerve of the collection of balls of radius  $r$  centered at  $S$ . The second one is the Vietoris-Rips  $\mathcal{R}(S, r)$  which is the largest simplicial complex sharing with  $\mathcal{C}(S, r)$  the same set of vertices and edges. In other words, a simplex belongs to  $\mathcal{R}(S, r)$  if and only if all its vertices and edges belong to  $\mathcal{C}(S, r)$ . Such a simplicial complex which enjoys the property to be completely determined by the set of its vertices and edges is called a *flag completion*. We study the situation in which the point set  $S$  samples sufficiently densely a given convex domain. As a second result, we obtain that, in this particular case, both the Čech complex and the Vietoris-Rips complex of  $S$  are collapsible for a well-chosen scale parameter. Furthermore, when reducing the Vietoris-Rips complex to a point by a sequence of collapses, we can do it in such a way that we preserve at all time the property of the complex to be a flag completion. This opens the possibility to compute the sequence of collapses by maintaining the graph formed by the vertices and edges, thus avoiding the complexity, possibly exponential in the dimension, required by an extensive complex representation; see [1] for an example of a data structure optimized for “almost-flag” completions.

## 2 Preliminaries

In this section we review the necessary background and explain some of our terms.

### 2.1 Euclidean space, distances and convex sets

$\mathbb{R}^d$  denotes the Euclidean space and  $\|x - y\|$  the Euclidean distance between two points  $x$  and  $y$  of  $\mathbb{R}^d$ . Given a point  $o \in \mathbb{R}^d$  and a subset  $X \subseteq \mathbb{R}^d$ , we write  $d(o, X)$  for the infimum of Euclidean distances between  $o$  and points in  $X$ . By convention, we set  $d(o, X) = +\infty$  whenever  $X$  is empty. We write  $\partial X$  for the boundary of  $X$ . The closed ball with center  $x$  and radius  $r$  is denoted by  $B(x, r)$ . The dilation of  $X$  by a ball of radius  $r$  centered at the origin is  $X^{\oplus r} = \bigcup_{x \in X} B(x, r)$  and referred to as the  $r$ -offset of  $X$ . If  $X$  is either compact or convex, so is  $X^{\oplus r}$ . It is easy to check that  $(X \cup Y)^{\oplus r} = X^{\oplus r} \cup Y^{\oplus r}$ . In this paper, we will use many times the following fact.

► **Remark.** Let  $o \in \mathbb{R}^d$  and  $X \subseteq \mathbb{R}^d$ . If  $X$  is a non-empty compact convex set, then there is a unique point  $x \in X$  such that  $d(o, X) = \|o - x\|$ .



## 2.2 Abstract simplicial complexes

An (*abstract*) *simplex* is any finite non-empty set. The *dimension* of a simplex  $\sigma$  is one less than its cardinality. A *k-simplex* designates a simplex of dimension  $k$ . If  $\tau \subseteq \sigma$  is a non-empty subset, we call  $\tau$  a *face* of  $\sigma$  and  $\sigma$  a *coface* of  $\tau$ . If in addition  $\tau \neq \sigma$ , we say that  $\tau$  is a *proper face* and  $\sigma$  is a *proper coface*. Given a set of simplices  $\Delta$  and a simplex  $\sigma \in \Delta$ , we say that  $\sigma$  is *inclusion-maximal* in  $\Delta$  if it has no proper coface in  $\Delta$ . Similarly, we say that  $\sigma$  is *inclusion-minimal* if it has no proper face in  $\Delta$ . An *abstract simplicial complex* is a collection of simplices  $K$ , that contains, with every simplex, the faces of that simplex. The vertex set of the abstract simplicial complex  $K$  is the union of its elements,  $\text{Vert}(K) = \bigcup_{\sigma \in K} \sigma$ . A *subcomplex* of  $K$  is a simplicial complex  $L \subseteq K$ . The *star* of  $\sigma$  in  $K$ , denoted  $\text{St}_K(\sigma)$ , is the set of cofaces of  $\sigma$ . The *link* of  $\sigma$  in  $K$ , denoted  $\text{Lk}_K(\sigma)$ , is the set of simplices  $\tau$  in  $K$  such that  $\tau \cup \sigma \in K$  and  $\tau \cap \sigma = \emptyset$ . It is a subcomplex of  $K$ . Another particular subcomplex is the *i-skeleton* consisting of all simplices of dimension  $i$  or less. We call the simplicial complex formed by a simplex and all its faces the *closure* of that simplex. The closure of a simplex is an example of cone. A *cone* is a simplicial complex  $L$  which contains a vertex  $o$  such that the following holds:  $\sigma \in L \implies \sigma \cup \{o\} \in L$ .

## 2.3 Collapses

Let  $\pi : \text{Vert}(K) \rightarrow \mathbb{R}^n$  be an injective map that sends the  $n$  vertices of  $K$  to  $n$  affinely independent points of  $\mathbb{R}^n$ , such as for instance the  $n$  vectors of the standard basis of  $\mathbb{R}^n$ . Let  $\text{Hull}(X)$  denote the convex hull of  $X \subseteq \mathbb{R}^n$ . The *underlying space* of  $K$  is the point set  $|K| = \bigcup_{\sigma \in K} \text{Hull}(\pi(\sigma))$  and is defined up to a homeomorphism. We shall say that an operation preserves the homotopy-type of  $K$  if the result is a simplicial complex  $K'$  whose underlying space is homotopy equivalent to that of  $K$ . We are interested in simplifying a simplicial complex through a sequence of homotopy-preserving operations.

Consider the operation that removes from  $K$  the set of simplices  $\Delta = \text{St}_K(\sigma)$ . This operation is known to preserve the homotopy-type in the following three cases:

1.  $\Delta = \{\sigma, \tau\}$  with  $\sigma \neq \tau$ . This case can also be characterized by the fact that the link of  $\sigma$  is reduced to a singleton. The operation is called an (*elementary*) *collapse*.
2.  $\Delta = \{\eta \mid \sigma \subseteq \eta \subseteq \tau\}$  with  $\sigma \neq \tau$ . This case can also be characterized by the fact that the link of  $\sigma$  is the *closure* of a simplex. The operation is called a (*classical*) *collapse*.
3. The link of  $\sigma$  is a cone. The operation is called an (*extended*) *collapse*.

Both classical and extended collapses can be expressed as compositions of elementary collapses. A simplicial complex is said to be *collapsible* if it can be reduced to a single vertex by a finite sequence of collapses (either elementary, classical or extended).

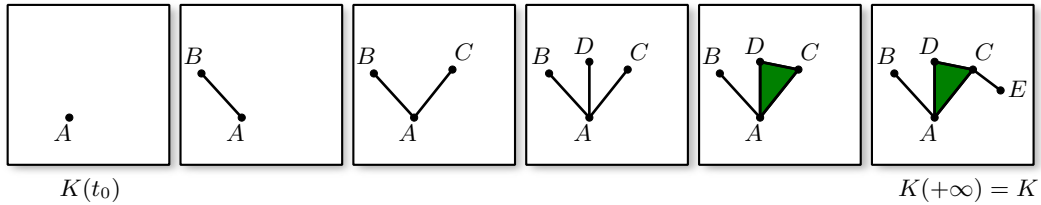
## 2.4 Filtrations

In the next two sections, we establish collapsibility of certain simplicial complexes using the following strategy. We associate to simplicial complex  $K$  a filtration  $\{K(t)\}_{t \in \mathbb{R}}$  which is a nested one-parameter family of simplicial complexes such that  $K(-\infty) = \emptyset$  and  $K(+\infty) = K$ . The filtration induces a strict order on simplices of  $K$  defined by  $\eta \prec_K \nu \iff \{\eta \in K(t_i) \text{ and } \nu \in K(t_j) \setminus K(t_i) \text{ for some } t_i < t_j\}$ . Simply put, as time  $t$  goes by, if a simplex  $\eta$  shows up in  $K(t)$  strictly before another simplex  $\nu$ , then  $\eta \prec_K \nu$ . As we continuously increases the parameter  $t$  from  $-\infty$  to  $+\infty$ , the simplicial complex  $K(t)$  changes only at finitely many values of  $t$  for which the set of simplices  $\Delta(t) = K(t) \setminus \lim_{u \rightarrow t^-} K(u)$  is non-empty, where  $\lim_{u \rightarrow t^-} K(u)$  designates the limit of  $K(u)$  as  $u$  approaches  $t$  from below. Let  $t_0$  be the first time at which a non-empty simplicial complex appears in the filtration. In other words,  $K(t_0)$

## 11:4 When Convexity Helps Collapsing Complexes

is the smallest non-empty simplicial complex in the filtration. To prove that  $K$  is collapsible, it suffices to show that: (1)  $K(t_0)$  is collapsible; (2) the operation that removes  $\Delta(t)$  from  $K(t)$  is a collapse for all  $t > t_0$ . For this purpose, it will be crucial to build filtrations that are simple (see Figure 1):

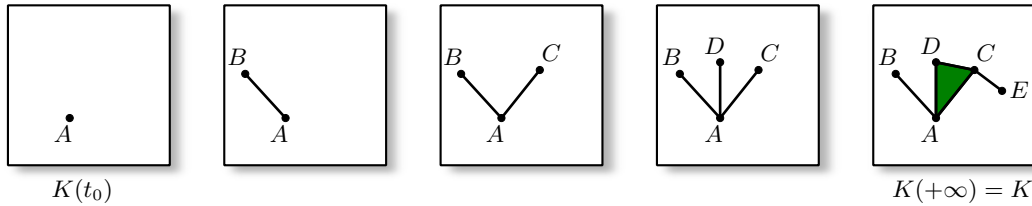
► **Definition 1.** *The filtration  $\{K(t)\}_{t \in \mathbb{R}}$  is simple if for all  $t > t_0$ , then  $\Delta(t)$  has a unique inclusion-minimal element  $\sigma_t$ .*



■ **Figure 1** The 6 simplicial complexes form a filtration of  $K$  which is simple.

When the filtration is simple,  $\Delta(t)$  is precisely the star of  $\sigma_t$  in  $K(t)$  and removing  $\Delta(t)$  from  $K(t)$  is a collapse if one of the three conditions listed in Section 2.3 is satisfied.

► **Definition 2.** *Two simplices  $\sigma^1$  and  $\sigma^2$  are in conjunction in filtration  $\{K(t)\}_{t \in \mathbb{R}}$  if they are both inclusion-minimal elements of  $\Delta(t)$  for some  $t > t_0$  (see Figure 2).*



■ **Figure 2** The 5 simplicial complexes form a filtration of  $K$  which is not simple as simplices  $CD$  and  $E$  are in conjunction.

Clearly, a filtration is simple if and only if it contains no pair of simplices in conjunction. The following definition will be useful when, in the next section, we perturb a filtration to make it simple.

► **Definition 3.** *Consider a filtration  $\{K(t)\}_{t \in \mathbb{R}}$  of  $K$  and a filtration  $\{L(t)\}_{t \in \mathbb{R}}$  of  $L$ . We say that the filtration  $\{L(t)\}_{t \in \mathbb{R}}$  is finer than the filtration  $\{K(t)\}_{t \in \mathbb{R}}$  if for all  $t \in \mathbb{R}$ , there exists  $t' \in \mathbb{R}$  such that  $K(t) = L(t')$ .*

► **Remark.** If  $\{L(t)\}_{t \in \mathbb{R}}$  finer than  $\{K(t)\}_{t \in \mathbb{R}}$ , then  $\eta \prec_K \nu \implies \eta \prec_L \nu$ .

### 3 Collapsing nerves of compact convex sets

#### 3.1 Statement of results

Given a finite collection of sets  $\mathcal{C}$ , we write  $\bigcup \mathcal{C}$  for the union of sets in  $\mathcal{C}$  and  $\bigcap \mathcal{C}$  for the common intersection of sets in  $\mathcal{C}$ . The nerve of  $\mathcal{C}$  is the abstract simplicial complex that consists of all non-empty subcollections whose sets have a non-empty common intersection,

$$\text{Nrv } \mathcal{C} = \{\eta \subseteq \mathcal{C} \mid \bigcap \eta \neq \emptyset\}.$$



The nerve theorem implies that if all sets in  $\mathcal{C}$  are compact and convex, then the nerve of  $\mathcal{C}$  is homotopy equivalent to the union of sets in  $\mathcal{C}$ , that is  $\text{Nrv } \mathcal{C} \simeq \bigcup \mathcal{C}$ . We get immediately that if furthermore  $\bigcup \mathcal{C}$  is a non-empty convex set, then  $\text{Nrv } \mathcal{C}$  is contractible. In this paper, we prove that under the same hypotheses on  $\mathcal{C}$ , we have a stronger result, namely, that  $\text{Nrv } \mathcal{C}$  is collapsible. Formally:

► **Theorem 4.** *If  $\mathcal{C}$  is a finite collection of compact convex sets whose union  $\bigcup \mathcal{C}$  is a non-empty convex set, then  $\text{Nrv } \mathcal{C}$  is collapsible.*

Two corollaries follow immediately. First, we obtain collapsibility of the Delaunay complex. Recall that given a finite point set  $S \subseteq \mathbb{R}^d$ , the *Voronoi region* of  $a \in S$  is  $V_a = \{x \in \mathbb{R}^d \mid \|x - a\| \leq \|x - s\|, \forall s \in S\}$  and the *Delaunay complex* of  $S$  is the nerve of Voronoi regions,  $\text{Del}(S) = \text{Nrv}\{V_s \mid s \in S\}$ .

► **Corollary 5.** *The Delaunay complex of any finite point set  $S \subseteq \mathbb{R}^d$  is collapsible.*

**Proof.** Apply Theorem 4 to the collection  $\mathcal{C} = \{V_s \cap B \mid s \in S\}$ , where  $B$  is any ball large enough to intersect all common intersection  $\bigcap_{s \in \sigma} V_s$  for  $\sigma$  ranging over  $\text{Nrv } S$ . The result follows because  $\text{Del}(S)$  is isomorphic to  $\text{Nrv } \mathcal{C}$  which is collapsible by Theorem 4. ◀

To state the second corollary, recall that the Čech complex of a finite point set  $S \subseteq \mathbb{R}^d$  with parameter  $r \in \mathbb{R}$  is the nerve of the collection of balls,  $\mathcal{C}(S, r) = \text{Nrv}\{B(s, r) \mid s \in S\}$  and denote by  $\text{Hull}(S)$  the convex hull of  $S$ .

► **Corollary 6.** *The Čech complex of a finite point set  $S \subseteq \mathbb{R}^d$  with parameter  $r \in \mathbb{R}$  is collapsible whenever  $\text{Hull}(S) \subseteq S^{\oplus r}$ .*

Before proving the corollary, notice that the condition on  $S$  is tight as for any  $\delta > 0$ , if  $\text{Hull}(S) \subseteq S^{\oplus(r+\delta)}$ , then  $\mathcal{C}(S, r)$  is not necessarily collapsible. Take for instance the Čech complex of two points at distance  $2(r + \delta)$ .

**Proof.** Apply Theorem 4 to the collection  $\mathcal{C} = \{B(s, r) \cap \text{Hull}(S) \mid s \in S\}$  and notice that  $\text{Nrv } \mathcal{C}$  is isomorphic to  $\mathcal{C}(S, r)$ . ◀

We prove Theorem 4 by adopting the following strategy. Consider a finite collection  $\mathcal{C}$  of compact convex sets whose union  $\bigcup \mathcal{C}$  is non-empty and convex. We first build a filtration of  $\text{Nrv } \mathcal{C}$  from which we derive a sequence of collapses reducing  $\text{Nrv } \mathcal{C}$  to a vertex. The rest of the section is devoted to the proof of Theorem 4. In Section 3.2, we build the filtration and show that the smallest non-empty simplicial complex in the filtration is collapsible. In Section 3.3, we show that we can always perturb the collection  $\mathcal{C}$  so that the filtration associated to  $\text{Nrv } \mathcal{C}$  enjoys nice properties (in a sense to be made precise). In Section 3.4, we study the filtration of the perturbed collection and show that events through the filtration are collapses.

### 3.2 Building a filtration

Let  $\mathcal{C}$  be a family of subsets of  $\mathbb{R}^d$  whose union  $\bigcup \mathcal{C}$  is non-empty. Let  $o$  be a fix point in  $\mathbb{R}^d$  that belongs to  $\bigcup \mathcal{C}$ . We build a filtration of  $\text{Nrv } \mathcal{C}$  by sweeping the space  $\mathbb{R}^d$  with a sphere centered at  $o$  and whose radius  $t \geq 0$  continuously increases from 0 to  $+\infty$ . Simplicial complexes  $K_{o, \mathcal{C}}(t)$  in the filtration are obtained by keeping simplices in  $\text{Nrv } \mathcal{C}$  which are subcollections of  $\mathcal{C}$  whose common intersections have a distance to  $o$  equal to or less than  $t$ :

$$K_{o, \mathcal{C}}(t) = \{\eta \subseteq \mathcal{C} \mid d(o, \bigcap \eta) \leq t\}.$$

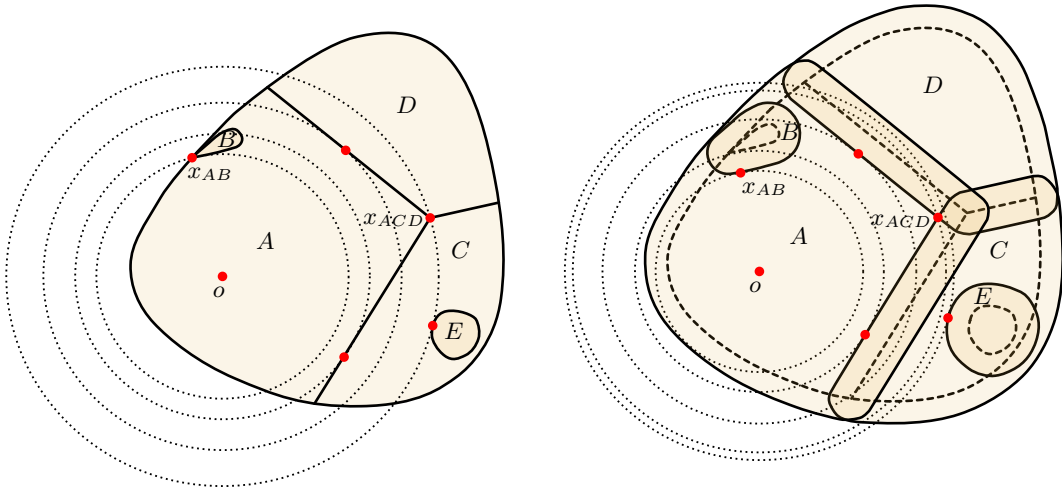
## 11:6 When Convexity Helps Collapsing Complexes

Clearly,  $K_{o,\mathcal{C}}(t) = \emptyset$  for all  $t < 0$  and  $K_{o,\mathcal{C}}(+\infty) = \text{Nrv } \mathcal{C}$ . Notice that  $K_{o,\mathcal{C}}(0)$  is non-empty because point  $o$  belongs to at least one set  $C$  in the collection  $\mathcal{C}$  and thus  $K_{o,\mathcal{C}}(0)$  contains at least vertex  $C$ . It follows that  $K_{o,\mathcal{C}}(0)$  is the smallest non-empty simplicial complex in the filtration. As we continuously increase the parameter  $t$  from  $-\infty$  to  $+\infty$ , the simplicial complex  $K_{o,\mathcal{C}}(t)$  changes only at finitely many values of  $t$  for which the set of simplices

$$\Delta_{o,\mathcal{C}}(t) = \{\eta \subseteq \mathcal{C} \mid d(o, \bigcap \eta) = t\}$$

is non-empty. When these events happen, the sphere centered at  $o$  with radius  $t$  passes through particular points of  $\bigcup \mathcal{C}$  that we call *trigger points* and defined below; see Figure 3.

► **Definition 7.** Consider  $\eta \subseteq \mathcal{C}$  such that  $\bigcap \eta \neq \emptyset$ . The trigger point of  $\eta$  (with respect to  $o$ ) is the point of  $\bigcap \eta$  at which the smallest distance to  $o$  is achieved. There is thus one trigger point per simplex in  $\text{Nrv } \mathcal{C}$  and the set of all these points is referred to as the trigger points of  $\text{Nrv } \mathcal{C}$  (with respect to  $o$ ).



■ **Figure 3** Left: Collection of five convex sets  $\{A, B, C, D, E\}$  whose union is convex. The nerve possesses six trigger points (red dots) among which one of them  $x_{AB}$  lies on the boundary of the union and another one  $x_{ACD}$  does not lie in the interior of any convex set in the collection. The filtration associated to the nerve is depicted in Figure 2 and is not simple. Right: as we offset sets in the collection while keeping the union convex and the nerve unchanged, the trigger point  $x_{AB}$  moves in the interior of the union and the trigger point  $x_{ACD}$  moves in the interior of at least one convex set (namely  $A$ ). The associated filtration is depicted in Figure 1 and is simple.

► **Lemma 8.** Let  $\mathcal{C}$  be a family of subsets of  $\mathbb{R}^d$  whose union  $\bigcup \mathcal{C}$  is non-empty and convex and let  $o \in \bigcup \mathcal{C}$ . Then,  $K_{o,\mathcal{C}}(0)$  is collapsible.

**Proof.** Let  $\tau_0 = \{C \in \mathcal{C} \mid o \in C\}$ . The simplicial complex  $K_{o,\mathcal{C}}(0) = \{\eta \subseteq \mathcal{C} \mid d(o, \bigcap \eta) = 0\}$  consists of  $\tau_0$  together with all its faces and is clearly collapsible. ◀

### 3.3 Perturbing filtrations

Let  $\mathcal{C}$  be a finite family of compact convex sets whose union  $\bigcup \mathcal{C}$  is convex and non-empty and pick a point  $o$  in  $\bigcup \mathcal{C}$ . In this section, we establish that it is always possible to perturb the family  $\mathcal{C}$  into a family of compact convex sets so as to make the filtration  $K_{o,\mathcal{C}}(t)$  simple while leaving  $\bigcup \mathcal{C}$  convex and  $\text{Nrv } \mathcal{C}$  unchanged. Roughly speaking, we shall perturb sets in the collection by thickening them.

► **Lemma 9.** *For all  $\delta > 0$ , there exists a non-negative map  $\alpha : \mathcal{C} \rightarrow \mathbb{R}$  bounded above by  $\delta$  such that if we replace each set  $C \in \mathcal{C}$  by set  $C^{\oplus\alpha(C)} \cap \bigcup \mathcal{C}$ , we perturb  $\mathcal{C}$  in such a way that (1) the nerve of  $\mathcal{C}$  is left unchanged; (2) the filtration  $K_{o,\mathcal{C}}(t)$  becomes simple.*

The proof of Lemma 9 relies on three technical lemmas. To state them, we need some notation and definitions. A *perturbation* of  $\mathcal{C}$  is a map  $f : \mathcal{C} \rightarrow \mathcal{P}(\mathbb{R}^d)$ , where  $\mathcal{P}(\mathbb{R}^d)$  designates the power set of  $\mathbb{R}^d$ . For a subcollection  $\eta \subseteq \mathcal{C}$  and a simplicial complex  $L$  over  $\mathcal{C}$ , we write  $f(\eta) = \{f(C) \mid C \in \eta\}$  and  $f(L) = \{f(\eta) \mid \eta \in L\}$ . For any non-negative map  $\alpha : \mathcal{C} \rightarrow \mathbb{R}$ , we write  $\eta^{\oplus\alpha} = \{C^{\oplus\alpha(C)} \mid C \in \eta\}$  and  $L^{\oplus\alpha} = \{\eta^{\oplus\alpha} \mid \eta \in L\}$ . A simple compactness argument implies the first lemma of which the proof is omitted.

► **Lemma 10.** *Let  $\mathcal{X}$  be a finite collection of compact sets. There exists  $\varepsilon > 0$  such that for all non-negative maps  $\xi : \mathcal{X} \rightarrow \mathbb{R}$  bounded above by  $\varepsilon$ , we have  $(\text{Nrv } \mathcal{X})^{\oplus\xi} = \text{Nrv}(\mathcal{X}^{\oplus\xi})$ .*

► **Lemma 11.** *There exists  $\varepsilon > 0$  such that for all non-negative maps  $\alpha : \mathcal{C} \rightarrow \mathbb{R}$  bounded above by  $\varepsilon$  and all  $0 \leq \beta \leq \varepsilon$ , if we replace each set  $C \in \mathcal{C}$  by set  $C^{\oplus\alpha(C)} \cap (\bigcup \mathcal{C})^{\oplus\beta}$ , we perturb  $\mathcal{C}$  in such a way that (1) the nerve of  $\mathcal{C}$  is left unchanged; (2) the filtration  $K_{o,\mathcal{C}}(t)$  after perturbation is finer than what it was before perturbation.*

**Proof.** Consider a non-negative map  $\alpha$  bounded above by  $\varepsilon$  and  $0 \leq \beta \leq \varepsilon$ . Write  $f_{\alpha,\beta}(C) = C^{\oplus\alpha(C)} \cap (\bigcup \mathcal{C})^{\oplus\beta}$ . To prove (1), we apply Lemma 10 to the set  $\mathcal{X} = \mathcal{C} \cup \{\bigcup \mathcal{C}\}$  and the map  $\xi$  defined by  $\xi(C) = \alpha(C)$  for all  $C \in \mathcal{C}$  and  $\xi(\bigcup \mathcal{C}) = \beta$ . We know that for  $\varepsilon > 0$  small enough,  $\text{Nrv } \mathcal{X} = \text{Nrv}(\mathcal{X}^{\oplus\xi})$  or equivalently  $\bigcap \eta \neq \emptyset \Leftrightarrow \bigcap f_{\alpha,\beta}(\eta) \neq \emptyset$  for all  $\eta \subseteq \mathcal{C}$ , showing that (1) holds. To prove (2), write

$$K(t) = \{\eta \subseteq \mathcal{C} \mid d(o, \bigcap \eta) \leq t\}$$

$$L_{\alpha,\beta}(t) = \{\eta \subseteq \mathcal{C} \mid d(o, \bigcap f_{\alpha,\beta}(\eta)) \leq t\}$$

and let us establish that  $L_{\alpha,\beta}(t)$  is finer than  $K(t)$ . As we continuously increases the parameter  $t$  from  $-\infty$  to  $+\infty$ , the simplicial complex  $K(t)$  changes only at finitely many values  $0 = t_0 < \dots < t_m$  of  $t$ . Let  $B_i = B(o, t_i)$  for  $i \in \{0, \dots, m\}$ . Let us apply Lemma 10 to the set  $\mathcal{X} = \mathcal{C} \cup \{\bigcup \mathcal{C}\} \cup \{B_0, \dots, B_m\}$  and the map  $\xi$  defined by  $\xi(C) = \alpha(C)$  for all  $C \in \mathcal{C}$ ,  $\xi(\bigcup \mathcal{C}) = \beta$  and  $\xi(B_i) = 0$ . Lemma 10 implies that for  $\varepsilon > 0$  small enough, we have the following five equivalences:  $\eta \in K(t_i) \Leftrightarrow d(o, \bigcap \eta) \leq t_i \Leftrightarrow B_i \cap \bigcap \eta \neq \emptyset \Leftrightarrow B_i \cap \bigcap f_{\alpha,\beta}(\eta) \neq \emptyset \Leftrightarrow d(o, \bigcap f_{\alpha,\beta}(\eta)) \leq t_i \Leftrightarrow \eta \in L_{\alpha,\beta}(t_i)$ . Thus, for all  $t \geq 0$ , there exists  $i$  such that  $K(t) = K(t_i) = L(t_i)$  and  $L_{\alpha,\beta}(t)$  is finer than  $K(t)$ . ◀

Next lemma ensures that when we replace one of the set  $C$  in  $\mathcal{C}$  by  $C^{\oplus\varepsilon} \cap \bigcup \mathcal{C}$ , some of the events in the corresponding filtration happen at an earlier time. Precisely:

► **Lemma 12.** *Let  $C \in \mathcal{C}$  and  $\eta \subseteq \mathcal{C}$  such that  $C \not\subseteq \eta$ . Suppose  $\{C\} \cup \eta$  is inclusion-minimal in  $\Delta_{o,\mathcal{C}}(t)$ . Then, for all  $t > d(o, \bigcup \mathcal{C})$  and all  $\varepsilon > 0$ , we have  $d(o, C^{\oplus\varepsilon} \cap \bigcap \eta \cap \bigcup \mathcal{C}) < t$ .*

**Proof.** By assumption,  $d(o, C \cap \bigcap \eta) = t$ . If  $\eta \neq \emptyset$ , then  $\eta$  is a proper subset of  $\{C\} \cup \eta$  and the minimality of  $\{C\} \cup \eta$  in  $\Delta_{o,\mathcal{C}}(t)$  implies that  $d(o, \bigcap \eta) < t$  and therefore  $d(o, \bigcap \eta \cap \bigcup \mathcal{C}) < t$ . If  $\eta = \emptyset$ , observe that the later inequality holds because we have assumed that  $d(o, \bigcup \mathcal{C}) < t$ . Let  $x$  be the point in  $C \cap \bigcap \eta$  closest to  $o$  and let  $x'$  be the point in  $\bigcap \eta \cap \bigcup \mathcal{C}$  closest to  $o$ . As we go from  $x$  to  $x'$  on the segment connecting  $x$  to  $x'$ , the distance to  $o$  decreases in a sufficiently small neighborhood of  $x$  while we remain in both sets  $C^{\oplus\varepsilon}$  and  $\bigcap \eta \cap \bigcup \mathcal{C}$ . Thus,  $d(o, C^{\oplus\varepsilon} \cap \bigcap \eta \cap \bigcup \mathcal{C}) < t$ . ◀

Before proving Lemma 9, recall that two simplices  $\sigma^1$  and  $\sigma^2$  are in conjunction in filtration  $K_{o,\mathcal{C}}(t)$  if they are both inclusion-minimal elements of  $\Delta_{o,\mathcal{C}}(t)$  for some  $t > 0$ .

**Proof of Lemma 9.** The proof consists in applying a sequence of elementary perturbations to set  $\mathcal{C}$  while preserving  $\text{Nrv } \mathcal{C}$  and  $\bigcup \mathcal{C}$  until no two simplices remain in conjunction in the filtration  $K_{o,\mathcal{C}}(t)$ . Suppose two simplices are in conjunction, say  $\sigma^1$  and  $\sigma^2$ . Suppose  $C \in \sigma^1$  and  $C \notin \sigma^2$  and replace the convex set  $C$  by  $C^{\oplus\varepsilon} \cap \bigcup \mathcal{C}$ . Clearly,  $\mathcal{C}$  is still a collection of compact convex sets,  $\bigcup \mathcal{C}$  is left unchanged by the operation and Lemma 11 implies that for  $\varepsilon > 0$  small enough, the nerve of  $\mathcal{C}$  is also left unchanged. We prove below that for  $\varepsilon > 0$  small enough: (1)  $\sigma^1$  and  $\sigma^2$  are not in conjunction anymore after the operation; (2) two simplices  $\eta$  and  $\nu$  are not in conjunction after the operation unless a face  $\eta' \subseteq \eta$  and a face  $\nu' \subseteq \nu$  were already in conjunction before the operation. Introduce the map  $f : \mathcal{C} \rightarrow \mathcal{P}(\mathbb{R}^d)$  defined by  $f(C) = C^{\oplus\varepsilon} \cap \bigcup \mathcal{C}$  and  $f(C') = C'$  for all  $C' \neq C$ .

- (1) Let us prove that for  $\varepsilon > 0$  small enough,  $f(\sigma^1)$  and  $f(\sigma^2)$  are not in conjunction in  $K_{o,f(\mathcal{C})}(t)$ . Let  $t = d(o, \bigcap \sigma^1) = d(o, \bigcap \sigma^2)$  and suppose  $\sigma^1 = \{C\} \cup \eta$  with  $C \notin \eta$ . By construction,  $f(\sigma^1) = C^{\oplus\varepsilon} \cap \bigcap \eta \cap \bigcup \mathcal{C}$ . By Lemma 12, we have  $d(o, \bigcap f(\sigma^1)) < t = d(o, \bigcap f(\sigma^2))$ , showing that  $f(\sigma^1)$  and  $f(\sigma^2)$  are not in conjunction in  $K_{o,f(\mathcal{C})}(t)$ .
- (2) Let us prove that for  $\varepsilon > 0$  small enough, two simplices  $f(\eta)$  and  $f(\nu)$  cannot be in conjunction in  $K_{o,f(\mathcal{C})}(t)$  unless a face  $\eta' \subseteq \eta$  and a face  $\nu' \subseteq \nu$  are in conjunction in  $K_{o,\mathcal{C}}(t)$ . Consider two simplices  $f(\eta)$  and  $f(\nu)$  in conjunction in  $K_{o,f(\mathcal{C})}(t)$ . By Lemma 11, the filtration  $K_{o,f(\mathcal{C})}(t)$  is finer than the filtration  $f(K_{o,\mathcal{C}}(t))$  and the remark in Section 2.4 entails the implication:  $d(o, \bigcap f(\eta)) = d(o, \bigcap f(\nu)) \implies d(o, \bigcap \eta) = d(o, \bigcap \nu)$ . The result hence follows.

To remove all pair of simplices in conjunction, consider the partial order  $\prec$  on pair of simplices defined by  $(\nu', \eta') \prec (\nu, \eta)$  if  $\dim \nu' \leq \dim \nu$  and  $\dim \eta' \leq \dim \eta$ . Sort all pair of simplices in conjunction according to a total order compatible with this partial order. Take the smallest pair  $(\sigma^1, \sigma^2)$  and apply the elementary perturbation described above. Notice that the operation does not create any new pair of simplices in conjunction smaller than  $(\sigma^1, \sigma^2)$ . By repeating this operation a finite number of times, we thus get a new collection of compact convex sets possessing the same nerve and the same union but for which no two simplices are in conjunction anymore. In other words, the filtration associated to the new collection is simple. Since elementary perturbations can be made as small as wanted, their composition can be made smaller than any given  $\delta > 0$ .  $\blacktriangleleft$

### 3.4 Studying events in the filtration

Throughout this section,  $\mathcal{C}$  designates a finite collection of compact convex sets whose union is a non-empty convex set and  $o$  designates a point in  $\bigcup \mathcal{C}$ . In this section, we establish Theorem 4. Let us start with a few general observations. Consider  $t > 0$  such that  $\Delta_{o,\mathcal{C}}(t) \neq \emptyset$ . If we assume  $\{K_{o,\mathcal{C}}(t)\}_{t \in \mathbb{R}}$  to be simple, then by definition  $\Delta_{o,\mathcal{C}}(t)$  has a unique inclusion-minimal element  $\sigma_t$ . Let  $p_t$  be the point in  $\bigcap \sigma_t$  closest to  $o$  and let  $\tau_t = \{C \in \mathcal{C} \mid p_t \in C\}$ .

► **Lemma 13.** *If  $\{K_{o,\mathcal{C}}(t)\}_{t \in \mathbb{R}}$  is simple, then  $\Delta_{o,\mathcal{C}}(t) = \{\eta \subseteq \mathcal{C} \mid \sigma_t \subseteq \eta \subseteq \tau_t\}$ .*

**Proof.** Let us prove that for all  $\eta \subseteq \mathcal{C}$ , we have the equivalence

$$\sigma_t \subseteq \eta \subseteq \tau_t \iff d(o, \bigcap \eta) = t.$$

Consider first  $\eta$  such that  $\sigma_t \subseteq \eta \subseteq \tau_t$ . We have the following sequence of inclusions  $p_t \in \bigcap \tau_t \subseteq \bigcap \eta \subseteq \bigcap \sigma_t$ . Hence,

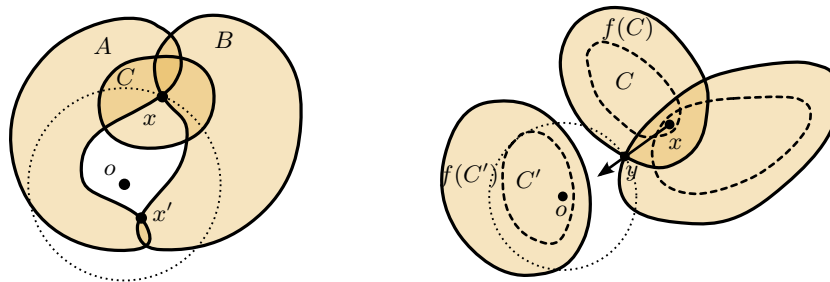
$$t = \|p_t - o\| \geq d(o, \bigcap \tau_t) \geq d(o, \bigcap \eta) \geq d(o, \bigcap \sigma_t) = t$$

showing that  $d(o, \cap \eta) = t$ . Conversely, consider  $\eta \subseteq \mathcal{C}$  such that  $d(o, \cap \eta) = t$  and let  $x$  be the point of  $\cap \eta$  closest to  $o$ . Observe that  $\sigma_t \subseteq \eta$  because  $\eta \in \Delta_{o, \mathcal{C}}(t)$  and therefore  $x \in \cap \eta \subseteq \cap \sigma_t$ . It follows that  $\cap \sigma_t$  contains two points  $x$  and  $p_t$  such that  $\|x - o\| = \|p_t - o\| = t = d(o, \cap \sigma_t)$ . Because  $\cap \sigma_t$  is convex, there is a unique point in  $\cap \sigma_t$  at which the smallest distance to  $o$  is achieved, showing that  $p_t = x$  and therefore  $\eta \subseteq \tau_t$ . ◀

Hence,  $\Delta_{o, \mathcal{C}}(t)$  has a unique inclusion-minimal element  $\sigma_t$  and a unique inclusion-maximal element  $\tau_t$ . More precisely,  $\Delta_{o, \mathcal{C}}(t)$  consists of all cofaces of  $\sigma_t$  in  $K_{o, \mathcal{C}}(t)$  and these cofaces are all faces of  $\tau_t$ . To prove that removing  $\Delta_{o, \mathcal{C}}(t)$  from  $K_{o, \mathcal{C}}(t)$  is a collapse, it suffices to establish that  $\tau_t \neq \sigma_t$ . Lemma 14 below shows that  $p_t$  lies on the boundary of all the convex sets in  $\sigma_t$ .

► **Lemma 14.** *Consider  $\eta \in \text{Nrv } \mathcal{C}$  and let  $x$  be the point in  $\cap \eta$  closest to  $o$ . Suppose  $o \neq x$ . If  $x$  lies in the interior of some  $C$ , then  $x$  is also the point in  $\cap (\eta \setminus \{C\})$  closest to  $o$ .*

**Proof.** Suppose that  $x$  lies in the interior of some  $C \in \eta$ ; see Figure 4, left. Because  $o \neq x$ , we cannot have  $\eta = \{C\}$  and therefore  $\eta' = \eta \setminus \{C\}$  is non-empty. Let us prove that  $x$  is the point of  $\cap \eta'$  closest to  $o$ . Suppose for a contradiction that there exists a point  $x'$  in  $\cap \eta'$  closer to  $o$  than  $x$ . Since the map  $m \mapsto \|m - o\|$  is convex and  $\|x - o\| > \|x' - o\|$ , the distance to  $o$  would be decreasing along the segment  $[x, x']$  in the vicinity of  $x$  and since this segment, in the vicinity of  $x$ , is contained in  $\cap \eta$  this would contradict the fact that  $x$  is the closest point to  $o$  in  $\cap \eta$ . ◀



■ **Figure 4** Left: Counterexample to Lemma 14 when sets in  $\mathcal{C}$  are not convex. Right: Notation for the proof of Lemma 15.

If we were able to prove that  $p_t$  belongs to the interior of one of the convex sets of  $\tau_t$ , we would be done because we would be sure that  $\tau_t \neq \sigma_t$ . Unfortunately, this is not true in general (see point  $x_{ACD}$  in Figure 3 for a counterexample) but becomes true if we slightly perturb  $\mathcal{C}$ , as explained in Lemma 15.

► **Lemma 15.** *Let  $f : \mathcal{C} \rightarrow \mathcal{P}(\mathbb{R}^d)$  be a map such that for all  $C \in \mathcal{C}$ , the subset  $f(C)$  is convex, compact and contains  $C$  in its interior. Suppose  $\text{Nrv } f(\mathcal{C}) = f(\text{Nrv } \mathcal{C})$ . Suppose furthermore that all trigger points of  $\text{Nrv } f(\mathcal{C})$  lie in the interior of  $\cup f(\mathcal{C})$ . Let  $y$  be one of those trigger points. If  $y \neq o$ , then  $y$  lies in the interior of some  $f(C) \in f(\mathcal{C})$ .*

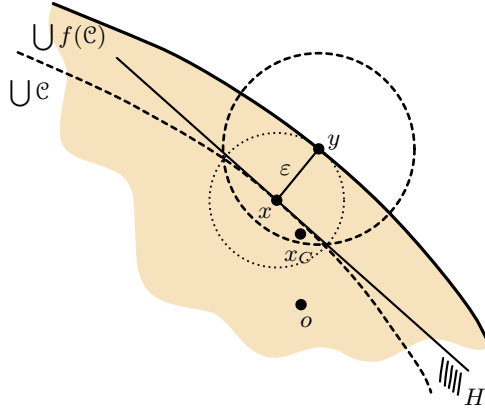
**Proof.** Let  $\tau = \{C \in \mathcal{C} \mid y \in f(C)\}$ . Suppose for a contradiction that  $y \neq o$  and that  $y$  lies on the boundary of  $f(C)$  for all  $C \in \tau$ ; see Figure 4, right. Since  $\text{Nrv } f(\mathcal{C}) = f(\text{Nrv } \mathcal{C})$ , we have that  $y \in \cap f(\tau) \neq \emptyset$  implies that  $\cap \tau \neq \emptyset$ . Let  $x$  be a point in the latter intersection. For all  $C \in \tau$ , we have that  $x$  belongs to the interior of  $f(C)$  while  $y$  belongs to the boundary of  $f(C)$ . Thus, the vector  $y - x$  points outward  $f(C)$  at  $y$  for all  $C \in \tau$ . Since all convex sets  $f(C')$  for which  $C'$  not in  $\tau$  are at some positive distance from  $y$ , it follows that, on the

## 11:10 When Convexity Helps Collapsing Complexes

segment starting from  $y$  in the direction  $y - x$ , points sufficiently close to  $y$  do not belong to  $\bigcup f(\mathcal{C})$ . In other words,  $y \in \partial \bigcup f(\mathcal{C})$ . But,  $y$  being a trigger point,  $y$  lies in the interior of  $\bigcup f(\mathcal{C})$ , yielding a contradiction.  $\blacktriangleleft$

Next lemma gives an example of perturbation  $f$  of  $\mathcal{C}$  which ensures that after perturbing  $\mathcal{C}$  with  $f$ , all trigger points of  $\text{Nrv } f(\mathcal{C})$  are in the interior of  $\bigcup f(\mathcal{C})$ ; see Figure 3.

► **Lemma 16.** *Consider  $\varepsilon > 0$  and a map  $\alpha : \mathcal{C} \rightarrow \mathbb{R}$  such that  $0 \leq \alpha(C) \leq (\sqrt{2} - 1)\varepsilon$ . Let  $f : \mathcal{C} \rightarrow \mathcal{P}(\mathbb{R}^d)$  defined by  $f(C) = C^{\oplus(\varepsilon + \alpha(C))} \cap (\bigcup \mathcal{C})^{\oplus \varepsilon}$ . Suppose  $o \in \bigcup \mathcal{C}$  and  $\text{Nrv } f(\mathcal{C}) = f(\text{Nrv } \mathcal{C})$ . Then, all trigger points of  $\text{Nrv } f(\mathcal{C})$  lie in the interior of  $\bigcup f(\mathcal{C})$ .*



■ **Figure 5** Notation for the proof of Lemma 16.

**Proof.** Consider  $\eta \subseteq \mathcal{C}$  such that  $\bigcap f(\eta) \neq \emptyset$  and suppose for a contradiction that the trigger point  $y$  of  $f(\eta)$  lies on the boundary of  $\bigcup f(\mathcal{C})$ ; see Figure 5. Let  $x$  be the point in  $\bigcup \mathcal{C}$  closest to  $y$  and notice that (1)  $\|x - y\| = \varepsilon$  because  $\bigcup f(\mathcal{C}) = (\bigcup \mathcal{C})^{\oplus \varepsilon}$  and (2)  $x \in \partial \bigcup \mathcal{C}$ . We claim that  $x \in \bigcap f(\eta)$ . Let  $H$  be the closed half-space which contains  $\bigcup \mathcal{C}$  and whose boundary passes through  $x$  while being orthogonal to the vector  $y - x$ . By construction, for all  $C \in \eta$ , there is a point  $x_C$  in  $\bigcup \mathcal{C}$  whose distance to  $y$  is equal to or less than  $\varepsilon + \alpha(C) \leq \sqrt{2}\varepsilon$ . Thus,  $x_C \in H \cap B(y, \sqrt{2}\varepsilon) \subseteq B(x, \varepsilon)$ . Hence,  $\|x - x_C\| \leq \varepsilon$  and therefore  $x$  belongs to  $f(C)$  for all  $C \in \eta$ , showing that  $x \in \bigcap f(\eta)$  as claimed. Note that  $o \in \bigcup \mathcal{C} \subseteq H$ . We thus have three points  $o$ ,  $x$  and  $y$  such that  $y \notin H$ ,  $x$  is the orthogonal projection of  $y$  onto  $H$  and  $o \in H$ . It follows that  $\|x - o\| \leq \|y - o\|$  and since  $x \in \bigcap f(\eta)$ , this contradicts the fact that  $y$  is the trigger point of  $\eta$ , that is, the point of  $\bigcap f(\eta)$  closest to  $o$ .  $\blacktriangleleft$

We are now ready to prove Theorem 4.

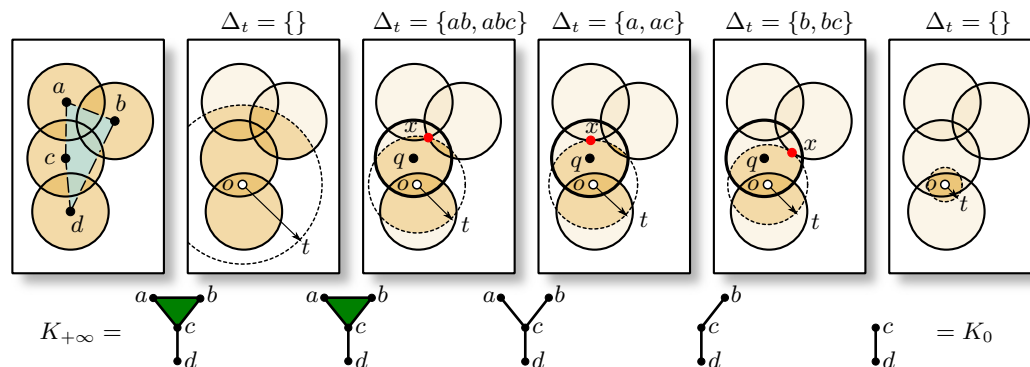
**Proof of Theorem 4.** Let  $o \in \bigcup \mathcal{C}$ . By Lemma 10, for  $\varepsilon > 0$  small enough,  $\text{Nrv } (\mathcal{C}^{\oplus \varepsilon}) = (\text{Nrv } \mathcal{C})^{\oplus \varepsilon}$ . Clearly,  $\mathcal{C}^{\oplus \varepsilon}$  is a collection of compact convex sets whose union is convex since  $\bigcup (\mathcal{C}^{\oplus \varepsilon}) = (\bigcup \mathcal{C})^{\oplus \varepsilon}$ . Applying Lemma 9 to the collection  $\mathcal{C}^{\oplus \varepsilon}$ , we obtain the existence of a perturbation  $f : \mathcal{C} \rightarrow \mathcal{P}(\mathbb{R}^d)$  of the form  $f(C) = C^{\oplus(\varepsilon + \alpha(C))} \cap (\bigcup \mathcal{C})^{\oplus \varepsilon}$  where  $\alpha : \mathcal{C} \rightarrow \mathbb{R}$  is a non-negative map such that (1)  $\text{Nrv } f(\mathcal{C}) = f(\text{Nrv } \mathcal{C})$  and (2) the filtration  $\{K_{o, f(\mathcal{C})}(t)\}_{t \geq 0}$  is simple. Furthermore, the map  $\alpha$  can be chosen arbitrarily small and in particular bounded above by  $(\sqrt{2} - 1)\varepsilon$ . Observe that such an  $f$  satisfies the assumptions of Lemma 16 and therefore of Lemma 15. Letting  $\mathcal{D} = f(\mathcal{C})$ , we prove in the following paragraph that  $\text{Nrv } \mathcal{D}$  is collapsible which entails immediately that  $\text{Nrv } \mathcal{C}$  is collapsible as the two are isomorphic.



By Lemma 8,  $K_{o,\mathcal{D}}(0)$  is collapsible. Let us prove that for all  $t > 0$  such that  $\Delta_{o,\mathcal{D}}(t) \neq \emptyset$ , the operation that removes  $\Delta_{o,\mathcal{D}}(t)$  from  $K_{o,\mathcal{D}}(t)$  is a collapse. Since the filtration  $\{K_{o,\mathcal{D}}(t)\}_{t \geq 0}$  is simple,  $\Delta_{o,\mathcal{D}}(t)$  has a unique inclusion-minimal element  $\sigma_t$ . Let  $p_t$  be the point in  $\bigcap \sigma_t$  closest to  $o$  and let  $\tau_t = \{D \in \mathcal{D} \mid p_t \in D\}$ . By Lemma 13,  $\Delta_{o,\mathcal{D}}(t)$  is the set of simplices  $\eta \in \text{Nrv } \mathcal{D}$  such that  $\sigma_t \subseteq \eta \subseteq \tau_t$ . Let us show that  $\sigma_t \neq \tau_t$ . By Lemma 14,  $p_t$  lies on the boundary of all sets in  $\sigma_t$ . By Lemma 15,  $p_t$  lies in the interior of at least one set  $D$  in  $\tau_t$ . Hence,  $\tau_t \neq \sigma_t$  and the operation that removes  $\Delta_{o,\mathcal{D}}(t)$  from  $K_{o,\mathcal{D}}(t)$  is a collapse. ◀

#### 4 Collapsing Rips complexes

In this section, we turn our attention to Rips complexes. Given a point set  $S$  and a scale parameter  $r$ , the Rips complex  $\mathcal{R}(S, r)$  is the simplicial complex whose simplices are subsets of points in  $S$  with diameter at most  $2r$ . Rips complexes are examples of flag completions. Recall that the flag completion of a graph  $G$ , denoted  $\text{Flag}(G)$ , is the maximal simplicial complex whose 1-skeleton is  $G$ . Let  $G(S, r)$  denote the graph whose vertices are the points  $S$  and whose edges connect all pairs of points within distance  $2r$ . The Rips complex of  $S$  with parameter  $r$  is  $\mathcal{R}(S, r) = \text{Flag}(G(S, r))$ . It is the largest simplicial complex sharing with the Čech complex  $\mathcal{C}(S, r)$  the same 1-skeleton. However, the Rips complex has the computational advantage over the Čech complex to be a flag completion: it suffices to compute its 1-skeleton to encode the whole complex. In this section, we prove that if  $S$  samples sufficiently densely  $\text{Hull}(S)$ , then the Rips complex is collapsible for a suitable value of the scale parameter.



■ **Figure 6** From left to right: Our proof technique (illustrated here when  $S = \{a, b, c, d\}$ ) consists in sweeping space with a sphere centered at  $o \in \text{Hull}(S)$  and whose radius  $t$  continuously decreases from  $+\infty$  to 0. We deduce from the sweep a sequence of collapses reducing  $\mathcal{R}(S, \alpha)$  to a vertex.

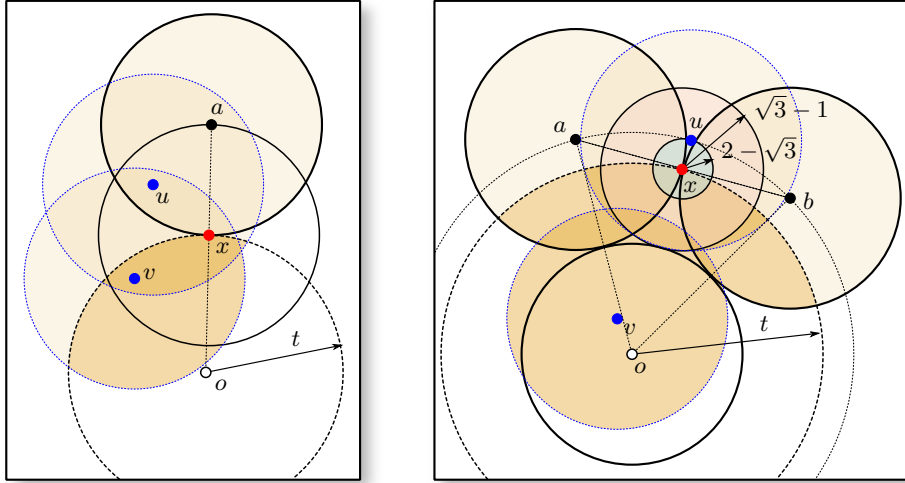
► **Theorem 17.** *Let  $S \neq \emptyset$  be a finite set of points in  $\mathbb{R}^d$  and  $r > 0$ . If  $\text{Hull}(S) \subseteq S^{\oplus(2-\sqrt{3})r}$ , then there exists a sequence of extended collapses reducing  $\mathcal{R}(S, r)$  to a vertex in such a way that the result of each extended collapse is a flag complex.*

**Proof.** Set  $r = 1$  and write  $B_x$  for the closed unit ball centered at  $x$ . Fix a point  $o$  in the convex hull of  $S$ . We construct a sequence of collapses by sweeping the space with a sphere centered at  $o$  and whose radius  $t \geq 0$  continuously decreases from  $+\infty$  to 0. Specifically, let  $G_t$  be the graph whose vertices are points  $s \in S$  such that  $B(o, t) \cap B_s \neq \emptyset$  and whose edges connect all pair of points  $a, b \in S$  such that  $B(o, t) \cap B_a \cap B_b \neq \emptyset$ . Let  $K_t = \text{Flag } G_t$ . Clearly,  $G_{+\infty} = G(S, 1)$  and  $K_{+\infty} = \text{Flag } G_{+\infty} = \mathcal{R}(S, 1)$ . We claim that  $K_0$  is collapsible. Indeed, the vertex set of  $K_0$  is the set of points  $\tau_0 = \{s \in S \mid o \in B_s\} = S \cap B_o$  which is non-empty since  $o \in \text{Hull}(S) \subseteq S^{\oplus 1}$ . It follows that  $K_0 = \text{Flag } G_0$  consists of  $\tau_0$  and all

## 11:12 When Convexity Helps Collapsing Complexes

its faces and is collapsible. As we continuously decrease  $t$  from  $+\infty$  to 0, changes in the simplicial complex  $K_t$  occur whenever a vertex or an edge disappears from the graph  $G_t$ ; see Figure 6. Generically, we may assume that these events do not happen simultaneously.

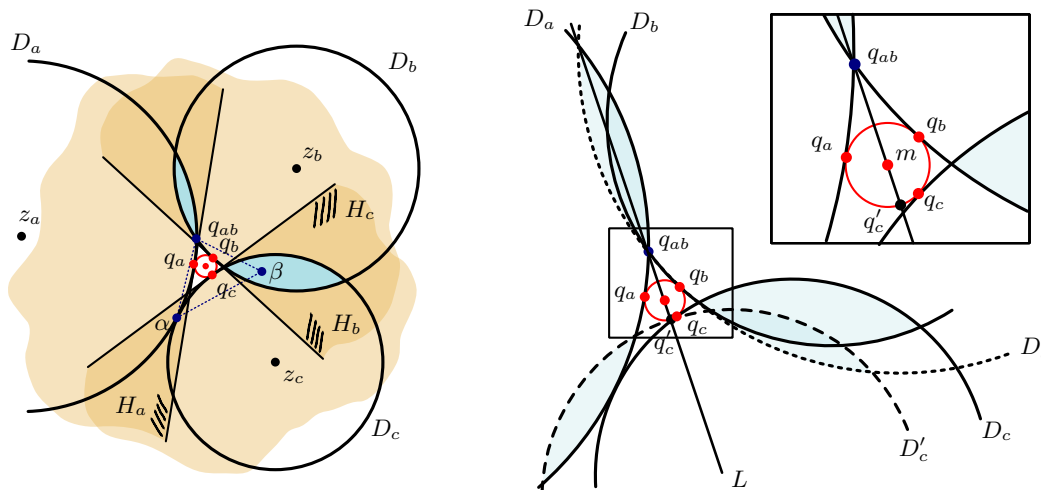
When a vertex  $a$  disappears from  $G_t$  at time  $t$ , the intersection  $B(o, t) \cap B_a$  reduces to a single point  $x$ ; see Figure 7, left. In this situation, we claim that the link of  $a$  in  $K_t$  is the closure of the simplex  $\tau_x = S \cap B_x \setminus \{a\}$ . First, note that  $\tau_x$  is non-empty since  $x$  lies on the segment connecting  $o$  to  $a$  and therefore belongs to the convex hull of  $S$  which is contained in  $S^{\oplus 2 - \sqrt{3}}$ . Hence, there is a point  $s \in S$  in the interior of  $B_x$  and  $\tau_x \neq \emptyset$ . Furthermore,  $\tau_x$  is precisely the vertex set of the link since an edge  $au$  belongs to  $K_t$  if and only if  $B(o, t) \cap B_a \cap B_u \neq \emptyset$  with  $u \in S \setminus \{a\}$  which can be reformulated as  $u \in S \cap B_x \setminus \{a\}$ . Finally, any two vertices  $u$  and  $v$  in the link are connected by an edge since clearly  $u, v \in \tau_x$  implies  $B(o, t) \cap B_u \cap B_v \supset \{x\}$ . We have just proved that the link of  $a$  in  $K_t$  is the closure of a simplex. Thus, removing the star of  $a$  from  $K_t$  is a classical collapse.



■ **Figure 7** Notation for the proof of Theorem 17. Two kinds of events may occur: either a vertex collapse (on the left) or an edge collapse (on the right). The edge collapse is illustrated when triangle  $oab$  is equilateral.

When an edge  $ab$  disappears from  $G_t$  at time  $t$ , there exists a point  $x$  such that  $\{x\} = B(o, t) \cap B_a \cap B_b$ ; see Figure 7, right. Note that  $x$  lies in the convex hull of  $\{a, b, o\}$  and therefore lies in the convex hull of  $S$ . Since  $\text{Hull}(S) \subseteq S^{\oplus 2 - \sqrt{3}}$ , there exists  $u \in S$  such that  $\|u - x\| \leq 2 - \sqrt{3}$ . In particular,  $x \in B_u$  which ensures that both  $x \in B(o, t) \cap B_a \cap B_u \neq \emptyset$  and  $x \in B(o, t) \cap B_b \cap B_u \neq \emptyset$  and therefore  $u$  belongs to the link of  $ab$  in  $K_t$ . We claim that the link of  $ab$  is a cone with apex  $u$ . Consider a point  $v \in S$  which belongs to the link of  $ab$  in  $K_t$ . Equivalently, both  $B(o, t) \cap B_a \cap B_v \neq \emptyset$  and  $B(o, t) \cap B_b \cap B_v \neq \emptyset$  and Lemma 19 implies that  $B(o, t) \cap B(x, \sqrt{3} - 1) \cap B_v \neq \emptyset$ . Since  $\|u - x\| \leq 2 - \sqrt{3}$ , we have  $B(x, \sqrt{3} - 1) \subseteq B_u$  and therefore we also have  $B(o, t) \cap B_u \cap B_v \neq \emptyset$ , showing that  $uv$  also belongs to the link of  $ab$  in  $K_t$ . We have just proved that the link of  $ab$  in  $K_t$  is a cone. Thus, removing the star of  $ab$  from  $K_t$  is an extended collapse. ◀





■ **Figure 8** Notation for the proof of Lemma 18.

### 4.1 Two geometric lemmas

The proof of Theorem 17 relies on two geometric lemmas. The first one states facts about three disks in the plane that intersect pairwise but have no common intersection (Lemma 18). It will allow us to deduce facts about the way four balls intersect in  $\mathbb{R}^d$  (Lemma 19). As before,  $B_x$  denotes the unit closed ball centered at  $x$ .

► **Lemma 18.** *Let  $D_a, D_b$  and  $D_c$  be three disks with radius equal to or less than one and such that any two disks have a non-empty intersection while the three together have no common intersection. Let  $q_{ab}$  be the point of  $D_a \cap D_b$  closest to the center of  $D_c$ . There exists a point  $q_c \in D_c$  such that:*

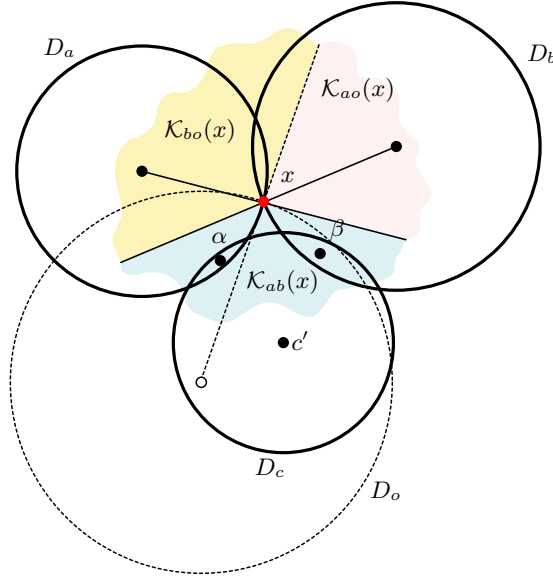
- for all points  $\alpha \in D_a \cap D_c$  and  $\beta \in D_b \cap D_c$ , the point  $q_c$  is in the convex hull of  $\alpha, \beta$  and  $q_{ab}$ ;
- $\|q_c - q_{ab}\| \leq \sqrt{3} - 1$ .

**Proof.** Consider the disk  $D_m$  whose boundary is tangent to the boundaries of the three disks  $D_a, D_b$  and  $D_c$  and whose interior intersects none of the three disks  $D_a, D_b$  and  $D_c$ . For  $x \in \{a, b, c\}$ , the two disks  $D_x$  and  $D_m$  intersect in a single point  $q_x$ ; see Figure 8, left. Let  $\alpha \in D_a \cap D_c$  and  $\beta \in D_b \cap D_c$ . We claim that  $q_c$  belongs to the convex hull of  $\alpha, \beta$  and  $q_{ab}$ . Indeed, for  $x \in \{a, b, c\}$ , let  $H_x$  be the half-plane that contains  $D_x$  and avoids the interior of  $D_m$ . We have  $\alpha \in H_a \cap H_c, \beta \in H_b \cap H_c$ , and  $q_{ab} \in H_a \cap H_b$ . The triangle  $\alpha\beta q_{ab}$  covers the closure of  $\mathbb{R}^2 \setminus (H_a \cup H_b \cup H_c)$  and therefore  $q_c$ .

Let us prove that  $\|q_c - q_{ab}\| \leq \sqrt{3} - 1$ . For  $x \in \{a, b, c\}$ , we denote the center of  $D_x$  by  $z_x$  and its radius by  $\rho_x$ . We are going to transform the three disks  $D_a, D_b$  and  $D_c$  in such a way that after the transformation:

- (i) the three disks intersect pairwise but have no common intersection;
- (ii) the distance between  $q_c$  and  $q_{ab}$  is at least as large as it was before the transformation;
- (iii)  $\rho_x \leq 1$  for  $x \in \{a, b, c\}$ ;
- (iv) the centers  $z_a, z_b$ , and  $z_c$  form an equilateral triangle of side length two.

Let  $q'_c$  be the point on the boundary of  $D_m$  that is farthest away from  $q_{ab}$ ; see Figure 8, right. Clearly,  $\|q'_c - q_{ab}\| \geq \|q_c - q_{ab}\|$ . The two tangency points  $q_a$  and  $q_b$  decompose the boundary of  $D_m$  in two arcs and it is not difficult to see that one of them contains both



■ **Figure 9** Notation for the proof of Lemma 19.

$q_c$  and  $q'_c$ . Consider the disk  $D'_c$  obtained by rotating  $D_c$  around  $m$  until it meets  $q'_c$ . As we do so, the rotated disk maintains a contact with at least one of the two disks  $D_a$  or  $D_b$ . Without loss of generality, we may assume that  $D_a \cap D'_c \neq \emptyset$ . Let  $L$  be the straight-line passing through  $q_{ab}$ ,  $q'_c$  and  $m$ . Let  $D'_b$  be the symmetric of  $D_a$  with respect to  $L$ . We have  $D'_b \cap D'_c \neq \emptyset$ . The two boundaries of  $D_a$  and  $D'_b$  meet in two points, one of them being  $q_{ab}$ . If we replace  $D_b$  by  $D'_b$  and  $D_c$  by  $D'_c$ , it is easy to check that now the three disks  $D_a$ ,  $D_b$  and  $D_c$  satisfies (i), (ii) and (iii) and their centers form an isosceles triangle. We can then further transform the three disks in such a way that after the transformation, they satisfy in addition (iv). When this is the case, we clearly have  $\|q_c - q_{ab}\| = \sqrt{3} - 1$ . ◀

► **Lemma 19.** *Let  $a$  and  $b$  be two points such that  $B_a$  and  $B_b$  have a non-empty intersection. Let  $o$  be a point such that  $d(o, B_a \cap B_b) = t > 0$ . Let  $x$  be the (unique) point of  $B_a \cap B_b$  closest to  $o$ . Any unit ball which has a non-empty intersection with both  $B_a \cap B(o, t)$  and  $B_b \cap B(o, t)$  has a non-empty intersection with  $B(x, \sqrt{3} - 1) \cap B(o, t)$ .*

**Proof.** Note that  $x \in B(o, t)$ . Let  $c$  such that  $B_c \cap B_a \cap B(o, t) \neq \emptyset$  and  $B_c \cap B_b \cap B(o, t) \neq \emptyset$ . If  $x \in B_c$ , then the claim holds trivially since  $x \in B(x, \sqrt{3} - 1) \cap B(o, t)$ . Let us assume from now on that  $x \notin B_c$ . Take  $\alpha \in B_c \cap B_a \cap B(o, t)$  and  $\beta \in B_c \cap B_b \cap B(o, t)$  and consider a 2-plane  $\Pi$  that contains the three points  $x$ ,  $\alpha$  and  $\beta$ . This 2-plane intersects the four balls  $B_a$ ,  $B_b$ ,  $B_c$  and  $B(o, t)$  in four disks that we denote respectively  $D_a$ ,  $D_b$ ,  $D_c$  and  $D_o$ ; see Figure 9. The three disks  $D_a$ ,  $D_b$  and  $D_o$  have a non-empty intersection reduced to point  $x$ . We have  $\alpha \in D_c \cap D_a \cap D_o \neq \emptyset$ ,  $\beta \in D_c \cap D_b \cap D_o \neq \emptyset$  and  $x \notin D_c$ .

Let  $z_c$  be the center of  $D_c$ . We claim that  $x$  is the point of  $D_a \cap D_b$  closest to  $z_c$ . Define the *outer cone* of  $D_a \cap D_o$  at  $x$  as the set of points:

$$\mathcal{K}_{ao}(x) = \{y \in \Pi \mid \forall z \in D_a \cap D_o, \langle y - x, z - x \rangle \leq 0\}.$$

Equivalently,  $\mathcal{K}_{ao}(x)$  is the set of points whose distance from  $x$  is less than or equal to the distance from any other point of  $D_a \cap D_o$ . The fact that  $x \notin D_c$  while  $\alpha \in D_c$  implies that  $\|z_c - \alpha\| \leq \|z_c - x\|$ . Thus,  $\alpha$  is a point in the intersection  $D_a \cap D_o$  closer to  $z_c$  than  $x$ . Equivalently,  $z_c \notin \mathcal{K}_{ao}(x)$ . Similarly,  $z_c \notin \mathcal{K}_{bo}(x)$ . Since  $\mathcal{K}_{ao}(x) \cup \mathcal{K}_{bo}(x) \cup \mathcal{K}_{ab}(x) = \Pi$ , it follows that  $z_c \in \mathcal{K}_{ab}(x)$ . In other words,  $x$  is the point of  $D_a \cap D_b$  closest to  $z_c$  as claimed.

Therefore, Lemma 18 can be applied and shows the existence of a point  $x' \in D_c$  in the convex hull of  $\alpha$ ,  $\beta$  and  $x$  such that  $x' \in B(x, \sqrt{3} - 1)$ . Since all three points  $\alpha$ ,  $\beta$  and  $x$  belong to  $B(o, t)$ , it follows that  $x' \in B(o, t)$ , yielding the result. ◀

## 5 Future work

We envision that our work could create a bridge towards a non-smooth discrete Morse theory. Continuous Morse theory studies how the homology of a smooth manifold is determined by the critical points of a Morse function. Robin Forman has introduced a discrete Morse Theory [7] enjoying similar properties but defined on simplicial complexes. This has been generalized in [8, 4]. The filtrations that we built can be interpreted as defining a Morse function with a single critical event creating a connected component. By removing the convexity assumption on the union of convex sets, or by varying the function defining the filtration (here the distance to  $o$ ), we get a Morse Theory that may characterize the homology of any non-smooth set that can be expressed as a finite union of convex sets (such as embedded simplicial or polyhedral complexes for examples). While this paper does not explore this generalization, it opens the possibility for a non-smooth, discrete Morse Theory.

---

### References

- 1 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *International Journal of Computational Geometry and Applications (IJCGA)*, 22(4):279–303, 2012.
- 2 D. Attali, A. Lieutier, and D. Salinas. Vietoris-Rips complexes also provide topologically correct reconstructions of sampled shapes. *Computational Geometry: Theory and Applications (CGTA)*, 2012. doi:10.1016/j.comgeo.2012.02.009.
- 3 Dominique Attali and André Lieutier. Geometry-driven collapses for converting a Čech complex into a triangulation of a nicely triangulable shape. *Discrete & Computational Geometry*, 54(4):798–825, 2015.
- 4 Ulrich Bauer and Herbert Edelsbrunner. The Morse theory of Čech and Delaunay complexes. *Transactions of the American Mathematical Society*, 369(5):3741–3762, 2017.
- 5 Bruno Benedetti. Discrete Morse theory for manifolds with boundary. *Transactions of the American Mathematical Society*, 364(12):6631–6670, 2012.
- 6 Karol Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fundamenta Mathematicae*, 35(1):217–234, 1948. URL: <http://eudml.org/doc/213158>.
- 7 Robin Forman. Morse theory for cell complexes, 1998.
- 8 Ragnar Freij. Equivariant discrete Morse theory. *Discrete Mathematics*, 309(12):3821–3829, 2009.
- 9 A. Hatcher. *Algebraic topology*. Cambridge University Press, 2002.



# Optimal Algorithm for Geodesic Farthest-Point Voronoi Diagrams

Luis Barba

Department of Computer Science, ETH Zürich, Switzerland

luis.barba@inf.ethz.ch

---

## Abstract

Let  $P$  be a simple polygon with  $n$  vertices. For any two points in  $P$ , the geodesic distance between them is the length of the shortest path that connects them among all paths contained in  $P$ . Given a set  $\mathcal{S}$  of  $m$  sites being a subset of the vertices of  $P$ , we present the first randomized algorithm to compute the geodesic farthest-point Voronoi diagram of  $\mathcal{S}$  in  $P$  running in expected  $O(n + m)$  time. That is, a partition of  $P$  into cells, at most one cell per site, such that every point in a cell has the same farthest site with respect to the geodesic distance. This algorithm can be extended to run in expected  $O(n + m \log m)$  time when  $\mathcal{S}$  is an arbitrary set of  $m$  sites contained in  $P$ .

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Geodesic distance, simple polygons, farthest-point Voronoi diagram

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.12

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1809.01481>.

## 1 Introduction

Let  $P$  be a simple  $n$ -gon. Let  $\mathcal{S}$  be a set of  $m \geq 3$  weighted *sites* (points) contained in  $V(P)$ , where  $V(P)$  denotes the set of vertices of  $P$ . That is, we have a function  $w : \mathcal{S} \rightarrow \mathbb{R}$  that assigns to each site of  $\mathcal{S}$  a non-negative weight. We also extend the weight function to any point in  $P$  by setting  $w(x) = 0$  for all  $x \in P \setminus \mathcal{S}$ . While we could allow the sites to lie anywhere on the boundary,  $\partial P$ , of  $P$ , as long as we know their clockwise order along  $\partial P$ , we can split the edges of  $P$  at the sites, and produce a new polygon where each site coincides with a vertex. Therefore, we assume that  $\mathcal{S} \subseteq V(P)$ .

Given two points  $x, y$  in  $P$  (either on the boundary or in the interior), the *geodesic path*  $\pi_P(x, y)$  is the shortest path contained in  $P$  connecting  $x$  with  $y$ . If the straight-line segment connecting  $x$  with  $y$  is contained in  $P$ , then  $\pi_P(x, y)$  is the straight-line segment  $xy$ . Otherwise,  $\pi_P(x, y)$  is a polygonal chain whose vertices (other than its endpoints) are reflex vertices of  $P$ . We refer the reader to [14] for more information on geodesic paths.

For a segment  $xy$ , we denote its Euclidean length by  $|xy|$ . For a path, its *Euclidean length* is the sum of the Euclidean length of all of its segments. Given two points  $x$  and  $y$  in  $P$ , their *geodesic distance*  $G^P(x, y)$  is the Euclidean length of  $\pi_P(x, y)$ . The *weighted geodesic distance* (or simply *w-distance*) between two points  $x$  and  $y$  in  $P$ , denoted by  $D_w^P(x \rightsquigarrow y)$ , is the sum of  $w(x)$  with the Euclidean length of  $\pi_P(x, y)$ , i.e.,  $D_w^P(x \rightsquigarrow y) = w(x) + G^P(x, y)$ . Notice that if all weights are set to zero, then the *w-distance* coincides with the classical definition of geodesic distance [14]. Moreover, notice that this distance is not symmetric unless the weights of  $x$  and  $y$  coincide.

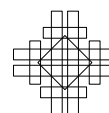
Given a point  $x \in P$ , an  *$\mathcal{S}$ -farthest site* of  $x$  in  $P$  is a site  $s$  of  $\mathcal{S}$  whose *w-distance* to  $x$  is maximized. To ease the description, we assume that each vertex of  $P$  has a unique  $\mathcal{S}$ -farthest neighbor. This *general position* condition was also assumed in [1, 3, 18] and can be obtained by applying a slight perturbation [10].

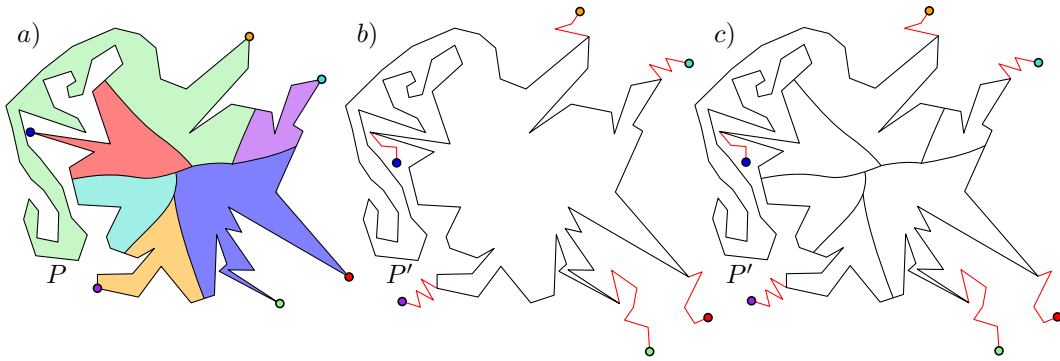


© Luis Barba;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 12; pp. 12:1–12:14



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





■ **Figure 1** a) A simple polygon  $P$  with a set  $\mathcal{S}$  of six weighted sites and their FVD. b) A new polygon  $P'$  where a path of length  $w(s)$  is added at the location of each site  $s \in \mathcal{S}$ . Then the weight is set to zero and moved at the endpoint of its corresponding path. c) The FVD of  $\mathcal{S}$  in  $P$  coincides with the FVD of the new sites in the new polygon  $P'$ .

For a site  $s \in \mathcal{S}$ , let  $\text{Cell}_P(s, \mathcal{S}) = \{x \in P : D_w^P(s \rightsquigarrow x) \geq D_w^P(s' \rightsquigarrow x), \forall s' \in \mathcal{S}\}$  be the (weighted farthest) *Voronoi cell* of  $s$  (in  $P$  with respect to  $\mathcal{S}$ ). That is,  $\text{Cell}_P(s, \mathcal{S})$  consists of all the points of  $P$  that have  $s$  as one of their  $\mathcal{S}$ -farthest sites. The union of all Voronoi cells covers the entire polygon  $P$ , and the closure of the set  $\text{int}(P) \setminus \cup_{s \in \mathcal{S}} \text{int}(\text{Cell}_P(s, \mathcal{S}))$  defines the (weighted farthest) *Voronoi graph* of  $\mathcal{S}$  in  $P$ .

The Voronoi graph together with the set of Voronoi cells defines the *weighted geodesic farthest-point Voronoi diagram* (or simply *FVD*) of  $\mathcal{S}$  in  $P$ , denoted by  $\text{VD}(\mathcal{S}, P)$ . Thus, we indistinctively refer to  $\text{VD}(\mathcal{S}, P)$  as a graph or as a set of Voronoi cells; see Figure 1.

Notice that having non-zero weights on our set of sites does not make the problem harder. To see this, consider a new polygon  $P'$ , where at the location of each site  $s \in \mathcal{S}$ , a path of length  $w(s)$  is attached to the boundary of  $P$ . Additionally, the site  $s$  is given weight zero and is moved to the other endpoint of this path; see Figure 1 for an illustration. In this way we obtain a new weakly simple polygon  $P'$  and a new set of sites  $\mathcal{S}'$  that defines the same FVD as  $\mathcal{S}$  in  $P$ . Therefore, a weighted FVD as described in this paper has the same properties as the classical farthest-point Voronoi diagram constructed using the geodesic distance [3]. In particular, we know that the Voronoi graph is a tree with leaves on the boundary of  $P$ . Also, each edge of this graph consists of a sequence of straight lines and hyperbolic arcs that may intersect  $\partial P$  only at its endpoints [3]. Thus, we refer to the Voronoi graph as a *Voronoi tree*. While working with weighted sites might seem an unnecessary complication, we decided to work with them to ease the description of the recursive construction that our algorithm uses.

Let  $F_P(x, \mathcal{S})$  be the function that maps each  $x \in P$  to the  $w$ -distance to a  $\mathcal{S}$ -farthest neighbor of  $x$  (i.e.,  $F_P(x, \mathcal{S}) = D_w^P(x \rightsquigarrow f_P(x, \mathcal{S}))$ ). Notice that  $F_P(x, \mathcal{S})$  can be seen as the upper envelope of the  $w$ -distance functions from the sites in  $\mathcal{S}$ . Throughout the paper, we will play with this alternative way of thinking of Voronoi diagrams, either as graphs or as upper envelopes. A point  $c \in P$  that minimizes  $F_P(x, \mathcal{S})$  is called the *geodesic center* of  $P$ . Similarly, a point  $s \in P$  that maximizes  $F_P(x, \mathcal{S})$  (together with  $f_P(s)$ ) forms a *diametral pair* and their  $w$ -distance is the *geodesic diameter*.

**Related work.** The problem of computing the geodesic center of a simple  $n$ -gon  $P$  (and its counterpart, the geodesic diameter) were central in the 80's in the computational geometry community. Chazelle [7] provided the first  $O(n^2)$ -time algorithm to compute the geodesic diameter. Suri [21] improved upon it by reducing the running time to  $O(n \log n)$ . Finally, Hershberger and Suri [12] introduced a matrix search technique that allowed them to obtain a linear-time algorithm for computing the diameter.

The first algorithm for computing the geodesic center of  $P$  was given by Asano and Toussaint [4], and runs in  $O(n^4 \log n)$  time. This algorithm computes a super set of the vertices of the Voronoi tree of  $\text{VD}(\mathcal{S}, P)$ , where  $\mathcal{S}$  is the set of vertices of  $P$ . Shortly after, Pollack et al. [20] improved the running time to  $O(n \log n)$ . This remained the best running time for many years until recently when Ahn et al. [1] settled the complexity of this problem by presenting a  $\Theta(n)$ -time algorithm to compute the geodesic center of  $P$ .

The problem of computing the FVD generalizes the problems of computing the geodesic center and the geodesic diameter. For a set  $\mathcal{S}$  of  $m \geq 3$  sites in a simple  $n$ -gon  $P$ , Aronov [3] presented an algorithm to compute  $\text{VD}(\mathcal{S}, P)$  in  $O((n+m) \log(n+m))$  time. While the best known lower bound is  $\Omega(n+m \log m)$ , it was not known whether or not the dependence on  $n$ , the complexity of  $P$ , is linear in the running time. In fact, this problem was explicitly posed by Mitchell [14, Chapter 27] in the Handbook of Computational Geometry, and solving it has become a prominent area of research in recent years. Oh et al. [18] (SoCG'16) present the first improvement to this problem in more than 20 years. Using the new tools presented by Ahn et al. [1], they introduce an  $O(n \log \log n + m \log m)$ -time algorithm to compute  $\text{VD}(\mathcal{S}, P)$ . As a stepping stone, they present an  $O((n+m) \log \log n)$ -time algorithm for the simpler case where all sites are vertices of  $P$ . In fact, any improvement on the latter algorithm translates directly to an improvement on the general problem. In particular, a linear time algorithm for the simpler case with sites on the boundary of  $P$  suffices to match the lower bound and close the problem presented by Mitchell [14, Chapter 27] in the case of farthest-point Voronoi diagrams.

Recently, not only farthest-point Voronoi diagrams have received attention. For the nearest-point geodesic Voronoi diagram, two papers have focused in finding algorithms matching the same lower bound of  $\Omega(n+m \log m)$  [13, 16]. While these results work only for a limited range of  $m$  with respect to  $n$ , both papers have appeared in consecutive years in the Symposium on Computational Geometry (SoCG). In a recent breakthrough to appear in SODA'19, Oh [15] presents an optimal algorithm to compute the nearest-point geodesic Voronoi diagram running in  $O(n+m \log m)$  time. However, the techniques in these papers are fundamentally based on data structures with logarithmic query time, and hence it is not conceivable to adapt them to obtain a linear-time algorithm for the case when the sites are vertices of the polygon.

**Our results.** In this paper, we provide an optimal, albeit randomized, algorithm to compute  $\text{VD}(\mathcal{S}, P)$  for the important case where all sites of  $\mathcal{S}$  are vertices of  $P$ . Our algorithm runs in expected  $\Theta(n+m)$  time, and uses a completely different set of tools to solve the problem. Using the reduction presented by Oh et al. [18], we immediately obtain an algorithm for the general case where the sites can be arbitrary points in  $P$ . This algorithm matches the lower bound and runs in expected  $\Theta(n+m \log m)$  time thereby solving the problem posed by Mitchell [14, Chapter 27] in the case of farthest-point Voronoi diagrams. It remains open to find a deterministic algorithm with the same running time.

**Our approach.** Let  $P$  be a simple  $n$ -gon and let  $\mathcal{S}$  be a set of  $m \geq 3$  sites contained in  $V(P)$ , where  $V(P)$  is the set of vertices of  $P$ . We present a randomized  $O(n+m)$ -time algorithm to compute the FVD of  $\mathcal{S}$  in  $P$ . We would like to use a variation of the randomized incremental construction (*RIC*) for Euclidean farthest-point Voronoi diagrams [9]. This algorithm inserts the sites, one by one, in random order, and constructs the cell of each newly inserted site in time proportional to its size. By bounding the expected size of each cell using backwards analysis, the incremental construction can be carried out in total linear time.



In the geodesic case however, the complexity of a cell depends not only on the set of sites, but also on the complexity of the polygon [2]. Already the FVD of 3 sites can have  $\Omega(n)$  vertices and arcs. Moreover, there is an additional complication when using  $w$ -distances. To achieve an incremental construction, one would need to have at hand a complete description of the  $w$ -distance function  $D_w^P(s \rightsquigarrow x)$  inside of the newly created cell for the inserted site  $s$ . If this function is precomputed in the entire polygon, this would be too costly. Thus, one needs to define these functions only at the specific locations where they are needed. An additional problem is that for a RIC, the first inserted sites must have their  $w$ -distance defined in almost the entire polygon. Thus, already the description-size of the  $w$ -distances needed for the first batch of sites (say the first  $m/100$ ) becomes superlinear. Therefore, it seems hopeless to try a RIC without somehow reducing the complexity of  $P$  throughout the process. Nevertheless, a RIC works well for all the sites that come after this first batch. Intuitively, the latter insertions define smaller cells, and the space needed to describe their  $w$ -distances can be nicely bounded. Thus, the main question is how to deal with this first fraction of the sites.

In this paper we overcome these difficulties with a novel approach, and manage to deal with this first fraction of the sites using pruning. First, we randomly partition the sites into  $B$  and  $R$ , where  $|B| \leq \alpha m$  for some constant  $0 < \alpha < 1$  (Section 3.1). Then, we construct recursively an “approximation” of the FVD of  $B$  (Section 3.2). To this end, we define a new weakly simple polygon  $Q$  containing  $B$  with only a constant fraction of the vertices of  $P$ . Essentially we prune from  $P$  all the vertices that have nothing to do with geodesic paths connecting sites in  $B$  with points in their respective Voronoi cells. Our approximation comes from recursively computing the FVD of  $B$  in  $Q$ . We show that the complexity of  $Q$  decreases sufficiently so that the recursive call leads to a linear overall running time.

However, reducing the complexity comes with a price. The  $w$ -distance from sites of  $B$  inside of  $Q$  turns out to be only “similar” to that in  $P$ . However, we make sure that these functions are accurate where it matters. After computing this “Voronoi-like” diagram for  $B$ , we need to deal with the sites of  $R$ . To this end, we turn to the RIC (Section 4). We compute the  $w$ -distance from sites in  $R$  only inside of specific parts of  $P$ , making sure that they suffice for our purpose, while their overall complexity remains linear. Another challenge comes from the fact that the  $w$ -distances from  $B$  are with respect to  $Q$ , while the ones from  $R$  are not. Thus, we need to prove that the upper envelope of these functions induces a Voronoi diagram. Once we deal with these technical details, we end up with an upper envelope of functions that we prove to coincide with the FVD of  $S$  in  $P$ , finishing our construction.

We show that the insertion of each site  $r \in R$  can be carried out in expected  $O(n/m)$  time. Thus, inserting all sites of  $R$  can be done in expected  $O(n + m)$  time. After inserting the sites of  $R$ , the expected total running time of our algorithm is given by the simple recurrence  $E[T(n, m)] \leq T(n/2, m/2) + O(n + m) = O(n + m)$ . The crucial aspect of our approach that could not be achieved before this paper, is the reduction in the complexity of the polygon. Overall, we combine many different tools, from recursion, pruning, and randomization, together with all the machinery to deal with geodesic functions. Due to space constraints, the proof of all results marked with [\*] have been omitted and can be found in the full version of this paper [5].

## 2 Preliminaries

Let  $P$  be a simple  $n$ -gon and let  $S$  be a set of  $m \geq 3$  sites contained in  $V(P)$ . Because  $S \subseteq V(P)$ , we know that  $m = |S| \leq n$ .

A subset  $G \subseteq P$  is *geodesically convex* in  $P$  if for each  $x, y \in G$ , the geodesic path between  $x$  and  $y$  is contained in  $G$ , i.e., if  $\pi_P(x, y) \subseteq G$ . Given a set  $A$  of points in  $P$ , the *geodesic hull* of  $A$  in  $P$  is the minimum geodesically convex set in  $P$  that contains  $A$ . In particular,



if  $A \subseteq \partial P$ , then the boundary of the geodesic hull of  $A$  is obtained by joining consecutive points of  $A$  along  $\partial P$  by the geodesic path between them. Note that this geodesic hull is not necessarily a simple polygon but a weakly simple polygon. Geodesic functions in weakly simple polygons behave in the exact same way as in simple polygons, and the existing machinery applies directly with no overhead [6]. Thus, while many papers state their results for simple polygons, they apply directly to weakly simple polygons. In particular, all results and tools presented in this paper apply directly to weakly simple polygons. This remark is already crucial in several recent papers [17, 18].

► **Lemma 1** (Restatement of Lemma 2 of [19]). *Let  $A \subseteq \partial P$  be a set of  $O(n)$  points sorted along  $\partial P$ . Then the geodesic hull of  $A$  in  $P$  can be computed in  $O(n)$  time.*

Let  $\text{VD}_{\partial}(\mathcal{S}, P)$  be the FVD of  $\mathcal{S}$  restricted to the boundary of  $P$ . More formally, for each  $s \in \mathcal{S}$ , let  $\text{bCell}_P(s, \mathcal{S}) = \text{Cell}_P(s, \mathcal{S}) \cap \partial P$  be the *boundary cell* of  $s$  and let  $\text{VD}_{\partial}(\mathcal{S}, P)$  be the union of these boundary cells. The construction of  $\text{VD}_{\partial}(\mathcal{S}, P)$  has often been a stepping stone in the computation of  $\text{VD}(\mathcal{S}, P)$  [3, 18], and our algorithm follows the same approach. The following result from [18] allows us to compute it efficiently.

► **Theorem 2** (Theorem 9 of [18]). *Let  $P$  be an  $n$ -gon and let  $\mathcal{S} \subseteq V(P)$  be a set of sites. Then, we can compute  $\text{VD}_{\partial}(\mathcal{S}, P)$  in  $O(n)$  time.*

Using this procedure, we can find out in  $O(n)$  time which sites of  $\mathcal{S}$  have a non-empty Voronoi cell. Therefore, we can forget about the sites with empty cells and assume without loss of generality from now on that all sites of  $\mathcal{S}$  have non-empty Voronoi cells.

Given a site  $s \in \mathcal{S}$  and a polygonal chain  $C \subseteq \partial P$  with endpoints  $p$  and  $p'$ , the *funnel* of  $s$  to  $C$  in  $P$ , denoted by  $\text{Funnel}_P(s \rightarrow C)$ , is the geodesic hull of  $s$  and  $C$  in  $P$ . It is known that  $\text{Funnel}_P(s \rightarrow C)$  coincides with the weakly simple polygon contained in  $P$  bounded by  $C$ ,  $\pi_P(s, p')$  and  $\pi_P(s, p)$  [1]. For ease of notation, we denote  $\text{Funnel}_P(s \rightarrow \text{bCell}_P(s, \mathcal{S}))$  simply by  $\text{Funnel}_P(s, \mathcal{S})$ , i.e., the funnel with apex  $s$  that goes to  $\text{bCell}_P(s, \mathcal{S})$ . The following lemma shows the relation between Voronoi cells and their funnels.

► **Lemma 3** (Consequence of Lemma 4.1 of [1]). *Given a site  $s \in \mathcal{S}$ , the Voronoi cell  $\text{Cell}_P(s, \mathcal{S})$  is contained in the funnel  $\text{Funnel}_P(s, \mathcal{S})$ .*

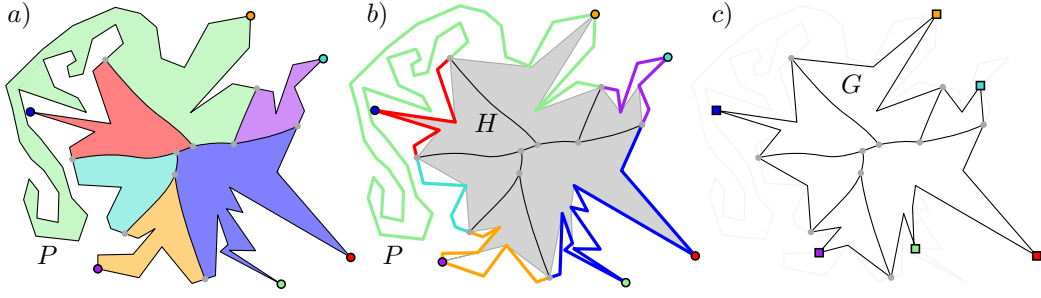
We are also interested in bounding the total complexity of the funnels of sites in  $\mathcal{S}$ . Given a polygon  $Q$ , let  $|Q|$  denote its *combinatorial complexity* (or just *complexity*), i.e., the number of vertices and edges used to represent it.

► **Lemma 4** (Consequence of Corollaries 3.8 and 4.4 of [1]). *Given an  $n$ -gon  $P$  and a set  $\mathcal{S} \subseteq V(P)$ ,  $\sum_{s \in \mathcal{S}} |\text{Funnel}_P(s, \mathcal{S})| = O(n)$ . Also, all funnels can be computed in  $O(n)$  time.*

## 2.1 The simplification transformation

The following transformation allows us to modify the input of our problem and assume some nice structural properties without loss of generality. In this section and for this transformation, we allow the given polygon  $P$  to be weakly simple instead of a simple  $n$ -gon. The result of the transformation described in this section takes a weakly simple polygon with a set of weighted sites as input, and produces a new simple polygon with a new set of weighted sites. Moreover, this resulting polygon has a particular structure that is crucial in the recursive calls of our algorithm.

Let  $\mathcal{L}_{\mathcal{S}, P}$  be the set of leaves of  $\text{VD}(\mathcal{S}, P)$ . We first notice that we can focus on a specific geodesically convex subpolygon of  $P$  to compute  $\text{VD}(\mathcal{S}, P)$ .



■ **Figure 2** a) A simple polygon  $P$  with a set  $\mathcal{S}$  of six weighted sites and their FVD. b) The polygon  $H$  being the geodesic hull of  $\mathcal{L}_{\mathcal{S},P}$  and  $\mathcal{S}$ . c) The simplification transformation allows to redefine the problem inside a simpler polygon  $G$  with a new set of weighted sites and obtain the same FVD.

► **Lemma 5.** *Let  $H$  be the geodesic hull of  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$  in  $P$ . Then, for each  $s \in \mathcal{S}$ ,  $\text{Cell}_H(s, \mathcal{S}) \subseteq \text{Cell}_P(s, \mathcal{S})$ . Moreover, the Voronoi trees of  $\text{VD}(\mathcal{S}, H)$  and  $\text{VD}(\mathcal{S}, P)$  coincide.*

**Proof.** Let  $s$  be a site of  $\mathcal{S}$  and let  $x \in H$ . Because  $H$  is a geodesically convex subset of  $P$ , and since  $x, s \in H$ , we know that  $\pi_H(x, s) = \pi_P(x, s)$ . That is, the  $w$ -distance to  $x$  from each site in  $\mathcal{S}$  is the same in  $P$  and  $H$ . Therefore, the  $\mathcal{S}$ -farthest sites of  $x$  are also preserved, which implies that  $\text{Cell}_H(s, \mathcal{S}) \subseteq \text{Cell}_P(s, \mathcal{S})$ . Because this happens for each Voronoi cell of  $\mathcal{S}$  in  $H$ , and since the leaves belong also to  $H$ , the Voronoi trees coincide. ◀

While the geodesic hull  $H$  of  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$  in  $P$  does not necessarily have lower complexity than  $P$ , it has some nice structure. We know that its boundary consists of geodesic paths that connect consecutive points in  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$  along  $\partial P$ . However, this geodesic hull  $H$  is not necessarily a simple polygon; see Figure 2. To make it simple, we need to deal with “dangling paths” as follows.

We say that a vertex of  $H$  is  $H$ -open if it is incident to the interior of  $H$ . For each  $s \in \mathcal{S}$ , let  $a_s$  be the  $H$ -open vertex of  $\text{Funnel}_P(s, \mathcal{S})$  that is geodesically closest to  $s$ . Note that all paths from  $s$  to any point in the interior of  $H$  pass through  $a_s$ . However, as long as the length of the geodesic path  $\pi_P(s, a_s)$  remains the same, the shape of this path is irrelevant. In fact, this is equivalent to giving  $a_s$  a weight such that each distance measured from  $a_s$  to points in the interior of  $H$  has an added value of  $D_w^P(s \rightsquigarrow a_s)$ .

To formalize this intuition, we define a new polygon, a new set of sites, and a new weighted distance function as follows. Let  $\mathcal{A} = \{a_s : s \in \mathcal{S}\}$  be the set of  $m$   $H$ -open vertices defined by  $\mathcal{S}$ . These vertices are our new set of sites. Let  $G(P, \mathcal{S})$  (or simply  $G$  if  $P$  and  $\mathcal{S}$  are clear from the context) be the geodesic hull of  $\mathcal{A} \cup \mathcal{L}_{\mathcal{S},P}$  in  $P$ . Note that  $G \subseteq H$  is a simple polygon by the definition of each  $a_s$  in  $\mathcal{A}$ . We define a new weight function  $w' : G \rightarrow \mathbb{R}$  so that  $w'(x) = \begin{cases} D_w^P(s \rightsquigarrow a_s) & \text{if } x = a_s \in \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$ , if  $s$  and  $a_s$  coincide, then their weight is  $w(s)$ .

With this new weight function, we can think of  $\mathcal{A}$  as a set of weighted sites in  $G$ .

► **Lemma 6.** *[\*] It holds that  $\text{VD}(\mathcal{A}, G)$  and  $\text{VD}(\mathcal{S}, P)$  have the same Voronoi trees.*

By Lemma 6, we can always transform the problem of computing the FVD of  $\mathcal{S}$  in  $P$  as follows. Recall that  $\mathcal{L}_{\mathcal{S},P}$  is the set containing each leaf of  $\text{VD}(\mathcal{S}, P)$  and that  $m = |\mathcal{S}|$ . Compute  $\text{VD}_{\partial}(\mathcal{S}, P)$  and the funnel  $\text{Funnel}_P(s, \mathcal{S})$  of each site in  $\mathcal{S}$  in total  $O(n)$  time. Because  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P} \subseteq \partial P$  and has size  $2m = O(n)$ , we can compute  $H$  the geodesic hull of  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S},P}$  in  $P$  in  $O(n)$  time using Lemma 1. After that, consider the set  $\mathcal{A}$  of  $H$ -open vertices as defined above. Again, we can compute the geodesic hull  $G$  of  $\mathcal{A} \cup \mathcal{L}_{\mathcal{S},P}$  in  $P$  in

$O(n)$  time using Lemma 1. By Lemma 6,  $\text{VD}(\mathcal{A}, G)$  and  $\text{VD}(\mathcal{S}, P)$  coincide, so we can forget about  $P$  and  $\mathcal{S}$ , and focus simply on  $G$  and  $\mathcal{A}$  to compute the FVD. We call this process the *simplification transformation*; see Figure 2. Note that the only convex vertices of  $G$  are the sites in  $\mathcal{A}$  and the leaves in  $\mathcal{L}_{\mathcal{S}, P}$ . We summarize the main properties of this simplification transformation in the following result.

► **Lemma 7.** *Let  $P$  be a simple  $n$ -gon and let  $\mathcal{S} \subseteq V(P)$  be a set of  $m \geq 3$  sites. The simplification transformation computes in  $O(n)$  time a new simple polygon  $G$  with at most  $n + m$  vertices and a new set  $\mathcal{A} \subseteq V(G)$  of  $m$  weighted sites such that (1) the Voronoi trees of  $\text{VD}(\mathcal{A}, G)$  and  $\text{VD}(\mathcal{S}, P)$  coincide, and (2) the set of convex vertices of  $G$  is exactly  $\mathcal{A} \cup \mathcal{L}_{\mathcal{S}, P}$ .*

### 3 Computing the FVD

Let  $P$  be a simple polygon and let  $\mathcal{S}$  be a set of  $m \geq 3$  weighted sites contained in  $V(P)$ . Using the simplification transformation defined in Section 2.1, we can assume without loss of generality that  $P$  is a simple polygon with at most  $n + m$  vertices, and among them, its convex vertices are exactly the sites in  $\mathcal{S}$  and the leaves of  $\mathcal{L}_{\mathcal{S}, P}$  (see Lemma 7). That is, it consists of at most  $2m$  convex vertices. If we consider consecutive vertices in  $\mathcal{S} \cup \mathcal{L}_{\mathcal{S}, P}$  along  $\partial P$ , the chain connecting them consists only of reflex vertices of  $P$ , or is a single edge. The next step explained in the following section is to randomly partition  $\mathcal{S}$ . Note that if  $m = O(1)$ , we can compute  $\text{VD}(\mathcal{S}, P)$  in  $O(n)$  time by computing their bisectors and considering their overlay. Thus, we assume that  $m$  is larger than some predefined constant.

#### 3.1 First phase: the partition

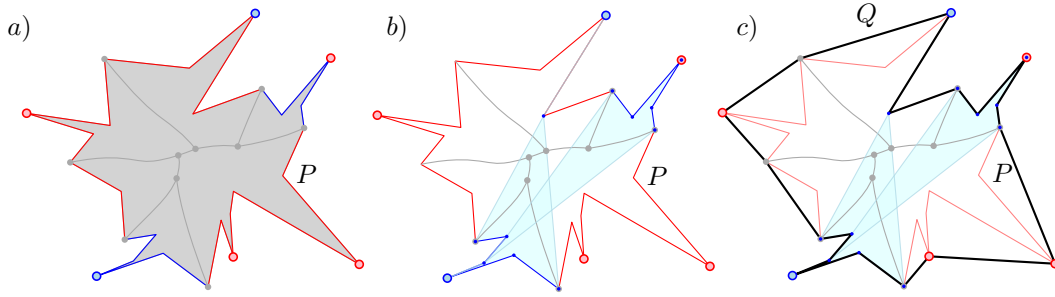
We compute in linear time  $\text{VD}_{\partial}(\mathcal{S}, P)$  using Lemma 2, and let  $\mathcal{L}_{\mathcal{S}, P}$  be the set of leaves of  $\text{VD}(\mathcal{S}, P)$ . Note that  $|\mathcal{L}_{\mathcal{S}, P}| = m$ . For each  $s \in \mathcal{S}$ , we compute the funnel  $\text{Funnel}_P(s, \mathcal{S})$ . Given a subset  $R \subseteq \mathcal{S}$ , let  $\kappa(R) = \sum_{r \in R} |\text{Funnel}_P(r, \mathcal{S})|$  be the *magnitude* of  $R$ . Lemma 4 implies that  $\kappa(\mathcal{S}) = O(n)$ , and that all these funnels can be computed in  $O(n)$  time. To be more precise, let  $\tau \geq 2$  be the constant hidden by the big  $O$  notation, i.e.,  $\kappa(\mathcal{S}) \leq \tau n$ . Next, we compute a random permutation  $\Pi$  of  $\mathcal{S}$ . Let  $0 < \alpha < 1$  be some constant to be defined later. Let  $B$  and  $R$  be a partition of  $\mathcal{S}$  such that  $B$  consists of the first  $\lfloor \frac{\alpha}{\tau} m \rfloor$  sites according to  $\Pi$ , and  $R = \mathcal{S} \setminus B$ . We refer to  $B$  and  $R$  as the sets of *blue* and *red* sites of  $\mathcal{S}$ , respectively.

► **Observation 8.** *It holds that  $\mathbb{E}[\kappa(B)] \leq \alpha n$  and  $|B| = \lfloor \frac{\alpha}{\tau} m \rfloor \leq \alpha m$ .*

We would like to recursively compute a Voronoi-like diagram of the sites in  $B$  while forgetting for a while about the red sites. Once we have this recursively computed diagram, we perform a randomized incremental construction of  $\text{VD}(\mathcal{S}, P)$  by inserting the sites of  $R$  in the random order according to permutation  $\Pi$ . In the next section we discuss the recursive call to compute a diagram for  $B$ , and later spend Section 4 detailing the insertion process.

#### 3.2 A smaller polygon

While it would be great to compute  $\text{VD}(B, P)$ , this may be too expensive as the diagram can have large complexity, and we need our recursive call to have smaller complexity (a constant fraction reduction in the size). Thus, we would not compute  $\text{VD}(B, P)$  exactly, but we will compute an “approximation” of it. Notice that we can see  $\text{VD}(B, P)$  as the upper envelope of the  $w$ -distances  $D_w^P(b \rightsquigarrow x)$ . Because these functions have a complexity that depends on the size of the polygon, we need to simplify them. To achieve this, for a site  $b \in B$ , this



■ **Figure 3** *a)* The polygon obtained from the simplification transformation, and the decomposition of its boundary into red and blue chains. *b)* The funnels of the sites in  $B$  are depicted, as well as all vertices in  $V_B$ . *c)* The polygon  $Q$  is obtained by taking a rubber band and keeping attached at all vertices in  $V_B \cup V_C$  and letting it snap.

simpler distance function will be completely accurate inside of  $\text{Cell}_P(b, \mathcal{S})$ . However, for any point  $x$  outside of  $\text{Cell}_P(b, \mathcal{S})$ , this new distance from  $s$  to  $x$  will be only upper bounded by  $D_w^P(b \rightsquigarrow x)$ . That is, distances from  $b$  can only get shorter, and only outside of  $\text{Cell}_P(b, \mathcal{S})$ .

To define these new distance functions, we define a new polygon  $Q$  of lower complexity than  $P$  (although  $P \subseteq Q$ ). Let  $V_B$  be the set consisting of all vertices of  $P$  that belong to the funnel  $\text{Funnel}_P(b, \mathcal{S})$  of some  $b \in B$ . By Observation 8, we know that the expected size of  $V_B$  is at most  $\alpha n$ . Note that we could repeat the construction of  $B$  and  $R$  an expected constant number times, until we guarantee that  $V_B \leq \alpha n$ . Let  $V_C$  be the set of convex vertices of  $P$ . By our assumption that the simplification transformation has already been applied to  $P$ , we know by Lemma 2.1 that  $V_C$  consists of the union of  $\mathcal{S}$  and  $\mathcal{L}_{\mathcal{S}, P}$ , where  $\mathcal{L}_{\mathcal{S}, P}$  is the set of leaves of  $\text{VD}(\mathcal{S}, P)$ , i.e.,  $|V_C| \leq 2m$ . Let  $Q$  be a polygon defined as follows. Imagine the boundary of  $P$  being a rubber band, and each vertex of  $V_B \cup V_C$  being a pin. By letting the rubber band free while keeping it attached at the pins, this rubber band snaps to a closed curve defining a weakly simple polygon  $Q$ ; see Figure 3.

► **Lemma 9.** *[\*] The polygon  $Q$  contains  $P$ , is weakly simple, can be computed in expected  $O(n)$  time, and has at most  $\alpha n + 4m$  vertices. Moreover, for  $x, y \in P$ , it holds that  $G^Q(x, y) \leq G^P(x, y)$ . In particular, for each  $b \in B$  and  $x \in P$ ,  $D_w^Q(b \rightsquigarrow x) \leq D_w^P(b \rightsquigarrow x)$ , and if  $x \in \text{Cell}_P(b, \mathcal{S})$ , then  $D_w^Q(b \rightsquigarrow x) = D_w^P(b \rightsquigarrow x)$ .*

**Sketch proof.** The boundary of  $Q$  can be constructed by connecting consecutive points in  $V_B \cup V_C$  by geodesics contained in the complement of  $P$ , i.e., in a domain that has  $P$  as a hole or obstacle ( $Q$  is also known as the relative hull of  $V_B \cup V_C$  in this domain). Only convex vertices of  $P$  can be in these paths, and they can be visited only twice. Therefore,  $Q$  consists of at most  $|V_B| + 2|V_C| \leq \alpha n + 4m$  vertices. ◀

Our plan is now to compute the FVD of  $B$  in the new polygon  $Q$ , and then use this as a “good” approximation of  $\text{VD}(B, P)$  in  $P$ . By “good” we mean that the red sites can be randomly inserted in this diagram, and that the result of this whole process is indeed  $\text{VD}(\mathcal{S}, P)$ . We will prove these properties in the next section, but for now, we focus on describing the recursive algorithm.

Let  $I(n, m)$  be the time to insert back the sites of  $R$  and obtain  $\text{VD}(\mathcal{S}, P)$  after having recursively computed  $\text{VD}(B, Q)$ . Because  $Q$  consists of at most  $\alpha n + 4m$  vertices by Lemma 9, and since  $|B| \leq \alpha m$  by Lemma 8, we get a recursion of the form  $T(n, m) = T(\alpha n + 4m, \alpha m) + I(n, m)$  for the running time of our algorithm. We claim that  $I(n, m) = O(n + m)$ , and we prove it in the next section. However, for  $T(n, m)$  to solve to  $O(n + m)$ , we need to look at one

more iteration of the recursion, as it can be that  $\alpha n + 4m$  is not really smaller than  $n$  if  $m$  is large. Fortunately, because  $T(\alpha n + 4m, \alpha m) = T(\alpha(\alpha n + 4m) + 4\alpha m, \alpha^2 m) + I(\alpha n + 4m, \alpha m)$ , and by our assumption on the running time of  $I(n, m)$ , we get that

$$T(n, m) = T(\alpha^2 n + 8\alpha m, \alpha^2 m) + O(n + m) + O(\alpha n + 4m + \alpha m).$$

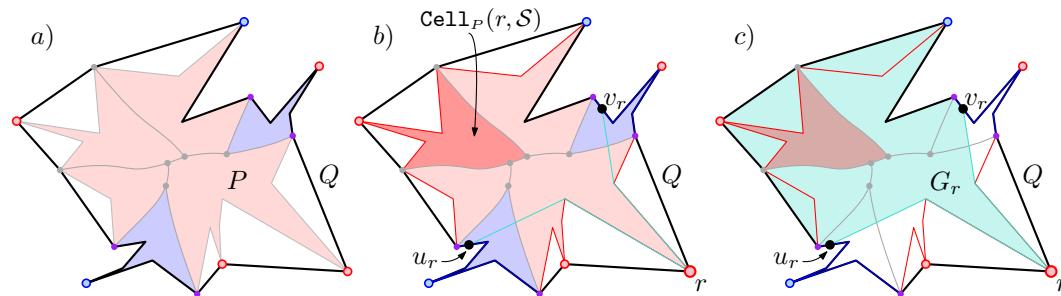
By choosing the constant  $\alpha$  sufficiently small, and since we assume that  $m \leq n$ , we can guarantee that  $T(n, m) \leq T(n/2, m/2) + O(n + m) = O(n + m)$  proving the main result of this paper. Therefore, it remains only to show that  $I(n, m)$  is indeed  $O(n + m)$ , i.e., in linear time we can insert back the red sites, and obtain the FVD  $\text{VD}(\mathcal{S}, P)$  from the recursively computed diagram of  $\text{VD}(B, Q)$ .

### 3.3 Preprocessing the red sites

Before going into the insertion process of the sites in  $R$ , we need to finish some preprocessing on them. To be able to insert these sites efficiently, we need to have a representation of the  $w$ -distance of each  $r \in R$  defined on a sufficiently large superset of  $\text{Cell}_P(r, \mathcal{S})$ .

Note that we cannot define these distance functions in the entire polygon, otherwise we are spending already too much time and space. On the other hand, if its representation is too narrow, then during the insertion it might be that the distance information is insufficient.

Recall that  $\mathcal{L}_{\mathcal{S}, P}$  denotes the set of leaves of  $\text{VD}(\mathcal{S}, P)$ . Color the leaves in  $\mathcal{L}_{\mathcal{S}, P}$  purple if they are incident to the Voronoi cell of a site in  $B$  and a site in  $R$ ; see Figure 4.



■ **Figure 4** a) The coloring of the purple leaves of  $\mathcal{L}_{\mathcal{S}, P}$ . b) For a site  $r \in R$ , the construction of  $u_r$  and  $v_r$  lying inside blue Voronoi cells. c) The polygon  $G_r$  where we compute the SPM of  $r$ .

Recall that given a polygon  $K$  and two points  $x$  and  $y$  on  $\partial K$ ,  $\partial K(x, y)$  denotes the polygonal chain that starts at  $x$  and follows the boundary of  $K$  clockwise until reaching  $y$ . For each  $r \in R$ , let  $u_r$  and  $v_r$  respectively be the first purple leaves of  $\mathcal{L}_{\mathcal{S}, P}$  reached from any point in  $\text{bCell}_P(r, \mathcal{S})$  when walking counterclockwise and clockwise along  $\partial P$ . We then move  $u_r$  and  $v_r$  slightly clockwise and counterclockwise, respectively, so that they both sit inside of a blue Voronoi cell. Moreover, we know that  $\text{bCell}_P(r, \mathcal{S})$  is contained in the interior of the path  $\partial P(u_r, v_r)$ . We define a new polygon where we define the  $w$ -function of  $r$  as follows. Because both  $u_r$  and  $v_r$  lie on  $\partial P$  and on Voronoi cells of blue sites, we know that  $u_r$  and  $v_r$  are both on  $\partial Q$ . Let  $G_r$  be a new weakly simple polygon bounded by the paths  $\pi_P(r, u_r)$ ,  $\partial Q(u_r, v_r)$  and  $\pi_P(v_r, r)$ . Notice however that the paths  $\pi_P(r, u_r)$  and  $\pi_P(v_r, r)$ , called the *walls*, are defined within the polygon  $P$ , while the other path bounding  $G_r$  is contained in  $\partial Q$ . Since  $P \subset Q$ , we know that  $G_r \subseteq Q$ . Moreover, the funnel  $\text{Funnel}_P(r \rightarrow \partial P(u_r, v_r))$  is contained in  $G_r$ ; see Figure 4.

## 12:10 Geodesic Farthest-Point Voronoi Diagrams

► **Lemma 10.** [\*] Given  $r \in R$ , it holds that  $\text{Cell}_P(r, \mathcal{S}) \subseteq G_r$ . Moreover, it holds that

$$\mathbb{E} \left[ \sum_{r \in R} |G_r| \right] = O(n) \text{ and } \bigcup_{r \in R} G_r = Q.$$

The last technical detail is the structure used to store the  $w$ -distances from the sites in  $R$ . Let  $s \in \mathcal{S}$  and let  $H \subseteq P$  be a subpolygon such that  $s \in H$ . The *shortest-path map* (or *SPM* for short) of  $s$  in  $H$  is a subdivision of  $H$  into triangles such that the geodesic path to all the points in one triangle has the same combinatorial structure. By precomputing the geodesic distance to each vertex of  $H$ , we get a constant-sized representation of the  $w$ -distance from  $s$  inside each triangle (for more information on shortest-path maps refer to [11]). We know also that the SPM of  $s$  in  $H$  can be computed in  $O(|H|)$  time [8, 11].

Using these SPM's, we describe our  $w$ -distances as follows. For each  $r \in R$ , we compute the SPM of  $r$  in  $G_r$ . Because the complexity of this SPM is  $O(|G_r|)$ , we conclude that the total expected complexity of all these SPM's is  $\mathbb{E} [\sum_{r \in R} |G_r|] = O(n)$  by Lemma 10.

### 4 Inserting back the red sites

After computing  $\text{VD}(B, Q)$  recursively, we would like to start the randomized incremental construction of sites of  $R$ . But first, we should specify how we store  $\text{VD}(B, Q)$ .

Using the SPM's, we introduce the *refined FVD* of  $B$  in  $Q$ . This refined FVD is a decomposition of  $Q$  into constant-size cells defined as follows: For each site  $b \in B$ , the Voronoi cell  $\text{Cell}_Q(b, B)$  is subdivided by the defining triangles of the SPM of  $b$  in  $Q$ . That is, we take the intersection of each defining triangle  $\Delta$  of the SPM of  $b$  and intersect it with the Voronoi cell of  $b$  to obtain a *refined triangle*. As usual, for each point in a refined triangle, the geodesic distance is measured from its apex  $a$  and added with  $D_w^P(b \rightsquigarrow a)$ . Thus, each refined triangle and its distance function can be described with  $O(1)$  space. We say that  $b$  *owns* these refined triangles. In other words, we have a way to describe the upper envelope of the  $w$ -distances of the sites in  $B$  within  $Q$  using a collection of constant-size refined triangles.

Assume inductively that  $\text{VD}(B, Q)$  is represented as a refined FVD as described above. That is, for each site  $b$  of  $B$ , there is a collection of refined triangles owned by  $b$  which cover the entire Voronoi cell  $\text{Cell}_Q(b, B)$ . Let  $f_b : Q \rightarrow \mathbb{R}$  such that  $f_b(x) = D_w^Q(b \rightsquigarrow x)$  for each  $x \in Q$ , i.e.,  $f_b$  is the function that maps each point to its  $w$ -distance to  $b$  in  $Q$ . Note that the refined triangles of  $b$  in the refined FVD of  $\text{VD}(B, Q)$  provide a representation of this  $w$ -distance inside  $\text{Cell}_Q(b, B)$ . We say that the geodesic path  $\pi_Q(b, x)$  is the *witness path* of the value of  $f_b(x)$ .

► **Observation 11.** Let  $b \in B$ . Given  $x \in P$ , it holds that  $f_b(x) \leq D_w^P(b \rightsquigarrow x)$ . Moreover, if  $x \in \text{Cell}_P(b, \mathcal{S})$ , then  $f_b(x) = D_w^P(b \rightsquigarrow x)$ .

Let  $r \in R$  and recall that  $G_r$  is the polygon associated with  $r$  defined in Section 3.1. As a preprocessing, we have computed the SPM of  $r$  within  $G_r$ . We define a function  $f_r : G_r \rightarrow \mathbb{R}$  that encodes the  $w$ -distances from  $r$  with respect to the polygon  $G_r$ , instead of  $Q$ . That is,  $f_r(x) = D_w^{G_r}(x \rightsquigarrow r)$  for each  $x \in G_r$ . In this case we say that  $\pi_{G_r}(r, x)$  is a *witness path* of the value of  $f_r(x)$ . Note that by Lemma 10, the functions  $f_r$  jointly cover polygon  $Q$ .

► **Lemma 12.** [\*] Let  $r \in R$ . Given  $x \in P$ , it holds that  $f_r(x) \leq D_w^P(r \rightsquigarrow x)$ . Moreover, if  $x \in \text{Cell}_P(r, \mathcal{S})$  and the path  $\pi_P(r, x)$  contains no point of  $\text{bCell}_P(r, \mathcal{S})$  other than its endpoints, then  $f_r(x) = D_w^P(r \rightsquigarrow x)$ .



Note that for each site of  $R$ , we have considered their  $w$ -distances inside of  $G_r$ , while for the sites in  $B$ , their  $w$ -distances are with respect to  $Q$ . Therefore, in the intermediate steps of our incremental construction, we will not have the FVD of the sites, but some Voronoi-like structure.

## 4.1 The envelope

Note that  $\text{VD}(B, Q)$  represents already the upper envelope of the functions  $f_b$  for the sites in  $B$ . We would like to complete this envelope by incrementally inserting the functions  $f_r$  for the sites in  $R$ . To deal with these upper envelopes, we introduce some definitions.

Consider the order of the sites of  $R$  according to the random permutation  $\Pi$  used to construct  $B$  and  $R$ . Let  $\mathcal{S}_0 = B$  and for each  $1 \leq i \leq |R|$ , let  $\mathcal{S}_i$  be the set consisting of  $B$  and the first  $i$  red sites according to the permutation  $\Pi$ . This is the order that we use for our randomized incremental construction. That is, on each insertion step we would like to maintain a Voronoi-like structure for the sites in  $\mathcal{S}_i$ .

Let  $s$  be a site of  $\mathcal{S}_i$ . Given a point  $x$  of  $Q$ , we say that  $x$  is  $i$ -dominated by  $s$  if  $f_s(x) \geq f_{s'}(x)$  for all  $s' \in \mathcal{S}_i$ . Notice that if a point  $x$  is  $i$ -dominated, then the witness path of  $f_s(x)$  must be defined.

► **Lemma 13.** [\*] *Let  $s$  be a site of  $\mathcal{S}$ , and let  $x$  be a point that is  $i$ -dominated by  $s$ . If  $z \in Q$  is a point that lies on the witness path of  $f_s(z)$ , then  $z$  is  $i$ -dominated by  $s$ .*

Let  $x$  be a point that is  $i$ -dominated by  $s$ . Extend the last segment of the witness path of  $f_s(x)$  until it touches the boundary of  $Q$  at a point  $x^*$ . We say that  $x^*$  is the  $s$ -shadow of  $x$ . A direct consequence of Lemma 13 is the following result.

► **Corollary 14.** *Let  $s \in \mathcal{S}_i$ . If  $x$  is  $i$ -dominated by  $s$ , then its  $s$ -shadow is also  $i$ -dominated by  $s$  and lies on  $\partial Q$ .*

The following result is crucial to guarantee the resulting structure after the incremental construction coincides with the desired FVD of  $\mathcal{S}$ .

► **Lemma 15.** [\*] *For each  $0 \leq i \leq |R|$  and for each site  $s \in \mathcal{S}_i$ , each point in the Voronoi cell  $\text{Cell}_P(s, \mathcal{S})$  is  $i$ -dominated by  $s$ .*

## 4.2 The insertion process

Let  $r$  be the  $i$ -th site of  $R$  inserted in our randomized incremental construction. To simplify our incremental construction, instead of constructing the entire set of points that are  $i$ -dominated by  $r$ , which might contain several connected components, we focus exclusively on constructing the connected component containing  $\text{Cell}_P(r, \mathcal{S})$ . This simplifies the structure of the upper envelope, and helps us to prove a bound on its complexity.

We define the *envelope-graph* of  $\mathcal{S}_i$  recursively. For the base case  $i = 0$ , the envelope-graph of  $\mathcal{S}_0$  is simply the Voronoi-tree of  $\text{VD}(B, Q)$ . This envelope-graph induces a decomposition of  $Q$  into  $0$ -patches. The  $0$ -patch of each site  $s \in \mathcal{S}_0$  is the connected component in this decomposition that contains  $\text{Cell}_P(s, \mathcal{S})$ .

Given the envelope-graph of  $\mathcal{S}_{i-1}$ , the envelope-graph of  $\mathcal{S}_i$  is defined as follows. We consider the set of all points of  $Q$  that are  $i$ -dominated by  $r$  and the connected components that they induce. The  $i$ -patch of  $r$  is the connected component that contains  $\text{Cell}_P(r, \mathcal{S})$  induced by these points. The envelope-graph of  $\mathcal{S}_i$  is then obtained by adding to it the boundary of the  $i$ -patch of  $r$ , and removing everything inside it. In this way, the  $(i-1)$ -patches of the envelope graph of  $\mathcal{S}_{i-1}$  might shrink. However, Lemma 15 guarantees that for each site  $s \in \mathcal{S}_i$ , the Voronoi cell  $\text{Cell}_P(s, \mathcal{S})$  is  $i$ -dominated by  $s$ . Therefore,  $\text{Cell}_P(s, \mathcal{S})$  is still contained in the  $i$ -patch of  $s$ , i.e., the  $i$ -patch of  $s$  is non-empty.

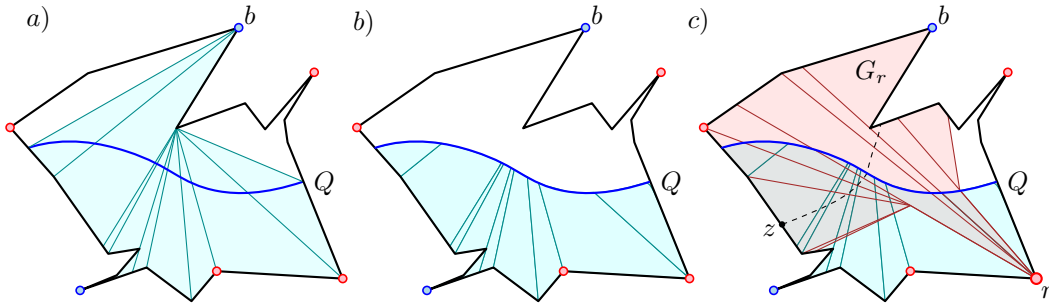
► **Lemma 16.** [\*] *The envelope-graph of  $\mathcal{S}_i$  is a tree with at most  $2|\mathcal{S}_i|$  leaves lying on the boundary of  $Q$ .*

**Proof sketch.** The proof is by induction, with  $\text{vd}(B, Q)$  as base case. When inserting the  $i$ -th site  $r \in R$  into the envelope-graph of  $\mathcal{S}_{i-1}$ , the  $i$ -patch of  $r$  is connected and intersects  $\partial Q$  in a single connected component. By adding this  $i$ -patch, the tree structure is preserved. ◀

► **Lemma 17.** [\*] *The  $i$ -patch of  $r$  is contained in  $G_r$  and does not intersect the walls of  $G_r$ .*

### 4.3 Algorithmic description

We proceed now to describe algorithmically how to carry on the incremental construction described above, and construct the envelope-graph of  $\mathcal{S}_i$ . Our algorithm starts with the refined FVD of  $\text{vd}(B, Q)$ , and on each round constructs the boundary of the  $i$ -patch of a new site of  $R$ . In addition to our envelope-graph, we maintain a set of refined triangles that cover each  $i$ -patch in the same way that they cover the Voronoi cells in the refined FVD; see Figure 5. We call this representation the *refined envelope* of  $\mathcal{S}_i$ . We assume inductively that the envelope-graph of  $\mathcal{S}_i$  is stored as a refined envelope. For the base case this holds as we assume that we have at hand the refined FVD of  $\text{vd}(B, Q)$ . In addition, we maintain the invariant that for each vertex  $v$  of  $Q$ , we know the site whose  $i$ -patch contains  $v$ . Moreover, we assume also that we have a pointer to the refined triangle of this  $i$ -patch that contains  $v$ .



■ **Figure 5** a) a site  $b \in B$  and the defining triangles of  $f_b$ . b) The refined triangles obtained by intersecting with the 0-patch of  $b$ . c) The insertion of the red site  $r$  and the update of the envelope.

For each  $b \in B$ , let  $\mu_b$  be the set of refined triangles that belong to  $b$ . For a site  $r \in R$ , let  $\mu_r$  denote the set of triangles used to describe  $f_r$ , i.e., the set of triangles in the SPM of  $r$  inside of  $G_r$ . Thus regardless of the case  $\mu_s$  denotes a set of triangles (and their associated distance function) *owned* by  $s$ .

► **Lemma 18.** [\*] *Let  $r$  be the  $i$ -th site of  $R$  inserted in our randomized incremental construction. The  $i$ -patch of  $r$  can be computed in  $O(M_r + D_r + |\mu_r|)$  time, where  $M_r$  is the size of the  $i$ -patch of  $r$ , and  $D_r$  is the number of arcs of the envelope-graph of  $\mathcal{S}_{i-1}$  that disappear. Moreover, the refined envelope of  $\mathcal{S}_i$  can be obtained within the same time from that of  $\mathcal{S}_{i-1}$ .*

**Proof Sketch.** To compute the  $i$ -patch of  $r$ , the first step is to find a point lying on its boundary, which can be done by locating the leaves of  $\mathcal{L}_{\mathcal{S}, P}$  bounding  $\text{bCell}_P(r, \mathcal{S})$ . Next, we walk along  $\partial Q$  until finding an endpoint  $z$  of the intersection of  $\partial Q$  with the  $i$ -patch of  $r$ . Finally, we trace the boundary of the  $i$ -patch of  $r$  inside of  $Q$  in the standard way used in the RIC of Euclidean Voronoi diagrams. That is, by walking along the overlay of the triangles in  $\mu_r$  and the refined triangles of the refined envelope of  $\mathcal{S}_{i-1}$ , and constructing each arc of the  $i$ -patch of  $r$  at a time. The insertion time is then proportional to the size of the  $i$ -patch. ◀



The *complexity* of the envelope-graph of  $\mathcal{S}_i$  is the number of vertices and arcs defining it.

► **Lemma 19.** [\*] *The expected complexity of the envelope-graph of  $\mathcal{S}_i$  is  $O(n)$ .*

**Proof sketch.** Recall that for  $b \in B$ ,  $\mu_b$  is the set of refined triangles that belong to  $b$ , and for  $r \in R$ ,  $\mu_r$  is the set of triangles used to describe  $f_r$  inside of  $G_r$ . Since  $\text{VD}(B, Q)$  is a FVD of  $B$  in  $Q$  and by Lemma 10, we know that  $\mathbb{E}[\sum_{s \in \mathcal{S}} |\mu_s|] = O(n)$ . Using a charging argument, we prove that the complexity of the envelope-graph of  $\mathcal{S}_i$  is at most  $\sum_{s \in \mathcal{S}} |\mu_s|$ . ◀

We are now ready to combine these lemmas into the main result of this section.

► **Theorem 20.** [\*] *The envelope-graph and refined envelope of  $\mathcal{S}_{|R|}$  can be computed in expected  $O(n)$  time. Moreover, for each  $s \in \mathcal{S}$ , the envelope-graph of  $\mathcal{S}_{|R|}$  coincides with the Voronoi tree of  $\text{VD}(\mathcal{S}, P)$ .*

**Proof Sketch.** By Lemma 15, the  $|R|$ -patch of each site  $s$  of  $\mathcal{S}$  contains its corresponding Voronoi cell  $\text{Cell}_P(s, \mathcal{S})$ . Because the union of these Voronoi cells covers  $P$ , we conclude that the  $|R|$ -patch of  $s$  and  $\text{Cell}_P(s, \mathcal{S})$  coincide inside  $P$  for each  $s \in \mathcal{S}$ . That is, the envelope-graph of  $\mathcal{S}_{|R|}$  coincides with the Voronoi tree of  $\text{VD}(\mathcal{S}, P)$ .

By Lemma 18, the time needed to insert all sites of  $R$ , is  $O(\sum_{r \in R} (M_r + D_r + |\mu_r|))$ . Using backwards analysis, we show that  $\mathbb{E}[M_r] = O(n/|S|)$ . Therefore,  $\mathbb{E}[\sum_{r \in R} M_r] = O(n)$ . Moreover, Lemma 10 implies that  $\sum_{r \in R} \mu_r = O(n)$ , and since an arc is destroyed only once, we can charge this cost to the creation of the arc. Putting everything together, we conclude that the expected time needed to insert all red sites is  $O(n)$ . ◀

We are ready to state our main result. As mentioned in Section 3.2, the running time of our algorithm is  $T(n, m) \leq T(n/2, m/2) + I(n, m)$ , where  $I(n, m)$  is the time to insert the red sites. By Theorem 20, and by our assumption that  $m \leq n$ ,  $I(n, m) = O(n + m)$ . Therefore, by solving the recurrence we obtain the following result.

► **Theorem 21.** *Let  $P$  be a simple polygon and let  $\mathcal{S}$  be a set of  $m \geq 3$  weighted sites contained in  $V(P)$ . We can compute the FVD of  $\mathcal{S}$  in  $P$  in expected  $O(n + m)$  time.*

---

## References

- 1 Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou Carufel, Matias Korman, and Eunjin Oh. A Linear-Time Algorithm for the Geodesic Center of a Simple Polygon. *Discrete & Computational Geometry*, 56(4):836–859, December 2016. doi:10.1007/s00454-016-9796-0.
- 2 Boris Aronov. On the geodesic Voronoi diagram of point sites in a simple polygon. *Algorithmica*, 4(1-4):109–140, 1989.
- 3 Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(1):217–255, 1993.
- 4 T. Asano and G.T. Toussaint. Computing the geodesic center of a simple polygon. Technical Report SOCS-85.32, McGill University, 1985.
- 5 Luis Barba. Geodesic farthest-point Voronoi diagram in linear time. *CoRR*, abs/1809.01481, 2018. arXiv:1809.01481.
- 6 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proceedings of the twenty-sixth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1655–1670. SIAM, 2014.
- 7 Bernard Chazelle. A theorem on polygon cutting with applications. In *Proceedings of FOCS*, pages 339–349, 1982. doi:10.1109/SFCS.1982.58.
- 8 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(1):485–524, 1991.

- 9 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.
- 10 Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.
- 11 Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.
- 12 John Hershberger and Subhash Suri. Matrix Searching with the Shortest-Path Metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.
- 13 Chih-Hung Liu. A Nearly Optimal Algorithm for the Geodesic Voronoi Diagram of Points in a Simple Polygon. In *LIPICs-Leibniz International Proceedings in Informatics*, volume 99. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 14 J. S. B. Mitchell. Geometric Shortest Paths and Network Optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.
- 15 Eunjin Oh. Optimal Algorithm for Geodesic Nearest-point Voronoi Diagrams. In *To appear in the Proceedings of the 31st annual ACM-SIAM Symposium on Discrete Algorithms*, page TBD, 2019.
- 16 Eunjin Oh and Hee-Kap Ahn. Voronoi diagrams for a moderate-sized point-set in a simple polygon. In *LIPICs-Leibniz International Proceedings in Informatics*, volume 77. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
- 17 Eunjin Oh, Sang Won Bae, and Hee-Kap Ahn. Computing a geodesic two-center of points in a simple polygon. In *Latin American Symposium on Theoretical Informatics*, pages 646–658. Springer, 2016.
- 18 Eunjin Oh, Luis Barba, and Hee-Kap Ahn. The farthest-point geodesic Voronoi diagram of points on the boundary of a simple polygon. In *LIPICs-Leibniz International Proceedings in Informatics*, volume 51. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
- 19 Evanthia Papadopoulou.  $k$ -Pairs Non-Crossing Shortest Paths in a Simple Polygon. *International Journal of Computational Geometry and Applications*, 9(6):533–552, 1999.
- 20 Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(1):611–626, 1989.
- 21 Subhash Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39(2):220–235, 1989.

# Upward Book Embeddings of *st*-Graphs

**Carla Binucci**

Università degli Studi di Perugia, Perugia, Italy  
carla.binucci@unipg.it

**Giordano Da Lozzo**

Roma Tre University, Rome, Italy  
giordano.dalozzo@uniroma3.it

**Emilio Di Giacomo**

Università degli Studi di Perugia, Perugia, Italy  
emilio.digiaco@unipg.it

**Walter Didimo**

Università degli Studi di Perugia, Perugia, Italy  
walter.didimo@unipg.it

**Tamara Mchedlidze**

Karlsruhe Institute of Technology, Karlsruhe, Germany  
mched@iti.uka.de

**Maurizio Patrignani**

Roma Tre University, Rome, Italy  
maurizio.patrignani@uniroma3.it

---

## Abstract

We study *k*-page upward book embeddings (*k*UBEs) of *st*-graphs, that is, book embeddings of single-source single-sink directed acyclic graphs on *k* pages with the additional requirement that the vertices of the graph appear in a topological ordering along the spine of the book. We show that testing whether a graph admits a *k*UBE is NP-complete for  $k \geq 3$ . A hardness result for this problem was previously known only for  $k = 6$  [Heath and Pemmaraju, 1999]. Motivated by this negative result, we focus our attention on  $k = 2$ . On the algorithmic side, we present polynomial-time algorithms for testing the existence of 2UBEs of planar *st*-graphs with branchwidth  $\beta$  and of plane *st*-graphs whose faces have a special structure. These algorithms run in  $O(f(\beta) \cdot n + n^3)$  time and  $O(n)$  time, respectively, where  $f$  is a singly-exponential function on  $\beta$ . Moreover, on the combinatorial side, we present two notable families of plane *st*-graphs that always admit an embedding-preserving 2UBE.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Mathematics of computing  $\rightarrow$  Graph algorithms

**Keywords and phrases** Upward Book Embeddings, *st*-Graphs, SPQR-trees, Branchwidth, Sphere-cut Decomposition

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.13

**Related Version** Details for the omitted and sketched proofs can be found in the full version of the paper [23], which is available at <https://arxiv.org/abs/1903.07966>.

**Funding** This work was supported in part by project “Algoritmi e sistemi di analisi visuale di reti complesse e di grandi dimensioni” – Ricerca di Base 2018, Dipartimento di Ingegneria dell’Università degli Studi di Perugia (Binucci, Di Giacomo, and Didimo), and in part by MIUR Project “MODE” under PRIN 20157EFM5C, by MIUR Project “AHeAD” under PRIN 20174LF3T8, by MIUR-DAAD JMP N° 34120, by H2020-MSCA-RISE project 734922 – “CONNECT”, and by Roma Tre University Azione 4 Project “GeoView” (Da Lozzo and Patrignani).

**Acknowledgements** This research began at the Bertinoro Workshop on Graph Drawing 2018.



© Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani; licensed under Creative Commons License CC-BY

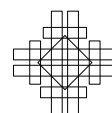
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 13; pp. 13:1–13:22



Leibniz International Proceedings in Informatics

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



## 1 Introduction

A  $k$ -page book embedding  $\langle \pi, \sigma \rangle$  of an undirected graph  $G = (V, E)$  consists of a vertex ordering  $\pi : V \leftrightarrow \{1, 2, \dots, |V|\}$  and of an assignment  $\sigma : E \rightarrow \{1, \dots, k\}$  of the edges of  $G$  to one of  $k$  sets, called *pages*, so that for any two edges  $(a, b)$  and  $(c, d)$  in the same page, with  $\pi(a) < \pi(b)$  and  $\pi(c) < \pi(d)$ , we have neither  $\pi(a) < \pi(c) < \pi(b) < \pi(d)$  nor  $\pi(c) < \pi(a) < \pi(d) < \pi(b)$ . From a geometric perspective, a  $k$ -page book embedding can be associated with a *canonical drawing*  $\Gamma(\pi, \sigma)$  of  $G$  where the  $k$  pages correspond to  $k$  half-planes sharing a vertical line, called the *spine*. Each vertex  $v$  is a point on the spine with  $y$ -coordinate  $\pi(v)$ ; each edge  $e$  is a circular arc on the  $\sigma(e)$ -th page, and the edges in the same page do not cross.

For  $k$ -page book embeddings of directed graphs (digraphs), a typical requirement is that all the edges are oriented in the upward direction. This implies that  $G$  is acyclic and that all the vertices appear along the spine in a topological ordering. This type of book embedding for digraphs is called an *upward  $k$ -page book embedding* of  $G$  (for short, *kUBE*). Note that, when  $k = 2$  and the two pages are coplanar, drawing  $\Gamma(\pi, \sigma)$  is an *upward planar drawing* of  $G$ , i.e., a planar drawing where all the edges monotonically increase in the upward direction. The study of upward planar drawings is a most prolific topic in the theory of graph visualization [6, 7, 20, 21, 25, 26, 28, 31, 33, 35, 51, 69].

The *page number* of a (di)graph  $G$  (also called *book thickness*) is the minimum number  $k$  such that  $G$  admits a (upward)  $k$ -page book embedding. Computing the page number of directed and undirected graphs is a widely studied problem, which finds applications in a variety of domains, including VLSI design, fault-tolerant processing, parallel process scheduling, sorting networks, parallel matrix computations [29, 53, 68], computational origami [2], and graph drawing [22, 37, 52, 76]. See [42] for additional references.

**Book embeddings of undirected graphs.** Seminal results on book embeddings of undirected graphs are described in the paper of Bernhart and Kainen [19]. They prove that the graphs with page number one are exactly the outerplanar graphs, while graphs with page number two are the sub-Hamiltonian graphs. This second result implies that it is NP-complete to decide whether a graph admits a 2-page book embedding [75]. Yannakakis [77] proved that every planar graph has a 4-page book embedding, while the fascinating question whether the page number of planar graphs can be reduced to three is still open. The aforementioned works have inspired several papers about the page number of specific families of undirected graphs (e.g., [16, 18, 29, 46]) and about the relationship between the page number and other graph parameters (e.g., [43, 50, 60, 61]). Different authors studied constrained versions of  $k$ -page book embeddings where either the vertex ordering  $\pi$  is (partially) fixed [8, 30, 62, 73, 74] or the page assignment  $\sigma$  for the edges is given [10, 9, 11, 57]. Relaxed versions of book embeddings where edge crossings are allowed (called  *$k$ -page drawings*) or where edges can cross the spine (called *topological book embeddings*) have also been considered (e.g., [1, 14, 24, 27, 36, 44, 45]). Finally, 2-page (topological) book embeddings find applications to point-set embedding and universal point set (e.g., [12, 13, 39, 40, 47, 59]).

**Book embeddings of directed graphs.** As for undirected graphs, there are many papers devoted to the study of upper and lower bounds on the page number of directed graphs. Heath et al. [56] show that directed trees and unicyclic digraphs have page number one and two, respectively. Alzohairi and Rival [4], and later Di Giacomo et al. [37] with an improved linear-time construction, show that series-parallel digraphs have page number two. Mchedlidze and Symvonis [63] generalize this result and prove that  $N$ -free upward planar

digraphs, which contain series-parallel digraphs, also have page number two (a digraph is upward planar if it admits an upward planar drawing). Frati et al. [49] give several conditions under which upward planar triangulations have bounded page number. Overall, the question asked by Nowakowski and Parker [66] almost 30 years ago, of whether the page number of upward planar digraphs is bounded, remains open. Several works study the page number of acyclic digraphs in terms of posets, i.e., the page number of their Hasse diagram (e.g., [3, 66]).

About the lower bounds, Nowakowski and Parker [66] give an example of a *planar st-graph* that requires three pages for an upward book embedding (see Fig. 9a). A planar *st-graph* is an upward planar digraph with a single source  $s$  and a single sink  $t$ . Hung [58] shows an upward planar digraph with page number four, while Heath and Pemmaraju [54] describe an acyclic planar digraph (which is not upward planar) requiring  $\lfloor n/2 \rfloor$  pages. Syslo [72] provides a lower bound on the page number of a poset in terms of its bump number.

Besides the study of upper and lower bounds on the page number of digraphs, several papers concentrate on the design of testing algorithms for the existence of  $k$ UBEs. The problem is NP-complete for  $k = 6$  [55]. For  $k = 2$ , Mchedlidze and Symvonis [65] give linear-time testing algorithms for outerplanar and planar triangulated *st-graphs*. An  $O(w^2n^w)$ -time testing algorithm for 2UBEs of planar *st-graphs* whose width is  $w$  is given in [63], where the *width* is the minimum number of directed paths that cover all the vertices. Heath and Pemmaraju [55] describe a linear-time algorithm to recognize digraphs that admit 1UBEs.

Finally, as for the undirected case, constrained or relaxed variants of  $k$ UBEs for digraphs are studied [2, 38, 52], as well as applications to the point-set embedding problem [37, 52].

**Contribution.** Our paper is motivated by the gap present in the literature about the computation of upward book embeddings of digraphs: Polynomial-time algorithms are known only for one page or for two pages and subclasses of planar digraphs, while NP-completeness is known only for exactly 6 pages. We shrink this gap and address the research direction proposed by Heath and Pemmaraju [55]: Identification of graph classes for which the existence of  $k$ UBEs can be solved efficiently. Our results are as follows:

- We prove that testing whether a digraph  $G$  admits a  $k$ UBE is NP-complete for every  $k \geq 3$ , even if  $G$  is an *st-graph* (Section 3). An analogous result was previously known only for the constrained version in which the page assignment is given [2].
- We describe another meaningful subclass of upward planar digraphs that admit a 2UBE (Section 4). This class is structurally different from the  $N$ -free upward planar digraphs, the largest class of upward 2-page book embeddable digraphs previously known.
- We give algorithms to test the existence of a 2UBE for notable families of planar *st-graphs*. First, we give a linear-time algorithm for plane *st-graphs* whose faces have a special structure (Section 5). Then, we describe an  $O(f(\beta) \cdot n + n^3)$ -time algorithm for  $n$ -vertex planar *st-graphs* of branchwidth  $\beta$ , where  $f$  is a singly-exponential function (Section 6). The algorithm works for both variable and fixed embedding. This result also implies a sub-exponential-time algorithm for general planar *st-graphs*.

## 2 Preliminaries

We assume familiarity with basic definitions on graph connectivity and planarity (see [15, 23]). We only consider (di)graphs without loops and multiple edges, and we denote by  $V(G)$  and  $E(G)$  the sets of vertices and edges of a (di)graph  $G$ .

A digraph  $G$  is a *planar st-graph* if and only if: (i) it is acyclic; (ii) it has a single source  $s$  and a single sink  $t$ ; and (iii) it admits a planar embedding  $\mathcal{E}$  with  $s$  and  $t$  on the outer face. A graph  $G$  together with  $\mathcal{E}$  is a *planar embedded st-graph*, also called a *plane st-graph*.

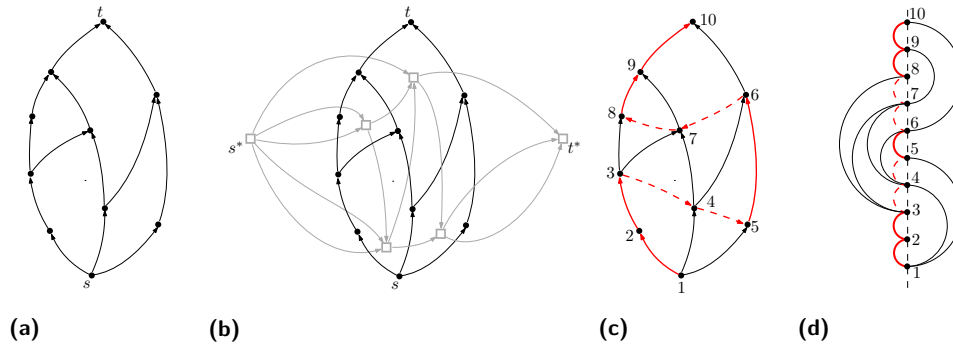
Let  $G$  be a plane  $st$ -graph and let  $e = (u, v)$  be an edge of  $G$ . The *left face* (resp. *right face*) of  $e$  is the face to the left (resp. right) of  $e$  while moving from  $u$  to  $v$ . The boundary of every face  $f$  of  $G$  consists of two directed paths  $p_l$  and  $p_r$  from a common source  $s_f$  to a common sink  $t_f$ . The paths  $p_l$  and  $p_r$  are the *left path* and the *right path* of  $f$ , respectively. The vertices  $s_f$  and  $t_f$  are the *source* and the *sink* of  $f$ , respectively. If  $f$  is the outer face,  $p_l$  (resp.  $p_r$ ) consists of the edges for which  $f$  is the left face (resp. right face); in this case  $p_l$  and  $p_r$  are also called the *left boundary* and the *right boundary* of  $G$ , respectively. If  $f$  is an internal face,  $p_l$  (resp.  $p_r$ ) consists of the edges for which  $f$  is the right face (resp. left face).

The *dual graph*  $G^*$  of a plane  $st$ -graph  $G$  is a plane  $st$ -graph (possibly with multiple edges) such that: (i)  $G^*$  has a vertex associated with each internal face of  $G$  and two vertices  $s^*$  and  $t^*$  associated with the outer face of  $G$ , that are the source and the sink of  $G^*$ , respectively; (ii) for each internal edge  $e$  of  $G$ ,  $G^*$  has a dual edge from the left to the right face of  $e$ ; (iii) for each edge  $e$  in the left boundary of  $G$ , there is an edge from  $s^*$  to the right face of  $e$ ; (v) for each edge  $e$  in the right boundary of  $G$ , there is an edge from the left face of  $e$  to  $t^*$ .

Consider a planar  $st$ -graph  $G$  and let  $\overline{G}$  be a planar  $st$ -graph obtained by augmenting  $G$  with directed edges in such a way that it contains a directed Hamiltonian  $st$ -path  $P_{\overline{G}}$ . The graph  $\overline{G}$  is an *HP-completion* of  $G$ . Consider now a plane  $st$ -graph  $G$  and let  $\mathcal{E}$  be a planar embedding of  $G$ . Let  $\overline{\mathcal{E}}$  be an embedded HP-completion of  $G$  whose embedding  $\overline{\mathcal{E}}$  is such that its restriction to  $G$  is  $\mathcal{E}$ . We say that  $\overline{\mathcal{E}}$  is an *embedding-preserving HP-completion* of  $G$ .

Bernhart and Kainen [19] prove that an undirected planar graph admits a 2-page book embedding if and only if it is *sub-Hamiltonian*, i.e., it can be made Hamiltonian by adding edges while preserving its planarity. Theorem 1 is an immediate consequence of the result in [19] for planar digraphs (see also Fig. 1); when we say that a 2UBE  $\langle \pi, \sigma \rangle$  is *embedding-preserving* we mean that the drawing  $\Gamma(\pi, \sigma)$  preserves the planar embedding of  $G$ .

► **Theorem 1.** *A planar (plane) st-graph  $G$  admits a (embedding-preserving) 2UBE  $\langle \pi, \sigma \rangle$  if and only if  $G$  admits a (embedding-preserving) HP-completion  $\overline{G}$ . Also, the order  $\pi$  coincides with the order of the vertices along  $P_{\overline{G}}$ .*



■ **Figure 1** (a) A plane  $st$ -graph  $G$ . (b) The dual of  $G$  is shown in gray. (c) An embedding-preserving HP-completion of  $G$ . (d) An embedding-preserving 2UBE  $\Gamma$  of  $G$  corresponding to (c).

### 3 NP-Completeness for $k$ UBE ( $k \geq 3$ )

We prove that the  $k$ UBE TESTING problem of deciding whether a digraph  $G$  admits an upward  $k$ -page book embedding is NP-complete for each fixed  $k \geq 3$ . The proof uses a reduction from the BETWEENNESS problem [67].



## BETWEENNESS

*Instance:* A finite set  $S$  of elements and a set  $R \subseteq S \times S \times S$  of triplets.

*Question:* Does there exist an ordering  $\tau : S \rightarrow \mathbb{N}$  of the elements of  $S$  such that for any element  $(a, b, c) \in R$  either  $\tau(a) < \tau(b) < \tau(c)$  or  $\tau(c) < \tau(b) < \tau(a)$ ?

We incrementally define a set of families of digraphs and prove some properties of these digraphs. Then, we use the digraphs of these families to reduce a generic instance of BETWEENNESS to an instance of 3UBE TESTING, thus proving the hardness result for  $k = 3$ . We then explain how the proof can be easily adapted to work for  $k > 3$ .

For a digraph  $G$ , we denote by  $u \rightsquigarrow v$  a directed path from a vertex  $u$  to a vertex  $v$  in  $G$ . Let  $\gamma = \langle \pi, \sigma \rangle$  be a 3UBE of  $G$ . Two edges  $(u, v)$  and  $(w, z)$  of  $G$  *conflict* if either  $\pi(u) < \pi(w) < \pi(v) < \pi(z)$  or  $\pi(w) < \pi(u) < \pi(z) < \pi(v)$ . Two conflicting edges cannot be assigned to the same page. The next property will be used in the following; it is immediate from the definition of book embedding and from the pigeonhole principle.

► **Property 1.** *In a 3UBE there cannot exist 4 edges that mutually conflict.*

**Shell digraphs.** The first family that we define are the *shell digraphs*, recursively defined as follows. Digraph  $G_0$ , depicted in Fig. 2a, consists of a directed path  $P$  with 8 vertices denoted as  $s_0, q_0, p_{-1}, t_{-1}, s'_0, q'_0, t'_0$ , and  $p_0$  in the order they appear along  $P$ . Besides the edges of  $P$ , the following directed edges exist in  $G_0$ :  $(s_0, s'_0), (q_0, q'_0), (t_{-1}, p_0)$ . Finally, there is a vertex  $t_0$  connected to  $P$  by means of the two directed edges  $(p_{-1}, t_0)$  and  $(t'_0, t_0)$ . Graph  $G_h$  is obtained from  $G_{h-1}$  with additional vertices and edges as shown in Fig. 2b. A new directed path of two vertices  $s_h$  and  $q_h$  is connected to  $G_{h-1}$  with the edge  $(q_h, s_{h-1})$ ; a second path of four vertices  $s'_h, q'_h, t'_h$ , and  $p_h$  is connected to  $G_h$  with the edge  $(t_{h-1}, s'_h)$ . The following edges exist between these new vertices:  $(s_h, s'_h), (q_h, q'_h), (t_{h-1}, p_h)$ . Finally, there is a vertex  $t_h$  connected to the other vertices by means of the two directed edges  $(p_{h-1}, t_h)$  and  $(t'_h, t_h)$ . For any  $h \geq 0$ , the edges  $(s_h, s'_h)$  and  $(q_h, q'_h)$  are called the *forcing edges* of  $G_h$ ; the edges  $(p_{h-1}, t_h)$  and  $(t_{h-1}, p_h)$  are the *channel edges* of  $G_h$ ; the edge  $(t'_h, t_h)$  is the *closing edge* of  $G_h$ . The vertices and edges of  $G_h \setminus G_{h-1}$  are the *exclusive vertices and edges* of  $G_h$ . The following lemma establishes some basic properties of the shell digraphs.

► **Lemma 2.** *Every shell digraph  $G_h$  for  $h \geq 0$  admits a 3UBE. In any 3UBE  $\gamma = \langle \pi, \sigma \rangle$  of  $G_h$  the following conditions hold for every  $i = 0, 1, \dots, h$ :*

**S1** *all vertices of  $G_i$  are between  $s_i$  and  $t_i$  in  $\pi$ ;*

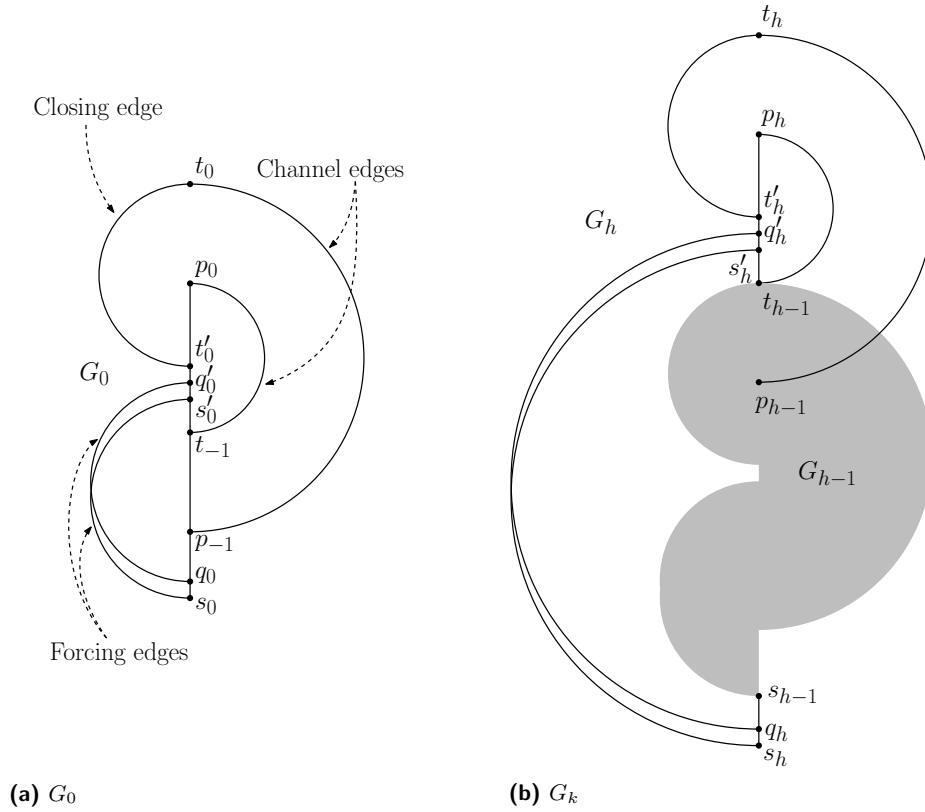
**S2** *the channel edges of  $G_i$  are in the same page;*

**S3** *if  $i > 0$ , the channel edges of  $G_i$  and those of  $G_{i-1}$  are in different pages.*

Note that Condition S1 uniquely defines the vertex ordering of  $G_h$  in every 3UBE. Namely, the path  $s_h \rightsquigarrow p_0$  precedes each path  $t_{i-1} \rightsquigarrow p_i$  (for  $i = 1, \dots, h$ ), and each path  $t_{i-1} \rightsquigarrow p_i$  precedes the path  $t_i \rightsquigarrow p_{i+1}$  (for  $i = 1, \dots, h - 1$ ) (see Fig. 3a for an example with  $h = 2$ ).

**Filled shell digraphs.** Let  $G_h$  be a shell digraph. A *filled shell digraph*  $H_{h,s}$  (for  $h \geq 0$  and  $s \geq 1$ ) is obtained from  $G_h$  by adding  $h + 2$  groups  $\alpha_{-1}, \alpha_0, \dots, \alpha_h$  of  $s$  vertices each; see Fig. 3b for an illustration. The vertices of group  $\alpha_i$  are denoted as  $v_{i,1}, v_{i,2}, \dots, v_{i,s}$ . These vertices will be used to map the elements of the set  $S$  of an instance of BETWEENNESS to an instance of 3UBE TESTING. For each vertex  $v_{-1,j}$  of the set  $\alpha_{-1}$  there is a directed edge  $(p_{-1}, v_{-1,j})$  and a directed edge  $(v_{-1,j}, t_{-1})$ . For each vertex  $v_{i,j}$  of the set  $\alpha_i$  with  $i \geq 0$  and  $i$  even, there is a directed edge  $(p_i, v_{i,j})$ . Finally, for each vertex  $v_{i,j}$  of the set  $\alpha_i$  with  $i \geq 0$ , there is a directed edge  $(v_{i-1,j}, v_{i,j})$ .





■ **Figure 2** Definition of shell digraphs. Edges are oriented from bottom to top.

► **Lemma 3.** *Every filled shell digraph  $H_{h,s}$  for  $s > 0$  and even  $h \geq 0$  admits a 3UBE. In any 3UBE  $\gamma = \langle \pi, \sigma \rangle$  of  $H_{h,s}$  the following conditions hold for every  $i = -1, 0, 1, \dots, h$ :*

- F1** *the vertices of the group  $\alpha_i$  are between  $p_i$  and  $t_i$  in  $\pi$ ;*
- F2** *if  $i \geq 0$  the vertices of  $\alpha_i$  are in reverse order with respect to those of  $\alpha_{i-1}$  in  $\pi$ ;*
- F3** *if  $i \geq 0$  each edge  $(v_{i-1,j}, v_{i,j})$  is in the page of the channel edges of  $G_i$  (for  $j = 1, \dots, s$ ).*

Observe that, by Condition F2, all groups  $\alpha_i$  with even index have the same ordering in  $\pi$  and all groups with odd index have the opposite order. As mentioned above the vertices in the groups  $\alpha_i$  will correspond to the elements of the set  $S$  of an instance of BETWEENNESS in the reduced instance of 3UBE TESTING. If the reduced instance admits a 3UBE, the order of the groups in  $\pi$  will give the desired order for the instance of BETWEENNESS.

**$\Lambda$ -filled shell digraphs and hardness proof.** Starting from a filled shell digraph  $H_{h,s}$ , a  $\Lambda$ -filled shell digraph  $\hat{H}_{h,s}$  is obtained by replacing some edges with a gadget that has two possible configurations in any 3UBE of  $\hat{H}_{h,s}$ . More precisely, we replace each edge  $(t'_i, p_i)$  of  $H_{h,s}$  for  $i$  odd with the gadget shown in Fig. 4a. The gadget replacing  $(t'_i, p_i)$  will be denoted as  $\Lambda_i$ . Notice that, this replacement preserves Conditions F1–F3 of Lemma 3.

► **Lemma 4.** *Every  $\Lambda$ -filled shell digraph  $\hat{H}_{h,s}$  for  $s > 0$  and even  $h \geq 0$  admits a 3UBE. In any 3UBE  $\gamma = \langle \pi, \sigma \rangle$  of  $\hat{H}_{h,s}$  the following conditions hold for every  $i = 1, 3, \dots, h-1$ :*

- G1** *the vertices of the gadget  $\Lambda_i$  are between  $t'_i$  and  $p_i$  in  $\pi$ ;*
- G2** *the vertices  $x_i$  and  $y_i$  are between  $w_i$  and  $z_i$  in  $\pi$  and there exists a 3UBE  $\gamma' = \langle \pi', \sigma' \rangle$  of  $\hat{H}_{h,s}$  where the order of  $x_i$  and  $y_i$  is exchanged in  $\pi'$ .*

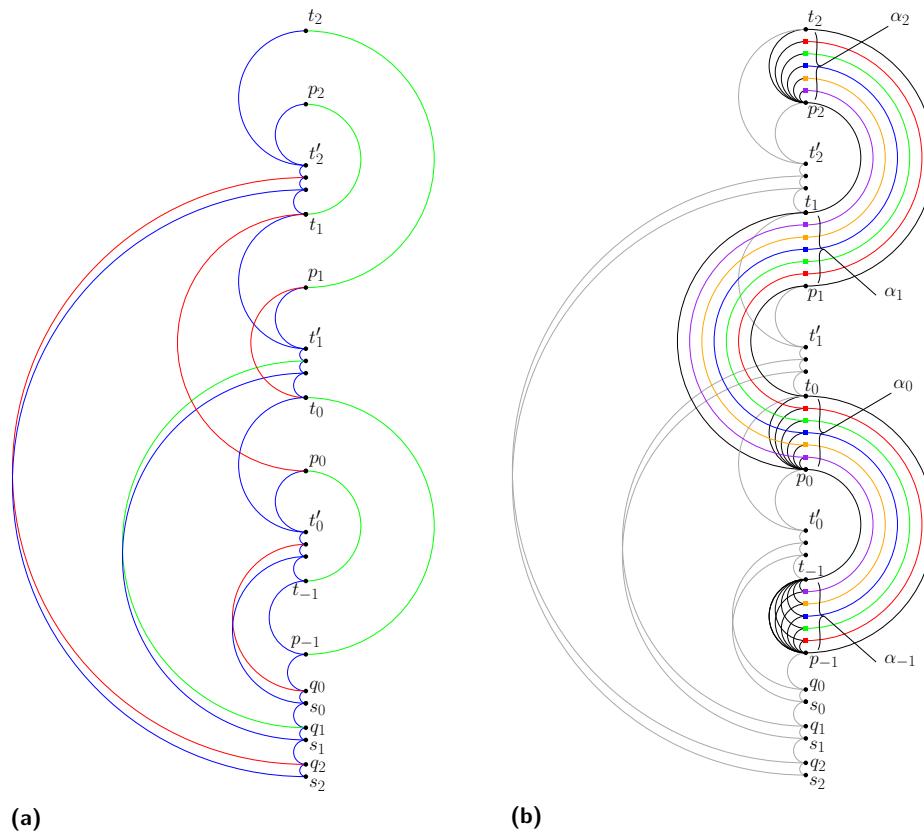


Figure 3 (a) A 3UBE of the shell digraph  $G_2$ ; the colors of the edges represent the pages. (b) Definition of  $H_{h,s}$  for  $h = 2$  and  $s = 5$ . In both figures edges are oriented from bottom to top.

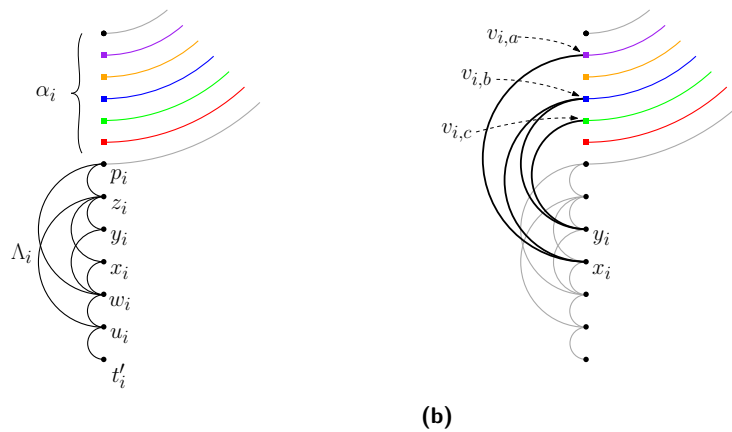
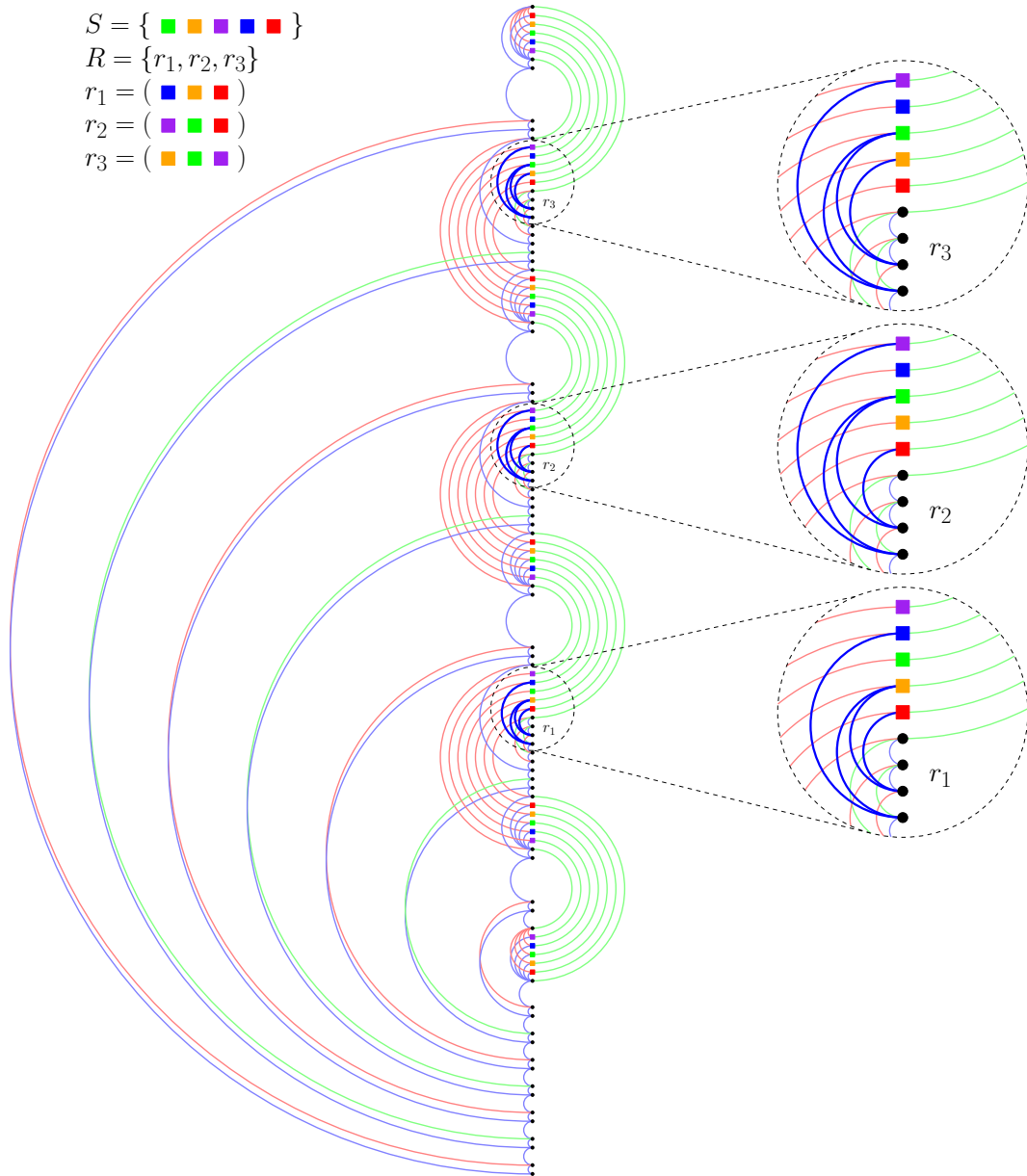


Figure 4 (a) A gadget  $\Lambda_i$  (black edges). (b) The triplet edges of  $G_i$  (bold edges).

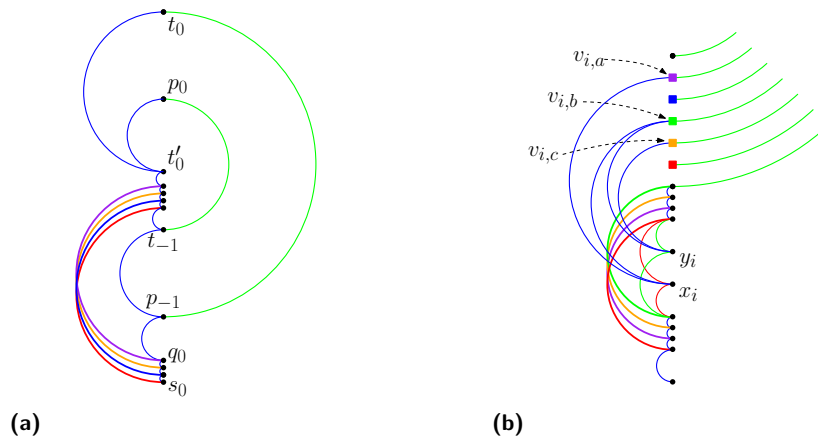
► **Theorem 5.** 3UBE TESTING is NP-complete even for  $st$ -graphs.

**Proof sketch.** 3UBE TESTING is clearly in NP. To prove the hardness we describe a reduction from BETWEENNESS. From an instance  $I = \langle S, R \rangle$  of BETWEENNESS we construct an instance  $G_I$  of 3UBE TESTING that is an  $st$ -graph; we start from the  $\Lambda$ -filled shell digraph  $\hat{H}_{h,s}$  with  $h = 2|R|$  and  $s = |S|$ . Let  $v_1, v_2, \dots, v_s$  be the elements of  $S$ . They are represented in  $\hat{H}_{h,s}$



■ **Figure 5** A 3UBE of the  $st$ -graph  $G_I$  reduced from a positive instance  $I = \langle S, R \rangle$  of BETWEENNESS; the edge colors represent the corresponding pages. Edges are oriented from bottom to top.

by the vertices  $v_{i,1}, v_{i,2}, \dots, v_{i,s}$  of the groups  $\alpha_i$ , for  $i = -1, 0, 1, \dots, h$ . In the reduction each group  $\alpha_i$  with odd index is used to encode one triplet and, in a 3UBE of  $G_I$ , the order of the vertices in these groups (which is the same by Condition F2) corresponds to the desired order of the elements of  $S$  for the instance  $I$ . Number the triplets of  $R$  from 1 to  $|R|$  and let  $(v_a, v_b, v_c)$  be the  $j$ -th triplet. We use the group  $\alpha_i$  and the gadget  $\Lambda_i$  with  $i = 2j - 1$  to encode the triplet  $(v_a, v_b, v_c)$ . More precisely, we add to  $\hat{H}_{h,s}$  the edges  $(x_i, v_{i,a}), (x_i, v_{i,b}), (y_i, v_{i,b})$ , and  $(y_i, v_{i,c})$  (see Fig. 4b). These edges are called *triplet edges* and are denoted as  $T_i$ . In any 3UBE of  $G_I$  the triplet edges are forced to be in the same page and this is possible if and only if the constraints defined by the triplets in  $R$  are respected. The digraph obtained



■ **Figure 6** Reduction for  $k$ UBE TESTING (example with  $k = 5$ ). (a) Replacement of the forcing edges. (b) Replacement of the gadget  $\Lambda_i$ . In both figures colors represent the pages.

by the addition of the triplet edges is not an  $st$ -graph because the vertices of the last group  $\alpha_h$  are all sinks. The desired instance  $G_I$  of 3UBE TESTING is the  $st$ -graph obtained by adding the edges  $(v_{h,j}, t_h)$  (for  $j = 1, 2, \dots, s$ ). Fig. 5 shows a 3UBE of the  $st$ -graph  $G_I$  reduced from a positive instance  $I$  of BETWEENNESS. ◀

For  $k > 3$ , the reduction from an instance  $I$  of BETWEENNESS to an instance  $G_I$  of  $k$ UBE TESTING is similar. In the shell digraph every pair of forcing edges is replaced by a bundle of  $k - 1$  edges that mutually conflict (see Fig. 6a). The edges in each such bundle require  $k - 1$  pages and force all edges that conflict with them to use the  $k$ -th page. Analogously, the two edges  $(u_i, z_i)$  and  $(w_i, p_i)$  of the gadget  $\Lambda_i$  are replaced by a bundle of  $k - 1$  edges that mutually conflict (see Fig. 6b); this forces the triplet edges to be in the  $k$ -th page.

▶ **Corollary 6.**  $k$ UBE TESTING is NP-complete for every  $k \geq 3$ , even for  $st$ -graphs.

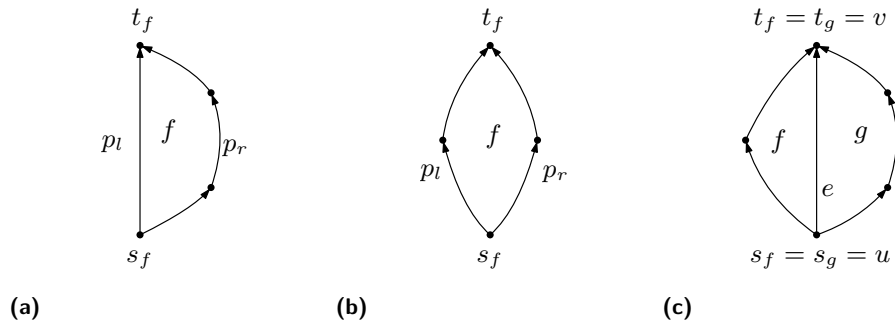
#### 4 Existential Results for 2UBE

Let  $f$  be an internal face of a plane  $st$ -graph, and let  $p_l$  and  $p_r$  be the left and the right path of  $f$ ;  $f$  is a *generalized triangle* if either  $p_l$  or  $p_r$  is a single edge (i.e., a transitive edge), and it is a *rhombus* if each of  $p_l$  and  $p_r$  consists of exactly two edges (see Figs. 7a and 7b).

Let  $G$  be a plane  $st$ -graph. A *forbidden configuration* of  $G$  consists of a transitive edge  $e = (u, v)$  shared by two internal faces  $f$  and  $g$  such that  $s_f = s_g = u$  and  $t_f = t_g = v$  (i.e., two generalized triangles sharing the transitive edge); see Fig. 7c. The absence of forbidden configurations is a necessary condition for the existence of an embedding-preserving 2UBE. If  $G$  is triangulated, the absence of forbidden configurations is also a sufficient condition [65].

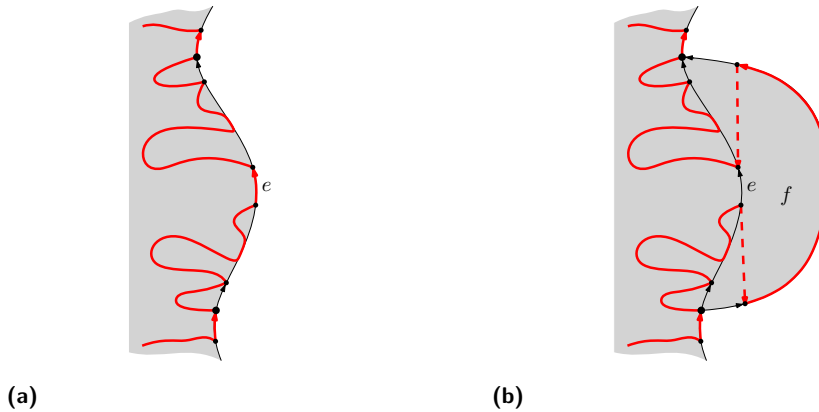
▶ **Theorem 7.** Any plane  $st$ -graph such that the left and the right path of every internal face contain at least two and three edges, respectively, admits an embedding-preserving 2UBE.

**Proof sketch.** We prove how to construct an embedding-preserving HP-completion. The idea is to construct  $\bar{G}$  by adding a face of  $G$  per time from left to right, according to a topological ordering of the dual graph of  $G$ . When a face  $f$  is added, its right path is attached to the right boundary of the current digraph. We maintain the invariant that at least one edge  $e$  in the left path of  $f$  belongs to the Hamiltonian path of the current digraph. The



■ **Figure 7** (a) A generalized triangle  $G$ . (b) A rhombus (c) A forbidden configuration.

Hamiltonian path is extended by replacing  $e$  with a path that traverses the vertices of the right path of  $f$ . To this aim, dummy edges are suitably inserted inside  $f$ . When all faces are added, the resulting graph is an HP-completion  $\overline{G}$  of  $G$ . The idea is illustrated in Fig. 8. ◀



■ **Figure 8** Idea of the construction in the proof of Theorem 7. Dummy edges are dashed.

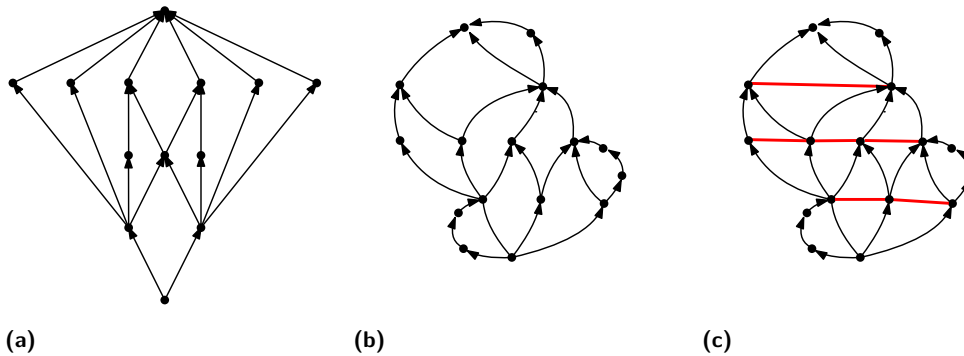
The next theorem is proved with a construction similar to that of Theorem 7.

► **Theorem 8.** *Let  $G$  be a plane st-graph such that every internal face of  $G$  is a rhombus. Then  $G$  admits an embedding-preserving 2UBE.*

## 5 Testing 2UBE for Plane Graphs with Special Faces

By Theorem 7, if all internal faces of a plane st-graph  $G$  are such that their left and right path contain at least two and at least three edges, respectively,  $G$  admits an embedding-preserving 2UBE. If these conditions do not hold, an embedding-preserving 2UBE may not exist (see Fig. 9a). We now describe an efficient testing algorithm for a plane st-graph  $G = (V, E)$  whose internal faces are generalized triangles or rhombi (see Fig. 9b). We construct a *mixed* graph  $G_M = (V, E \cup E_U)$ , where  $E_U$  is a set of undirected edges and  $(u, v) \in E_U$  if  $u$  and  $v$  are the two vertices of a rhombus face  $f$  distinct from  $s_f$  and  $t_f$  (red edges in Fig. 9c). For a rhombus face  $f$ , the graph obtained from  $G$  by adding the directed edge  $(u, v)$  inside  $f$  is still a plane st-graph (see, e.g. [17, 32]). Since there is only one edge of  $E_U$  inside each rhombus face of  $G$ , this implies the following property.

► **Property 2.** Every orientation of the edges in  $E_U$  transforms  $G_M$  into an acyclic digraph.



■ **Figure 9** (a) A plane  $st$ -graph that does not admit a 2UBE [66]. (b) A plane  $st$ -graph  $G$  whose faces are generalized triangles or rhombi. (c) The mixed graph  $G_M = (V, E, E_U)$ .

► **Theorem 9.** Let  $G$  be a plane  $st$ -graph such that every internal face of  $G$  is either a generalized triangle or a rhombus. There is an  $O(n)$ -time algorithm that decides whether  $G$  admits an embedding-preserving 2UBE, and which computes it in the positive case.

**Proof.** The edges of  $E_U$  are the only edges that can be used to construct an embedding-preserving HP-completion of  $G$ . This, together with Theorem 1, implies that  $G$  admits a 2UBE if and only if the undirected edges of  $G_M$  can be oriented so that the resulting digraph  $\overrightarrow{G_M}$  has a directed Hamiltonian path from  $s$  to  $t$ . By Property 2, any orientation of the undirected edges of  $G_M$  gives rise to an acyclic digraph. On the other hand an acyclic digraph is Hamiltonian if and only if it is unilateral (see, e.g. [5, Theorem 4]); we recall that a digraph is *unilateral* if each pair of vertices is connected by a directed path (in at least one of the two directions) [64]. Testing whether the undirected edges of  $G_M$  can be oriented so that the resulting digraph  $\overrightarrow{G_M}$  is unilateral, and computing such an orientation if it exists, can be done in time  $O(|V| + |E| + |E_U|) = O(n)$  [64, Theorem 4]. A Hamiltonian path of  $\overrightarrow{G_M}$  is given by a topological ordering of its vertices. ◀

## 6 Testing Algorithms for 2UBE Parameterized by the Branchwidth

In this section, we show that the 2UBE TESTING problem is fixed-parameter tractable with respect to the branchwidth of the input  $st$ -graph both in the fixed and in the variable embedding setting. Since the treewidth  $tw(G)$  and the branchwidth  $bw(G)$  of a graph  $G$  are within a constant factor from each other (i.e.,  $bw(G) - 1 \leq tw(G) \leq \lfloor \frac{3}{2}bw(G) \rfloor - 1$  [70]), our FPT algorithm also extends to graphs of bounded treewidth. Previously, the complexity of this problem was settled only for graphs of treewidth at most 2 in the variable embedding setting<sup>1</sup> [37].

We use the SPQR-tree data structure [34] to efficiently handle the planar embeddings of the input digraphs and sphere-cut decompositions [71] to develop a dynamic-programming approach on the skeletons of the rigid components. For the definition of the *SPQR-tree*  $\mathcal{T}$  of a biconnected graph and the related concepts of *skeleton*  $skel(\mu)$  and *pertinent graph*  $pert(\mu)$  of a node  $\mu$  of  $\mathcal{T}$ , *types* of the nodes of  $\mathcal{T}$  (namely, *S*-, *P*-, *Q*-, and *R*-nodes), and *virtual edges*

<sup>1</sup> To our knowledge, no efficient algorithm was known for treewidth 2 in the fixed embedding setting.

of a skeleton, see [23]. To ease the description, we can assume that each S-node has exactly two children [41] and that the skeleton of each node  $\mu$  does not contain the virtual edge representing the parent of  $\mu$ . In particular, we will exploit the following property of  $\mathcal{T}$  when  $G$  is an  $st$ -graph containing the edge  $e = (s, t)$  and  $\mathcal{T}$  is rooted at the Q-node of  $e$ .

► **Property 3** ([34]). *Let  $\mu \in \mathcal{T}$  with poles  $u$  and  $v$ . Without loss of generality, assume that the directed paths connecting  $u$  and  $v$  in  $G$  are oriented from  $u$  to  $v$ . Then,  $\text{pert}(\mu)$  is a  $uv$ -graph.*

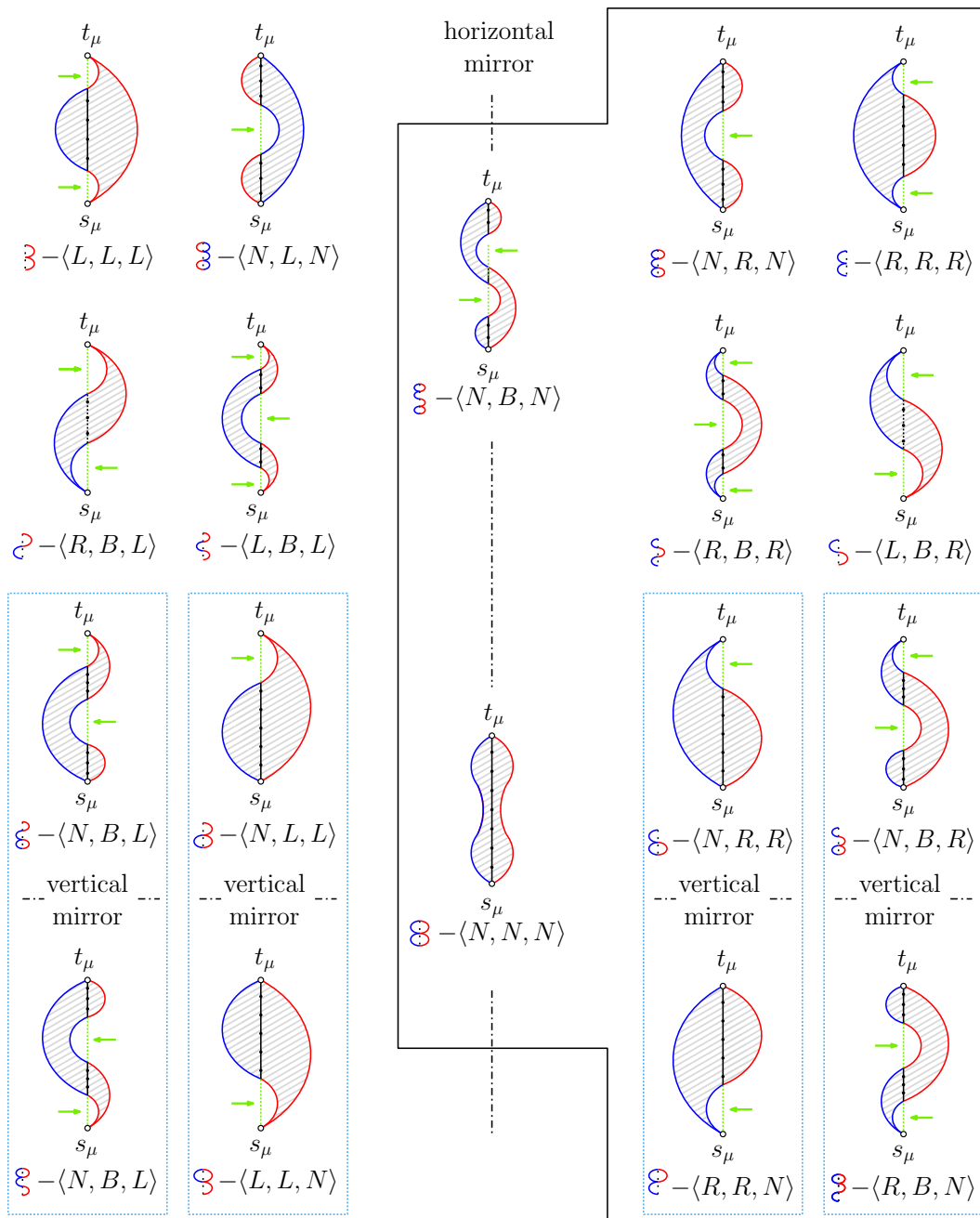
For the definition of *branchwidth* and *sphere-cut decomposition*, and for the related concepts of *middle set*  $\text{mid}(e)$  and *noose*  $\mathcal{O}_e$  of an arc  $e$  of the decomposition, and *length* of a noose, see [23]. We denote a sphere-cut decomposition of a plane graph  $G = (V, E)$  by the triple  $\langle T, \xi, \Pi = \bigcup_{a \in E(T)} \pi_a \rangle$ , where  $T$  is a ternary tree whose leaves are in a one-to-one correspondence with the edges of  $G$ , which is defined by a bijection  $\xi : \mathcal{L}(T) \leftrightarrow E(G)$  between the leaf set  $\mathcal{L}(T)$  of  $T$  and the edge set  $E$ , and where  $\pi_a$  is a circular order of  $\text{mid}(a)$ , for each arc  $a$  of  $T$ . In particular, we will exploit the property that each of the two subgraphs that lie in the interior and in the exterior of a noose is connected and that the set of nooses forms a laminar set family, that is, any two nooses are either disjoint or nested.

Without loss of generality, we assume that the input  $st$ -graph  $G$  contains the edge  $(s, t)$ , which guarantees that  $G$  is biconnected. In fact, in any 2UBE of  $G$  vertices  $s$  and  $t$  have to be the first and the last vertex of the spine, respectively. Thus, either  $(s, t)$  is an edge of  $G$  or it can be added to any of the two pages of the spine of a 2UBE of  $G$  to obtain a 2UBE  $\langle \pi, \sigma \rangle$  of  $G \cup (s, t)$ . Clearly, the edge  $(s, t)$  will be incident to the outer face of  $\Gamma(\pi, \sigma)$ .

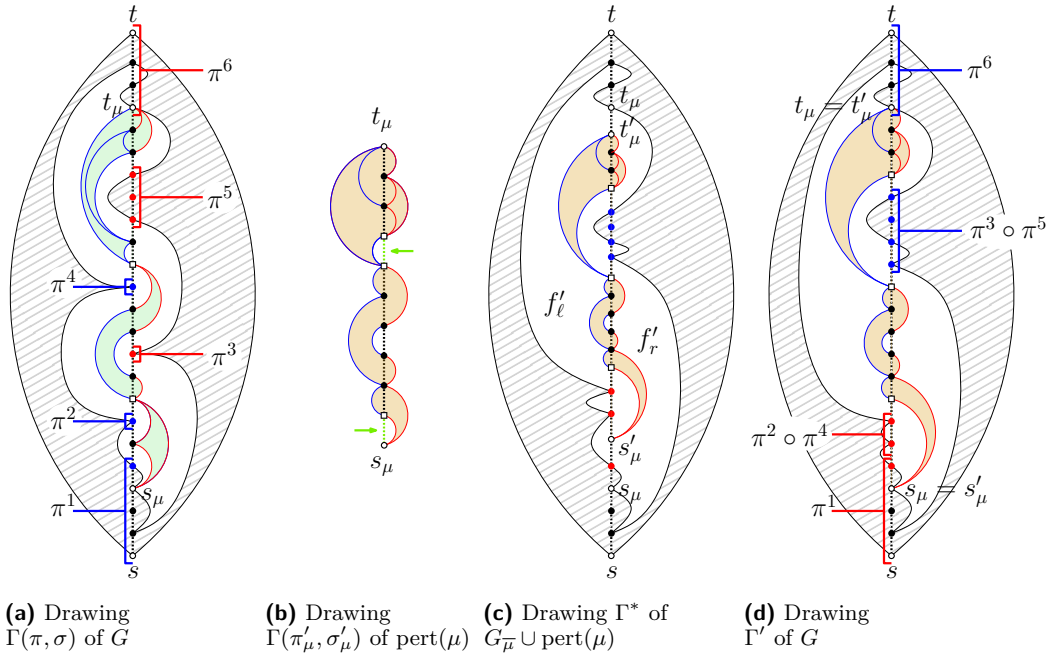
**Overview.** Our approach leverages on the classification of the embeddings of each triconnected component of the biconnected graph  $G$ . Intuitively, such classification is based on the visibility of the spine that the embedding “leaves” on its outer face. We show that the planar embeddings of a triconnected component that yield a 2UBE of the component can be partitioned into a finite number of equivalence classes, called *embedding types*. By visiting the SPQR-tree  $\mathcal{T}$  of  $G$  bottom-up, we describe how to compute all the *realizable embedding types* of each triconnected component, that is, those embedding types that are allowed by some embedding of the component. To this aim we will exploit the realizable embedding types of its child components. If the root of  $\mathcal{T}$ , which represents the whole  $st$ -graph  $G$ , admits at least one planar embedding belonging to some embedding type, then  $G$  admits a 2UBE. The most challenging part of this approach is handling the triconnected components that correspond to the P-nodes, where the problem is reduced to a maximum flow problem on a capacitated flow network with edge demands, and to the R-nodes, where a sphere-cut decomposition of bounded width is used to efficiently compute the feasible embedding types.

**Embedding Types.** Given a 2UBE  $\langle \pi, \sigma \rangle$ , the two pages will be called the *left page* (the one to the left of the spine) and the *right page* (the one to the right of the spine), respectively. We write  $\sigma(e) = L$  (resp.  $\sigma(e) = R$ ) if the edge  $e$  is assigned to the left page (resp. right page). A point  $p$  of the spine is *visible from the left (right) page* if it is possible to shoot a horizontal ray originating from  $p$  and directed leftward (rightward) without intersecting any edge in  $\Gamma(\pi, \sigma)$ . Let  $\mu$  be a node of the SPQR-tree  $\mathcal{T}$  of  $G$  rooted at  $(s, t)$ . Recall that, by Property 3, since  $\mathcal{T}$  has been rooted at  $(s, t)$ , the pertinent graph  $\text{pert}(\mu)$  and the skeleton  $\text{skel}(\mu)$  of  $\mu$  are  $s't'$ -graphs, where  $s'$  and  $t'$  are the poles of  $\mu$ . We denote by  $s_\mu$  (by  $t_\mu$ ) the pole of  $\mu$  that is the source (the sink) of  $\text{pert}(\mu)$  and of  $\text{skel}(\mu)$ . Let  $\langle \pi_\mu, \sigma_\mu \rangle$  be a 2UBE of  $\text{pert}(\mu)$  and let  $\mathcal{E}_\mu$  be the embedding of  $\Gamma(\pi_\mu, \sigma_\mu)$ . We say that  $\mathcal{E}_\mu$  has *embedding type* (or is of *Type*)  $\langle s\_vis, spine\_vis, t\_vis \rangle$  with  $s\_vis, t\_vis \in \{L, R, N\}$  and  $spine\_vis \in \{L, R, B, N\}$  where:





■ **Figure 10** Illustrations of the possible embedding types of a node  $\mu$  with poles  $s_\mu$  and  $t_\mu$ ; the portion of the spine that is visible from the left or from the right is green. Pairs of embedding types in the same dotted box are one the vertically-mirrored copy of the other. Embedding types on the top are the horizontally-mirrored copy of the ones on the bottom. Embedding types  $\langle N, B, N \rangle$  and  $\langle N, N, N \rangle$  are the horizontal and vertical mirrored copies of themselves.



■ **Figure 11** Illustrations for Lemma 11. The 2UBE of  $\text{pert}(\mu)$  are of Type  $\mathfrak{S}\text{-}\langle L, B, N \rangle$ .

1.  $s\_vis$  is  $L$  (resp.,  $R$ ) if in  $\mathcal{E}_\mu$  there is a portion of the spine incident to  $s$  and between  $s$  and  $t$  that is visible from the left page (resp., from the right page); otherwise,  $s\_vis$  is  $N$ .
2.  $t\_vis$  is  $L$  (resp.,  $R$ ) if in  $\mathcal{E}_\mu$  there is a portion of the spine incident to  $t$  and between  $s$  and  $t$  that is visible from the left page (resp., from the right page); otherwise,  $t\_vis$  is  $N$ .
3.  $spine\_vis$  is  $L$  (resp.,  $R$ ) if in  $\mathcal{E}_\mu$  there is a portion of the spine between  $s$  and  $t$  that is visible from the left page (resp., from the right page);  $spine\_vis$  is  $B$  if in  $\mathcal{E}_\mu$  there is a portion of the spine between  $s$  and  $t$  that is visible from the left page, and a portion of the spine between  $s$  and  $t$  that is visible from the right page; otherwise,  $spine\_vis$  is  $N$ .

We also say that a node  $\mu$  and  $\text{pert}(\mu)$  admits Type  $\langle x, y, z \rangle$  if  $\text{pert}(\mu)$  admits an embedding of Type  $\langle x, y, z \rangle$ . We have the following lemma.

► **Lemma 10.** *Let  $\mu$  be a node of  $\mathcal{T}$ , let  $\langle \pi_\mu, \sigma_\mu \rangle$  be a 2UBE of  $\text{pert}(\mu)$  and let  $\mathcal{E}_\mu$  be a planar embedding of  $\Gamma(\pi_\mu, \sigma_\mu)$ . Then  $\mathcal{E}_\mu$  has exactly one embedding type, where the possible embedding types are the 18 depicted in Fig. 10.*

Let  $\langle \pi, \sigma \rangle$  be a 2UBE of  $G$ , let  $\mu$  a node of  $\mathcal{T}$ , and let  $\langle \pi_\mu, \sigma_\mu \rangle$  be the restriction of  $\langle \pi, \sigma \rangle$  to  $\text{pert}(\mu)$ . Further, let  $\langle \pi'_\mu, \sigma'_\mu \rangle \neq \langle \pi_\mu, \sigma_\mu \rangle$  be a 2UBE of  $\text{pert}(\mu)$ .

► **Lemma 11.** *If  $\langle \pi'_\mu, \sigma'_\mu \rangle$  and  $\langle \pi_\mu, \sigma_\mu \rangle$  have the same embedding type, then  $G$  admits a 2UBE whose restriction to  $\text{pert}(\mu)$  is  $\langle \pi'_\mu, \sigma'_\mu \rangle$ .*

**Proof sketch.** First, insert a possibly squeezed copy of  $\Gamma(\pi'_\mu, \sigma'_\mu)$  (Fig. 11b) inside  $\Gamma(\pi, \sigma)$  (Fig. 11a) in the interior of the face  $f_\mu$  of the plane digraph  $G_{\bar{\mu}}$  resulting from removing  $\text{pert}(\mu)$  (except its poles) from  $\Gamma(\pi, \sigma)$ . Second, suitably move parts of the boundary of  $f_\mu$  along portions of the spine incident to the inserted drawing of  $\text{pert}(\mu)$  (Fig. 11c). Then, continuously move the copies of the poles of  $\mu$  inside  $f_\mu$  towards their copies in  $\Gamma(\pi, \sigma)$ , without intersecting any edge, to obtain a drawing  $\Gamma'$  of  $G$  (Fig. 11d). ◀

Recall that, for each node  $\mu$  of  $\mathcal{T}$ ,  $\text{pert}(\mu)$  may have exponentially many embeddings, given by the permutations of the children of the P-nodes and by the flips of the R-nodes. Lemma 11 is the reason why we only need to compute a single embedding for each embedding type realizable by  $\text{pert}(\mu)$ , i.e., a constant number of embeddings instead of an exponential number.

We first describe an algorithm to decide if  $G$  admits a 2UBE and its running time. The same procedure can be easily refined to actually compute a 2UBE of  $G$ , with no additional cost, by decorating each node  $\mu \in \mathcal{T}$  with the embedding choices performed at  $\mu$ , for each of its  $O(1)$  possible embedding types.

**Testing Algorithm.** The algorithm is based on computing, for each non-root node  $\mu$  of  $\mathcal{T}$ , the set of embedding types realizable by  $\text{pert}(\mu)$ , based on whether  $\mu$  is an S-, P-, Q-, or an R-node. Since, by Lemmas 10 and 11,  $G$  admits a 2UBE if and only if the pertinent graph of the unique child of the root Q-node admits an embedding of at least one of the 18 possible embedding types, this approach allows us to solve the 2UBE TESTING problem for  $G$ .

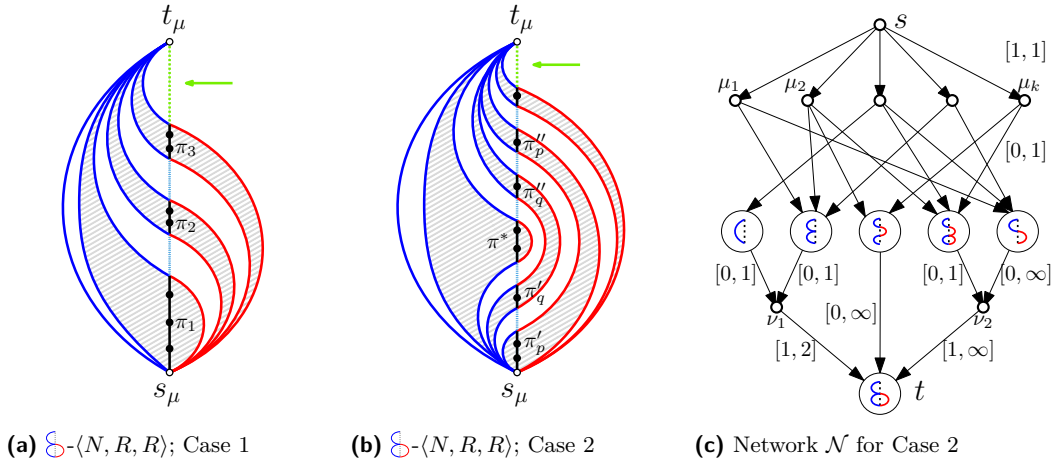
Recall that the only possible embedding choices for  $G$  happen at P- and R-nodes. While the treatment of Q- and S-nodes does not require any modification when considering the variable and the fixed embedding settings, for P- and R-nodes we will discuss how to compute the embedding types that are realizable by  $\text{pert}(\mu)$  in both such settings. In particular, in the fixed embedding scenario the above characterization needs to additionally satisfy the constraints imposed by the fixed embedding on the skeletons of the P- and R-nodes in  $\mathcal{T}$ .

Note that a leaf Q-node only admits embeddings of type  $\mathcal{D}\text{-}\langle L, L, L \rangle$  or  $\mathcal{C}\text{-}\langle R, R, R \rangle$ . Also, combining 2UBEs of the two children of an S-node  $\mu$  always yields a 2UBE of  $\text{pert}(\mu)$ , whose embedding type can be easily computed. In [23] we prove the following.

► **Lemma 12.** *Let  $\mu$  be an S-node. The set of embedding types realizable by  $\text{pert}(\mu)$  can be computed in  $O(1)$  time, both in the fixed and in the variable embedding setting.*

**P-nodes.** Let  $\mu$  be a P-node with poles  $s_\mu$  and  $t_\mu$ . Recall that an embedding for a P-node is obtained by choosing a permutation for its children and an embedding type for each child. Our approach to compute the realizable types of  $\text{pert}(\mu)$  consists of considering one type at a time for  $\mu$ . For each embedding type, we check whether the children of  $\mu$ , together with their realizable embedding types, can be arranged in a finite number of families of permutations (which we prove to be a constant number) so to yield an embedding of the considered type. In order to ease the following description, consider that the arrangements of the children for obtaining some embedding types can be easily derived from the arrangements to obtain the (horizontally) symmetric ones by (i) reversing the left-to-right sequence of the children in the construction and (ii) by taking, for each child, the horizontally-mirrored embedding type; for instance, the arrangements to construct an embedding of Type  $\mathcal{Q}\text{-}\langle L, L, N \rangle$  can be obtained from the ones to construct an embedding of Type  $\mathcal{P}\text{-}\langle R, R, N \rangle$ , and vice versa. Moreover, two embedding types, namely Type  $\mathcal{S}\text{-}\langle N, B, N \rangle$  and Type  $\mathcal{R}\text{-}\langle N, N, N \rangle$ , are (horizontally) self-symmetric. As a consequence, in order to consider all the embedding types that are realizable by  $\text{pert}(\mu)$  we describe how to obtain only 10 “relevant” embedding types (enclosed by a solid polygon in Fig. 10):  $\mathcal{E}\text{-}\langle R, R, R \rangle$ ,  $\mathcal{F}\text{-}\langle N, R, N \rangle$ ,  $\mathcal{G}\text{-}\langle R, B, R \rangle$ ,  $\mathcal{H}\text{-}\langle L, B, R \rangle$ ,  $\mathcal{I}\text{-}\langle N, R, R \rangle$ ,  $\mathcal{J}\text{-}\langle N, B, R \rangle$ ,  $\mathcal{K}\text{-}\langle R, R, N \rangle$ ,  $\mathcal{L}\text{-}\langle R, B, N \rangle$ ,  $\mathcal{M}\text{-}\langle N, B, N \rangle$ , and  $\mathcal{N}\text{-}\langle N, N, N \rangle$ .

Next, we give necessary and sufficient conditions under which the pertinent graph of a P-node admits an embedding of Type  $\mathcal{E}\text{-}\langle N, R, R \rangle$ . Then, we show how to test these conditions efficiently by exploiting a suitably defined flow network. The conditions for the remaining types, given in [23], can be tested with the same algorithmic strategy.



■ **Figure 12** Case 1 (a) and Case 2 (b) of Lemma 13 for a P-nodes  $\mu$  of Type  $\mathcal{S}\text{-}\langle N, R, R \rangle$ . The spine is colored either green, blue, or black. The green part is the portion of the spine that is visible from the right, the black parts correspond to the bottom-to-top sequences of the internal vertices of  $\text{pert}(\mu)$  inherited from the 2UBEs of the children of  $\mu$ , the blue parts join sequences inherited from different children. (c) Capacitated flow network  $\mathcal{N}$  with edge demands corresponding to (b).

► **Lemma 13** (Type  $\mathcal{S}\text{-}\langle N, R, R \rangle$ ). *Let  $\mu$  be a P-node. Type  $\mathcal{S}\text{-}\langle N, R, R \rangle$  is admitted by  $\mu$  in the variable embedding setting if and only if at least one of two cases occurs. (**Case 1**) The children of  $\mu$  can be partitioned into two parts: The first part consists either of a Type  $\mathcal{Q}\text{-}\langle R, R, R \rangle$  Q-node child, or of a Type  $\mathcal{S}\text{-}\langle N, R, R \rangle$  child, or both. The second part consists of any number, even zero, of Type  $\mathcal{S}\text{-}\langle L, B, R \rangle$  children. (**Case 2**) The children of  $\mu$  can be partitioned into three parts: The first part consists either of a Type  $\mathcal{Q}\text{-}\langle R, R, R \rangle$  Q-node child, or of a non-Q-node Type  $\mathcal{S}\text{-}\langle R, R, R \rangle$  child, or both. The second part consists of any number, even zero, of Type  $\mathcal{S}\text{-}\langle R, B, R \rangle$  children. The third part consists of any positive number of Type  $\mathcal{S}\text{-}\langle N, B, R \rangle$  or Type  $\mathcal{S}\text{-}\langle L, B, R \rangle$  children, with at most one Type  $\mathcal{S}\text{-}\langle N, B, R \rangle$  child.*

Regarding the time complexity of testing the existence of a Type  $\mathcal{S}\text{-}\langle N, R, R \rangle$  embedding of  $\text{pert}(\mu)$ , we show that deciding if one of (**Case 1**) or (**Case 2**) of Lemma 13 applies can be reduced to a network flow problem on a network  $\mathcal{N}$  with edge demands. The network for (**Case 2**) is depicted in Fig. 12c. The details of this construction are given in [23].

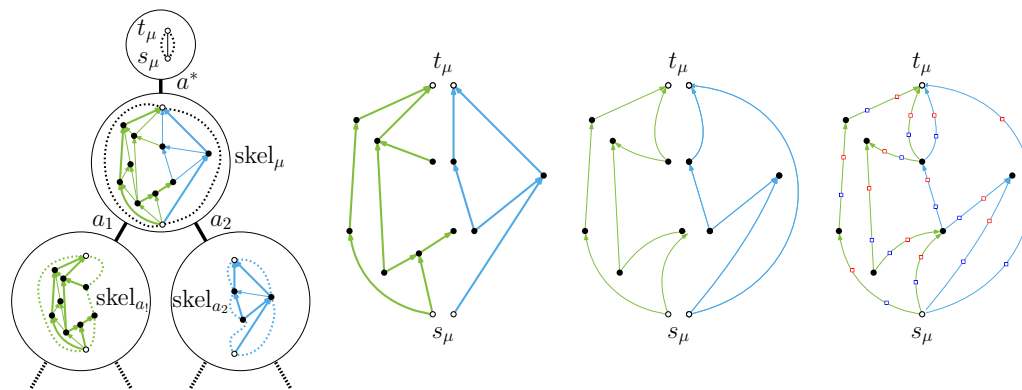
► **Lemma 14.** *Let  $\mu$  be a P-node with  $k$  children. The set of embedding types realizable by  $\text{pert}(\mu)$  can be computed in  $O(k^2)$  time in the variable embedding setting.*

The fixed embedding scenario for a P-node  $\mu$  can be addressed by processing the children of  $\mu$  in the left-to-right order defined by the given embedding of  $G$ . The details of such an approach are given in [23], where the following is proven.

► **Lemma 15.** *Let  $\mu$  be a P-node with  $k$  children. The set of embedding types realizable by  $\text{pert}(\mu)$  can be computed in  $O(k)$  time in the fixed embedding setting.*

Lemmas 12 and 15 yield a counterpart, in the fixed embedding setting, of the linear-time algorithm by Di Giacomo et al. [37] to compute 2UBEs of series-parallel graphs.

► **Theorem 16.** *There exists an  $O(n)$ -time algorithm to decide whether an  $n$ -vertex series-parallel st-graph admits an embedding-preserving 2UBE.*



(a) A sphere-decomposition of  $\text{skel}(\mu) \cup (s_\mu, t_\mu)$  (b) Outer faces of  $\text{skel}_{a_1}$  and  $\text{skel}_{a_2}$  (c) Directed paths of  $\text{skel}_{a_1}$  and  $\text{skel}_{a_2}$  (d) Graph  $A = A_1 \cup A_2$

■ **Figure 13** Illustrations for the R-node case. (a) A partial sphere-cut decomposition of the graph  $\text{skel}(\mu) \cup (s_\mu, t_\mu)$  rooted at the edge  $(s_\mu, t_\mu)$ , where  $\mu$  is an R-node. The nooses are dotted curves.

**R-nodes.** Let  $\mu$  be an R-node and  $(T, \xi, \Pi)$  be a sphere-cut decomposition of  $\text{skel}(\mu) \cup (s_\mu, t_\mu)$  of width  $\beta$ , rooted at the node  $\rho$  with  $\xi(\rho) = (s_\mu, t_\mu)$ ; refer to Fig. 13a. Each arc  $a$  of  $T$  is associated with a subgraph  $\text{skel}_a$  of  $\text{skel}(\mu)$  and with a subgraph  $\text{pert}_a$  of  $\text{pert}(\mu)$ , both bounded by the noose of  $a$ . Let  $a_1$  and  $a_2$  be the two arcs leading to  $a$  from the bottom of  $T$ . Intuitively, our strategy to compute the embedding types of  $\text{pert}(\mu)$  is to visit  $T$  bottom-up maintaining a succinct description of size  $O(\beta)$  of the properties of the 2UBEs of  $\text{pert}_a$ . To this aim, we construct cycles composed of directed edges that are in one-to-one correspondence with maximal directed paths along the outer face of  $\text{skel}_{a_1}$  and  $\text{skel}_{a_2}$  (Figs. 13b and 13c), which we use to define an auxiliary graph  $A$  whose 2UBEs concisely represent the possible 2UBEs of  $\text{pert}_a$  obtained by combining the 2UBEs of  $\text{pert}_{a_1}$  and  $\text{pert}_{a_2}$  (Fig. 13d). When we reach the arc  $a^*$  incident to  $\rho$  with  $\text{skel}_{a^*} = \text{skel}(\mu)$ , we use the computed properties to determine the embedding types realizable by  $\text{pert}(\mu)$ . We provide full details in [23].

► **Lemma 17.** *Let  $\mu$  be an R-node whose skeleton  $\text{skel}(\mu)$  has  $k$  children and branchwidth  $\beta$ . The set of embedding types realizable by  $\text{pert}(\mu)$  can be computed in  $O(2^{O(\beta \log \beta)} \cdot k)$  time, both in the fixed and in the variable embedding setting, provided that a sphere-cut decomposition  $\langle T_\mu, \xi_\mu, \Pi_\mu \rangle$  of width  $\beta$  of  $\text{skel}^+(\mu)$  is given.*

By Lemmas 12, 14, 15 and 17 and since  $\mathcal{T}$  has  $O(|G|)$  size [34, 41], we get the following.

► **Theorem 18.** *There exists an  $O(2^{O(\beta \log \beta)} \cdot n + n^2 + g(n))$ -time algorithm to decide if an  $n$ -vertex planar (plane) st-graph of branchwidth  $\beta$  admits a (embedding-preserving) 2UBE, where  $g(n)$  is the computation time of a sphere-cut decomposition of an  $n$ -vertex plane graph.*

Observe that  $g(n)$  is  $O(n^3)$  by the result in [71]. Thus, we get the following.

► **Corollary 19.** *There exists an  $O(2^{O(\beta \log \beta)} \cdot n + n^3)$ -time algorithm to decide whether an  $n$ -vertex planar (plane) st-graph of branchwidth  $\beta$  admits a (embedding-preserving) 2UBE.*

Since the branchwidth of a planar graph  $G$  is at most  $2.122\sqrt{n}$  [48], Corollary 19 immediately implies that the 2UBE TESTING problem can be solved in sub-exponential time.

► **Corollary 20.** *There exists an  $O(2^{O(\sqrt{n} \log \sqrt{n})} + n^3)$ -time algorithm to decide whether an  $n$ -vertex planar (plane) st-graph admits a (embedding-preserving) 2UBE.*

## 7 Conclusion and Open Problems

Our results provide significant advances on the complexity of the  $k$ UBE TESTING problem. We showed NP-hardness for  $k \geq 3$ ; we gave FPT- and polynomial-time algorithms for relevant families of planar  $st$ -graphs when  $k = 2$ . We point out that our FPT-algorithm can be refined to run in  $O(n^2)$  time for  $st$ -graphs of treewidth at most 3, by constructing in linear time a sphere-cut decomposition of their rigid components. We conclude with some open problems.

- The main open question is about the complexity of the 2UBE TESTING problem, which has been conjectured to be NP-complete in the general case [55].
- The digraphs in our NP-completeness proof are not upward planar. Since there are upward planar digraphs that do not admit a 3UBE [58], it would be interesting to study whether the problem remains NP-complete for three pages and upward planar digraphs.
- Finally, it is natural to investigate other families of planar digraphs for which a 2UBE always exists or polynomial-time testing algorithms can be devised.

---

### References

- 1 Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Gelasio Salazar. The 2-page crossing number of  $K_n$ . In Tamal K. Dey and Sue Whitesides, editors, *Symposium on Computational Geometry 2012, SoCG '12*, pages 397–404. ACM, 2012. doi:10.1145/2261250.2261310.
- 2 Hugo A. Akitaya, Erik D. Demaine, Adam Hesterberg, and Quanquan C. Liu. Upward Partitioned Book Embeddings. In Fabrizio Frati and Kwan-Liu Ma, editors, *GD 2017*, volume 10692 of *LNCS*, pages 210–223. Springer, 2017. doi:10.1007/978-3-319-73915-1\_18.
- 3 Mustafa Alhashem, Guy-Vincent Jourdan, and Nejib Zaguia. On The Book Embedding Of Ordered Sets. *Ars Combinatoria*, 119:47–64, 2015.
- 4 Mohammad Alzohairi and Ivan Rival. Series-Parallel Planar Ordered Sets Have Pagenumber Two. In Stephen C. North, editor, *Graph Drawing, GD '96*, volume 1190 of *LNCS*, pages 11–24. Springer, 1996. doi:10.1007/3-540-62495-3\_34.
- 5 Patrizio Angelini, Michael A. Bekos, Walter Didimo, Luca Grilli, Philipp Kindermann, Tamara Mchedlidze, Roman Prutkin, Antonios Symvonis, and Alessandra Tappini. Greedy Rectilinear Drawings. In Therese Biedl and Andreas Kerren, editors, *GD 2018*, volume 11282 of *LNCS*. Springer, 2018.
- 6 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Valentino Di Donato, Philipp Kindermann, Günter Rote, and Ignaz Rutter. Windrose Planarity: Embedding Graphs with Direction-Constrained Edges. *ACM Trans. Algorithms*, 14(4):54:1–54:24, 2018. doi:10.1145/3239561.
- 7 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, and Fabrizio Frati. Strip Planarity Testing for Embedded Planar Graphs. *Algorithmica*, 77(4):1022–1059, 2017. doi:10.1007/s00453-016-0128-9.
- 8 Patrizio Angelini, Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Intersection-Link Representations of Graphs. *J. Graph Algorithms Appl.*, 21(4):731–755, 2017. doi:10.7155/jgaa.00437.
- 9 Patrizio Angelini, Giordano Da Lozzo, and Daniel Neuwirth. Advancements on SEFE and Partitioned Book Embedding problems. *Theor. Comput. Sci.*, 575:71–89, 2015.
- 10 Patrizio Angelini, Marco Di Bartolomeo, and Giuseppe Di Battista. Implementing a Partitioned 2-Page Book Embedding Testing Algorithm. In *Graph Drawing*, volume 7704 of *LNCS*, pages 79–89. Springer, 2012.
- 11 Patrizio Angelini, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Ignaz Rutter. Testing the simultaneous embeddability of two graphs whose intersection is a biconnected or a connected graph. *J. Discrete Algorithms*, 14:150–172, 2012.



- 12 Patrizio Angelini, David Eppstein, Fabrizio Frati, Michael Kaufmann, Sylvain Lazard, Tamara Mchedlidze, Monique Teillaud, and Alexander Wolff. Universal Point Sets for Drawing Planar Graphs with Circular Arcs. *J. Graph Algorithms Appl.*, 18(3):313–324, 2014.
- 13 Melanie Badent, Emilio Di Giacomo, and Giuseppe Liotta. Drawing colored graphs on colored points. *Theor. Comput. Sci.*, 408(2-3):129–142, 2008.
- 14 Michael J. Bannister and David Eppstein. Crossing Minimization for 1-page and 2-page Drawings of Graphs with Bounded Treewidth. In Christian A. Duncan and Antonios Symvonis, editors, *GD 2014*, volume 8871 of *LNCS*, pages 210–221. Springer, 2014. doi:10.1007/978-3-662-45803-7\_18.
- 15 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.
- 16 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, 2017.
- 17 Michael A. Bekos, Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Chrysanthi N. Raftopoulou. Edge Partitions of Optimal 2-plane and 3-plane Graphs. In Andreas Brandstädt, Ekkehard Köhler, and Klaus Meer, editors, *Graph-Theoretic Concepts in Computer Science, WG 2018*, volume 11159 of *LNCS*, pages 27–39. Springer, 2018. doi:10.1007/978-3-030-00256-5\_3.
- 18 Michael A. Bekos, Martin Gronemann, and Chrysanthi N. Raftopoulou. Two-Page Book Embeddings of 4-Planar Graphs. *Algorithmica*, 75(1):158–185, 2016. doi:10.1007/s00453-015-0016-8.
- 19 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.
- 20 Paola Bertolazzi, Giuseppe Di Battista, and Walter Didimo. Quasi-Upward Planarity. *Algorithmica*, 32(3):474–506, 2002. doi:10.1007/s00453-001-0083-x.
- 21 Paola Bertolazzi, Giuseppe Di Battista, Carlo Mannino, and Roberto Tamassia. Optimal Upward Planarity Testing of Single-Source Digraphs. *SIAM J. Comput.*, 27(1):132–169, 1998. doi:10.1137/S0097539794279626.
- 22 Therese C. Biedl, Thomas C. Shermer, Sue Whitesides, and Stephen K. Wismath. Bounds for Orthogonal 3-D Graph Drawing. *J. Graph Algorithms Appl.*, 3(4):63–79, 1999.
- 23 Carla Binucci, Giordano Da Lozzo, Emilio Di Giacomo, Walter Didimo, Tamara Mchedlidze, and Maurizio Patrignani. Upward Book Embeddings of st-Graphs. *CoRR*, abs/1903.07966, 2019. arXiv:1903.07966.
- 24 Carla Binucci, Emilio Di Giacomo, Md. Iqbal Hossain, and Giuseppe Liotta. 1-page and 2-page drawings with bounded number of crossings per edge. *Eur. J. Comb.*, 68:24–37, 2018. doi:10.1016/j.ejc.2017.07.009.
- 25 Carla Binucci and Walter Didimo. Computing Quasi-Upward Planar Drawings of Mixed Graphs. *Comput. J.*, 59(1):133–150, 2016. doi:10.1093/comjnl/bxv082.
- 26 Franz-Josef Brandenburg. Upward planar drawings on the standing and the rolling cylinders. *Comput. Geom.*, 47(1):25–41, 2014. doi:10.1016/j.comgeo.2013.08.003.
- 27 Jean Cardinal, Michael Hoffmann, Vincent Kusters, Csaba D. Tóth, and Manuel Wettstein. Arc diagrams, flip distances, and Hamiltonian triangulations. *Comput. Geom.*, 68:206–225, 2018. doi:10.1016/j.comgeo.2017.06.001.
- 28 Steven Chaplick, Markus Chimani, Sabine Cornelsen, Giordano Da Lozzo, Martin Nöllenburg, Maurizio Patrignani, Ioannis G. Tollis, and Alexander Wolff. Planar L-Drawings of Directed Graphs. In Fabrizio Frati and Kwan-Liu Ma, editors, *GD 2017*, volume 10692 of *LNCS*, pages 465–478. Springer, 2017. doi:10.1007/978-3-319-73915-1\_36.
- 29 Fan R. K. Chung, Frank Thomson Leighton, and Arnold L. Rosenberg. Embedding graphs in books: A layout problem with applications to VLSI design. *SIAM Journal on Algebraic Discrete Methods*, 8(1):33–58, 1987.
- 30 Robert J. Cimikowski. An analysis of some linear graph layout heuristics. *J. Heuristics*, 12(3):143–153, 2006.



- 31 Giordano Da Lozzo, Giuseppe Di Battista, Fabrizio Frati, Maurizio Patrignani, and Vincenzo Roselli. Upward Planar Morphs. In Therese C. Biedl and Andreas Kerren, editors, *GD 2018*, volume 11282 of *LNCS*, pages 92–105. Springer, 2018. doi:10.1007/978-3-030-04414-5\_7.
- 32 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.
- 33 Giuseppe Di Battista and Roberto Tamassia. Algorithms for Plane Representations of Acyclic Digraphs. *Theor. Comput. Sci.*, 61:175–198, 1988. doi:10.1016/0304-3975(88)90123-5.
- 34 Giuseppe Di Battista and Roberto Tamassia. On-Line Planarity Testing. *SIAM Journal on Computing*, 25(5):956–997, 1996.
- 35 Giuseppe Di Battista, Roberto Tamassia, and Ioannis G. Tollis. Area Requirement and Symmetry Display of Planar Upward Drawings. *Discrete & Computational Geometry*, 7:381–401, 1992. doi:10.1007/BF02187850.
- 36 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Curve-constrained drawings of planar graphs. *Comput. Geom.*, 30(1):1–23, 2005. doi:10.1016/j.comgeo.2004.04.002.
- 37 Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, and Stephen K. Wismath. Book Embeddability of Series-Parallel Digraphs. *Algorithmica*, 45(4):531–547, 2006. doi:10.1007/s00453-005-1185-7.
- 38 Emilio Di Giacomo, Francesco Giordano, and Giuseppe Liotta. Upward Topological Book Embeddings of DAGs. *SIAM Journal on Discrete Mathematics*, 25(2):479–489, 2011. doi:10.1137/080731128.
- 39 Emilio Di Giacomo, Giuseppe Liotta, and Francesco Trotta. On Embedding a Graph on Two Sets of Points. *Int. J. Found. Comput. Sci.*, 17(5):1071–1094, 2006.
- 40 Emilio Di Giacomo, Giuseppe Liotta, and Francesco Trotta. Drawing Colored Graphs with Constrained Vertex Positions and Few Bends per Edge. *Algorithmica*, 57(4):796–818, 2010.
- 41 Walter Didimo, Francesco Giordano, and Giuseppe Liotta. Upward Spirality and Upward Planarity Testing. *SIAM J. Discrete Math.*, 23(4):1842–1899, 2009.
- 42 Vida Dujmović and David R. Wood. On Linear Layouts of Graphs. *Discrete Mathematics & Theoretical Computer Science*, 6(2):339–358, 2004.
- 43 Vida Dujmović and David R. Wood. Stacks, Queues and Tracks: Layouts of Graph Subdivisions. *Discrete Mathematics & Theoretical Computer Science*, 7(1):155–202, 2005. URL: <http://dmtcs.episciences.org/346>.
- 44 H. Enomoto and M. S. Miyauchi. Embedding Graphs into a Three Page Book with  $O(m \log n)$  Crossings of Edges over the Spine. *SIAM J. Discrete Math.*, 12(3):337–341, 1999.
- 45 H. Enomoto, M. S. Miyauchi, and K. Ota. Lower Bounds for the Number of Edge-crossings Over the Spine in a Topological Book Embedding of a Graph. *Discrete Applied Mathematics*, 92(2-3):149–155, 1999.
- 46 Hikoe Enomoto, Tomoki Nakamigawa, and Katsuhiko Ota. On the Pagenumber of Complete Bipartite Graphs. *Journal of Combinatorial Theory, Series B*, 71(1):111–120, 1997. doi:10.1006/jctb.1997.1773.
- 47 Hazel Everett, Sylvain Lazard, Giuseppe Liotta, and Stephen K. Wismath. Universal Sets of  $n$  Points for One-bend Drawings of Planar Graphs with  $n$  Vertices. *Discrete & Computational Geometry*, 43(2):272–288, 2010.
- 48 Fedor V. Fomin and Dimitrios M. Thilikos. New upper bounds on the decomposability of planar graphs. *Journal of Graph Theory*, 51(1):53–81, 2006. doi:10.1002/jgt.20121.
- 49 Fabrizio Frati, Radoslav Fulek, and Andres J. Ruiz-Vargas. On the Page Number of Upward Planar Directed Acyclic Graphs. *Journal of Graph Algorithms and Applications*, 17(3):221–244, 2013. doi:10.7155/jgaa.00292.
- 50 Joseph L. Ganley and Lenwood S. Heath. The pagenumber of  $k$ -trees is  $O(k)$ . *Discrete Applied Mathematics*, 109(3):215–221, 2001.

- 51 Ashim Garg and Roberto Tamassia. On the Computational Complexity of Upward and Rectilinear Planarity Testing. *SIAM J. Comput.*, 31(2):601–625, 2001. doi:10.1137/S0097539794277123.
- 52 Francesco Giordano, Giuseppe Liotta, Tamara Mchedlidze, Antonios Symvonis, and Sue Whitesides. Computing upward topological book embeddings of upward planar digraphs. *Journal of Discrete Algorithms*, 30:45–69, 2015.
- 53 L. Heath, F. Leighton, and A. Rosenberg. Comparing Queues and Stacks As Mechanisms for Laying Out Graphs. *SIAM Journal on Discrete Mathematics*, 5(3):398–412, 1992.
- 54 Lenwood S. Heath and Sriram V. Pemmaraju. Stack and Queue Layouts of Posets. *SIAM Journal on Discrete Mathematics*, 10(4):599–625, 1997.
- 55 Lenwood S. Heath and Sriram V. Pemmaraju. Stack and Queue Layouts of Directed Acyclic Graphs: Part II. *SIAM Journal on Computing*, 28(5):1588–1626, 1999.
- 56 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, 1999.
- 57 Seok-Hee Hong and Hiroshi Nagamochi. Simpler algorithms for testing two-page book embedding of partitioned graphs. *Theor. Comput. Sci.*, 725:79–98, 2018.
- 58 L. T. Q. Hung. *A Planar Poset which Requires 4 Pages*. PhD thesis, Institute of Computer Science, University of Wrocław, 1989.
- 59 Maarten Löffler and Csaba D. Tóth. Linear-Size Universal Point Sets for One-Bend Drawings. In *Graph Drawing*, volume 9411 of *LNCS*, pages 423–429. Springer, 2015.
- 60 Seth M. Malitz. Genus  $g$  Graphs Have Pagenumber  $O(\sqrt{g})$ . *J. Algorithms*, 17(1):85–109, 1994.
- 61 Seth M. Malitz. Graphs with  $E$  Edges Have Pagenumber  $O(\sqrt{E})$ . *J. Algorithms*, 17(1):71–84, 1994.
- 62 Sumio Masuda, Kazuo Nakajima, Toshinobu Kashiwabara, and Toshio Fujisawa. Crossing Minimization in Linear Embeddings of Graphs. *IEEE Trans. Computers*, 39(1):124–127, 1990.
- 63 Tamara Mchedlidze and Antonios Symvonis. Crossing-Free Acyclic Hamiltonian Path Completion for Planar  $st$ -Digraphs. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *Algorithms and Computation, ISAAC 2009*, volume 5878 of *LNCS*, pages 882–891. Springer, 2009.
- 64 Tamara Mchedlidze and Antonios Symvonis. Unilateral Orientation of Mixed Graphs. In *SOFSEM 2010*, volume 5901 of *LNCS*, pages 588–599. Springer, 2010.
- 65 Tamara Mchedlidze and Antonios Symvonis. Crossing-Optimal Acyclic HP-Completion for Outerplanar  $st$ -Digraphs. *Journal of Graph Algorithms and Applications*, 15(3):373–415, 2011. doi:10.7155/jgaa.00231.
- 66 Richard Nowakowski and Andrew Parker. Ordered sets, pagenumbers and planarity. *Order*, 6(3):209–218, 1989.
- 67 J. Opatrny. Total Ordering Problem. *SIAM Journal on Computing*, 8(1):111–114, 1979. doi:10.1137/0208008.
- 68 Sriram V. Pemmaraju. *Exploring the Powers of Stacks and Queues via Graph Layouts*. PhD thesis, Virginia Polytechnic Institute and State University at Blacksburg, Virginia, 1992.
- 69 Aimal Rextin and Patrick Healy. Dynamic Upward Planarity Testing of Single Source Embedded Digraphs. *Comput. J.*, 60(1):45–59, 2017. doi:10.1093/comjnl/bxw064.
- 70 Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. doi:10.1016/0095-8956(91)90061-N.
- 71 Paul D. Seymour and Robin Thomas. Call Routing and the Ratcatcher. *Combinatorica*, 14(2):217–241, 1994. doi:10.1007/BF01215352.
- 72 Maciej M. Syslo. Bounds to the Page Number of Partially Ordered Sets. In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science, WG '89*, volume 411 of *LNCS*, pages 181–195. Springer, 1989. doi:10.1007/3-540-52292-1\_13.
- 73 Walter Unger. On the  $k$ -Colouring of Circle-Graphs. In Robert Cori and Martin Wirsing, editors, *STACS 88*, volume 294 of *LNCS*, pages 61–72. Springer, 1988. doi:10.1007/BFb0035832.

## 13:22 Upward Book Embeddings of st-Graphs

- 74 Walter Unger. The Complexity of Colouring Circle Graphs (Extended Abstract). In Alain Finkel and Matthias Jantzen, editors, *STACS 92*, volume 577 of *LNCS*, pages 389–400. Springer, 1992. doi:10.1007/3-540-55210-3\_199.
- 75 Avi Wigderson. The complexity of the Hamiltonian circuit problem for maximal planar graphs. Technical report, 298, EECS Department, Princeton University, 1982.
- 76 David R. Wood. Bounded Degree Book Embeddings and Three-Dimensional Orthogonal Graph Drawing. In *Graph Drawing*, volume 2265 of *LNCS*, pages 312–327. Springer, 2001.
- 77 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.

# Bounded Degree Conjecture Holds Precisely for $c$ -Crossing-Critical Graphs with $c \leq 12$

**Drago Bokal** 

Department of Mathematics and Computer Science, University of Maribor, Maribor, Slovenia  
drago.bokal@um.si

**Zdeněk Dvořák** 

Computer Science Institute, Charles University, Prague, Czech Republic  
rakdver@iuuk.mff.cuni.cz

**Petr Hliněný** 

Faculty of Informatics, Masaryk University, Brno, Czech Republic  
hlineny@fi.muni.cz

**Jesús Leaños** 

Academic Unit of Mathematics, Autonomous University of Zacatecas, Zacatecas, Mexico  
jleanos@matematicas.reduaz.mx

**Bojan Mohar** 

Department of Mathematics, Simon Fraser University, Burnaby BC, Canada  
Institute of Mathematics, Physics, and Mechanics, Ljubljana, Slovenia  
mohar@sfu.ca

**Tilo Wiedera** 

Theoretical Computer Science, Osnabrück University, Germany  
tilo.wiedera@uos.de

---

## Abstract

We study  $c$ -crossing-critical graphs, which are the minimal graphs that require at least  $c$  edge-crossings when drawn in the plane. For every fixed pair of integers with  $c \geq 13$  and  $d \geq 1$ , we give first explicit constructions of  $c$ -crossing-critical graphs containing a vertex of degree greater than  $d$ . We also show that such unbounded degree constructions do not exist for  $c \leq 12$ , precisely, that there exists a constant  $D$  such that every  $c$ -crossing-critical graph with  $c \leq 12$  has maximum degree at most  $D$ . Hence, the bounded maximum degree conjecture of  $c$ -crossing-critical graphs, which was generally disproved in 2010 by Dvořák and Mohar (without an explicit construction), holds true, surprisingly, exactly for the values  $c \leq 12$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Mathematics of computing  $\rightarrow$  Graphs and surfaces

**Keywords and phrases** graph drawing, crossing number, crossing-critical, zip product

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.14

**Related Version** <https://arxiv.org/abs/1903.05363>

**Funding** *Drago Bokal*: ARRS Project J1-8130, ARRS Programme P1-0297.

*Zdeněk Dvořák*: Supported by project 17-04611S of Czech Science Foundation.

*Petr Hliněný*: Supported by Czech Science Foundation, project no. 17-00837S.

*Jesús Leaños*: Partially supported by PFCE-UAZ 2018-2019 grant.

*Bojan Mohar*: B.M. was supported in part by the NSERC Discovery Grant R611450 (Canada), by the Canada Research Chairs program, and by the Research Project J1-8130 of ARRS (Slovenia).

*Tilo Wiedera*: Supported by the German Research Foundation (DFG) project CH 897/2-2.

**Acknowledgements** We acknowledge the supporting environment of the workshop Crossing Numbers: Theory and Applications (18w5029) at the Banff International Research Station, where the fundamentals of this contribution were developed.



© Drago Bokal, Zdeněk Dvořák, Petr Hliněný, Jesús Leaños, Bojan Mohar, and Tilo Wiedera; licensed under Creative Commons License CC-BY

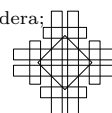
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 14; pp. 14:1–14:15

Leibniz International Proceedings in Informatics



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



## 1 Introduction

Minimizing the number of edge-crossings in a graph drawing in the plane (the *crossing number* of the graph, see Definition 2.1) is considered one of the most important attributes of a “nice drawing” of a graph. In the case of classes of dense graphs (those having superlinear number of edges in terms of the number vertices), the crossing number is necessarily very high – see the famous Crossing Lemma [1, 13]. However, within sparse graph classes (those having only linear number of edges), we may have planar graphs at one end and graphs with up to quadratic crossing number at the other end. In this situation, it is natural to study the “minimal obstructions” for low crossing number, with the following definition.

Let  $c$  be a positive integer. A graph  $G$  is called *c-crossing-critical* if the crossing number of  $G$  is at least  $c$ , but every proper subgraph has crossing number smaller than  $c$ . We say that  $G$  is *crossing-critical* if it is  $c$ -crossing-critical for some positive integer  $c$ .

Since any non-planar graph contains at least one crossing-critical subgraph, the understanding of the properties of the crossing-critical graphs is a central part of the theory of crossing numbers.

In 1984, Širáň gave the earliest construction of nonsimple  $c$ -critical-graphs for every fixed value of  $c \geq 2$  [18]. Three years later, Kochol [11] gave an infinite family of  $c$ -crossing-critical, simple, 3-connected graphs, for every  $c \geq 2$ . Another early result on  $c$ -crossing-critical graphs was reported in the influential paper of Richter and Thomassen [17], who proved that  $c$ -crossing-critical graphs have bounded crossing number in terms of  $c$ . They also initiated research on degrees in  $c$ -crossing-critical graphs by showing that, if there exists an infinite family of  $r$ -regular,  $c$ -crossing-critical graphs for fixed  $c$ , then  $r \in \{4, 5\}$ . Of these, 4-regular 3-critical graphs were constructed by Pinontoan and Richter [16], and 4-regular  $c$ -critical graphs are known for every  $c \geq 3$ ,  $c \neq 4$  [3]. Salazar observed that the arguments of Richter and Thomassen could be applied to average degree as well, showing that an infinite family of  $c$ -crossing-critical graphs of average degree  $d$  can exist only for  $d \in (3, 6)$ , and established their existence for  $d \in [4, 6)$ . Nonexistence of such families with  $d = 6$  was established much later by Hernández, Salazar, and Thomas [9], who proved that, for each fixed  $c$ , there are only finitely many  $c$ -crossing-critical simple graphs of average degree at least six. The existence of such families with  $d \in [\frac{7}{2}, 4)$  was established by Pinontoan and Richter [16], whereas the whole possible interval was covered by Bokal [2], who showed that, for sufficiently large crossing number, both the crossing number  $c$  and the average degree  $d \in (3, 6)$  could be prescribed for an infinite family of  $c$ -crossing critical graphs of average degree  $d$ .

In 2003, Richter conjectured that, for every positive integer  $c$ , there exists an integer  $D(c)$  such that every  $c$ -crossing-critical graph has maximum degree less than  $D(c)$  [14]. Reflecting upon this conjecture, Bokal in 2007 observed that the known 3-connected crossing-critical graphs of that time only had degrees 3, 4, 6, and asked for existence of such graphs with arbitrary other degrees, possibly appearing arbitrarily many times. Hliněný augmented his construction of  $c$ -crossing-critical graphs with pathwidth linear in  $c$  to show the existence of  $c$ -crossing-critical graphs with arbitrarily many vertices of every set of even degrees. Only a recent paper by Bokal, Bračić, Derňár, and Hliněný [3] provided the corresponding result for odd degrees, showing in addition that, for sufficiently high  $c$ , all the three parameters – crossing number  $c$ , rational average degree  $d$ , and the set of degrees  $D \subseteq \mathbb{N} \setminus \{1, 2\}$  that appear arbitrarily often in the graphs of the infinite family – can be prescribed. They also analysed the interplay of these parameters for 2-crossing-critical graphs that were recently completely characterized by Bokal, Oporowski, Richter, and Salazar [5].

Despite all this research generating considerable understanding of the behavior of degrees in known crossing-critical graphs as well as extending the construction methods of such graphs, the original conjecture of Richter was not directly addressed in the previous works. It was, however, disproved by Dvořák and Mohar [8], who showed that, for each integer  $c \geq 171$ , there exist  $c$ -crossing-critical graphs of arbitrarily large maximum degree. Their counterexamples, however, were not constructive, as they only exhibited, for every such  $c$ , a graph containing sufficiently many critical edges incident with a fixed vertex and argued that those edges belong to every  $c$ -crossing-critical subgraph of the exhibited graph. On the other hand, as a consequence of [5] it follows that, except for possibly some small examples, the maximum degree in a large 2-crossing-critical graph is at most 6, implying that Richter's conjecture holds for  $c = 2$ . In view of these results, and the fact that 1-crossing-critical graphs (subdivisions of  $K_5$  and  $K_{3,3}$ ) have maximum degree at most 4, this leaves Richter's conjecture unresolved for each  $c \in \{3, 4, \dots, 170\}$ .

The richness of  $c$ -crossing-critical graphs is restricted for every  $c$  by the result of Hliněný that  $c$ -crossing-critical graphs have bounded path-width [10]; this structural result is complemented by a recent classification of all large  $c$ -crossing-critical graphs for arbitrary  $c$  by Dvořák, Hliněný, and Mohar [7]. We use these results in Section 3 to show that Richter's conjecture holds for  $c \leq 12$ . The result is stated below. It is both precise and surprising and shows how unpredictable are even the most fundamental questions about crossing numbers.

► **Theorem 1.1.** *There exists an integer  $D$  such that, for every positive integer  $c \leq 12$ , every  $c$ -crossing-critical graph has maximum degree at most  $D$ .*

In fact, one can separately consider in Theorem 1.1 twelve upper bounds  $D_c$  for each of the values  $c \in \{1, 2, \dots, 12\}$ . For instance,  $D_1 = 4$  and the optimal value of  $D_2$  (we know  $D_2 \geq 8$ ) should also be within reach using [5] and continuing research. On the other hand, due to the asymptotic nature of our arguments, we are currently not able to give any “nice” numbers for the remaining upper bounds, and we leave this aspect to future investigations.

We cover the remaining values of  $c \geq 13$  in the gap with the following:

► **Theorem 1.2.** *For every positive integer  $d$ , there exists a 3-connected 13-crossing-critical graph  $G(d)$ , whose maximum degree is at least  $d$ .*

► **Corollary 1.3.** *For every two integers  $c \geq 13$  and  $d \geq 1$ , there exists a 3-connected  $c$ -crossing-critical graph  $G(c, d)$ , whose maximum degree is at least  $d$ .*

We also address the related question about the structure of  $c$ -crossing-critical graphs with more than one vertex of large degree. We show the following:

► **Corollary 1.4.** *For any integers  $c \geq 13$ ,  $i \geq 1$  and  $i$ , where  $1 \leq i \leq c/13$ , there exists a 3-connected  $c$ -crossing-critical graph  $G(c, d, i)$ , which contains  $i$  vertices of degree greater than  $d$ .*

Note that, without the 3-connectivity assumption, Corollary 1.4 is established simply by taking disjoint or vertex-identified copies of the graphs from Corollary 1.3.

The paper is structured as follows. The preliminaries, needed to help understanding the structure of large  $c$ -crossing critical graphs are defined in Section 2. We prove Theorem 1.1 in Section 3, and Theorem 1.2 in Section 4. This construction is combined with a new technical operation called 4-to-3 expansion and zip product to establish Corollaries 1.3 and 1.4 in Section 5. We conclude with some remarks and open problems in Section 6.



## 2 Graphs and the crossing number

In this paper, we consider multigraphs by default, even though we could always subdivide parallel edges (while sacrificing 3-connectivity) in order to make our graphs simple. We follow basic terminology of topological graph theory, see e.g. [15].

A *drawing* of a graph  $G$  in the plane is such that the vertices of  $G$  are distinct points and the edges are simple (polygonal) curves joining their end vertices. It is required that no edge passes through a vertex, and no three edges cross in a common point. A *crossing* is then an intersection point of two edges other than their common end. A *face* of the drawing is a maximal connected subset of the plane minus the drawing. A drawing without crossings in the plane is called a *plane drawing* of a graph, or shortly a *plane graph*. A graph having a plane drawing is *planar*.

The following are the core definitions used in this work.

► **Definition 2.1** (crossing number). *The crossing number  $\text{cr}(G)$  of a graph  $G$  is the minimum number of crossings of edges in a drawing of  $G$  in the plane. An optimal drawing of  $G$  is every drawing with exactly  $\text{cr}(G)$  crossings.*

► **Definition 2.2** (crossing-critical). *Let  $c$  be a positive integer. A graph  $G$  is  $c$ -crossing-critical if  $\text{cr}(G) \geq c$ , but every proper subgraph  $G'$  of  $G$  has  $\text{cr}(G') < c$ .*

Let us remark that a  $c$ -crossing-critical graph may have no drawing with only  $c$  crossings (for  $c = 2$ , such an example is the Cartesian product of two 3-cycles,  $C_3 \square C_3$ ).

Suppose  $G$  is a graph drawn in the plane with crossings. Let  $G'$  be the plane graph obtained from this drawing by replacing the crossings with new vertices of degree 4. We say that  $G'$  is the plane graph associated with the drawing, shortly the *planarization* of (the drawing of)  $G$ , and the new vertices are the *crossing vertices* of  $G'$ .

**Preliminaries.** In some of our constructions, we will have to combine crossing-critical graphs as described in the next definition.

► **Definition 2.3.** *Let  $d = 2$  or  $3$ . For  $i \in \{1, 2\}$ , let  $G_i$  be a graph and let  $v_i \in V(G_i)$  be a vertex of degree  $d$  that is only incident with simple edges, such that  $G_i - v_i$  is connected. Let  $u_i^j$ ,  $j \in \{1, \dots, d\}$  be the neighbors of  $v_i$ . The zip product of  $G_1$  and  $G_2$  at  $v_1$  and  $v_2$  is obtained from the disjoint union of  $G_1 - v_1$  and  $G_2 - v_2$  by adding the edges  $u_1^j u_2^j$ , for each  $j \in \{1, \dots, d\}$ .*

Note that, for different labellings of the neighbors of  $v_1$  and  $v_2$ , different graphs may result from the zip product. However, the following has been shown:

► **Theorem 2.4** ([4]). *Let  $G$  be a zip product of  $G_1$  and  $G_2$  as in Definition 2.3. Then,  $\text{cr}(G) = \text{cr}(G_1) + \text{cr}(G_2)$ . Furthermore, if for both  $i = 1$  and  $i = 2$ ,  $G_i$  is  $c_i$ -crossing-critical, where  $c_i = \text{cr}(G_i)$ , then  $G$  is  $(c_1 + c_2)$ -crossing-critical.*

For vertices of degree 2, this theorem was established already by Leños and Salazar in [12].

Dvořák, Hliněný, and Mohar [7] recently characterized the structure of large  $c$ -crossing-critical graphs. From their result, it can be derived that in a crossing-critical graph with a vertex of large degree, there exist many internally vertex-disjoint paths from this vertex to the boundary of a single face, see the following corollary for a more precise formulation. To keep our contribution self-contained, we also give a simple independent proof of this claim in the full version of the paper.



► **Corollary 2.5.** *There exists a function  $f_{2.5} : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that the following holds. Let  $c \geq 1$  and  $t \geq 3$  be integers and let  $G$  be an optimal drawing of a 2-connected  $c$ -crossing-critical graph. If  $G$  has maximum degree greater than  $f_{2.5}(c, t)$ , then there exists a path  $Q$  contained in the boundary of a face of  $G$  and internally vertex-disjoint paths  $P_1, \dots, P_t$  starting in the same vertex not in  $Q$  and ending in distinct vertices appearing in order on  $Q$  (and otherwise disjoint from  $Q$ ), such that no crossings of  $G$  appear on  $P_1, P_t$ , nor in the face of  $P_1 \cup P_t \cup Q$  that contains  $P_2, \dots, P_{t-1}$ .*

### 3 Crossing-critical graphs with at most 12 crossings

We now use Corollary 2.5 to prove the following “redrawing” lemma.

► **Lemma 3.1.** *Let  $G$  be a 2-connected  $c$ -crossing-critical graph. If  $G$  has maximum degree greater than  $f_{2.5}(c, 6c + 1)$ , then there exist integers  $r \geq 2$  and  $k \geq 0$  such that  $kr \leq c - 1$  and  $G$  has a drawing with at most  $c - 1 - kr + \binom{k}{2}$  crossings.*

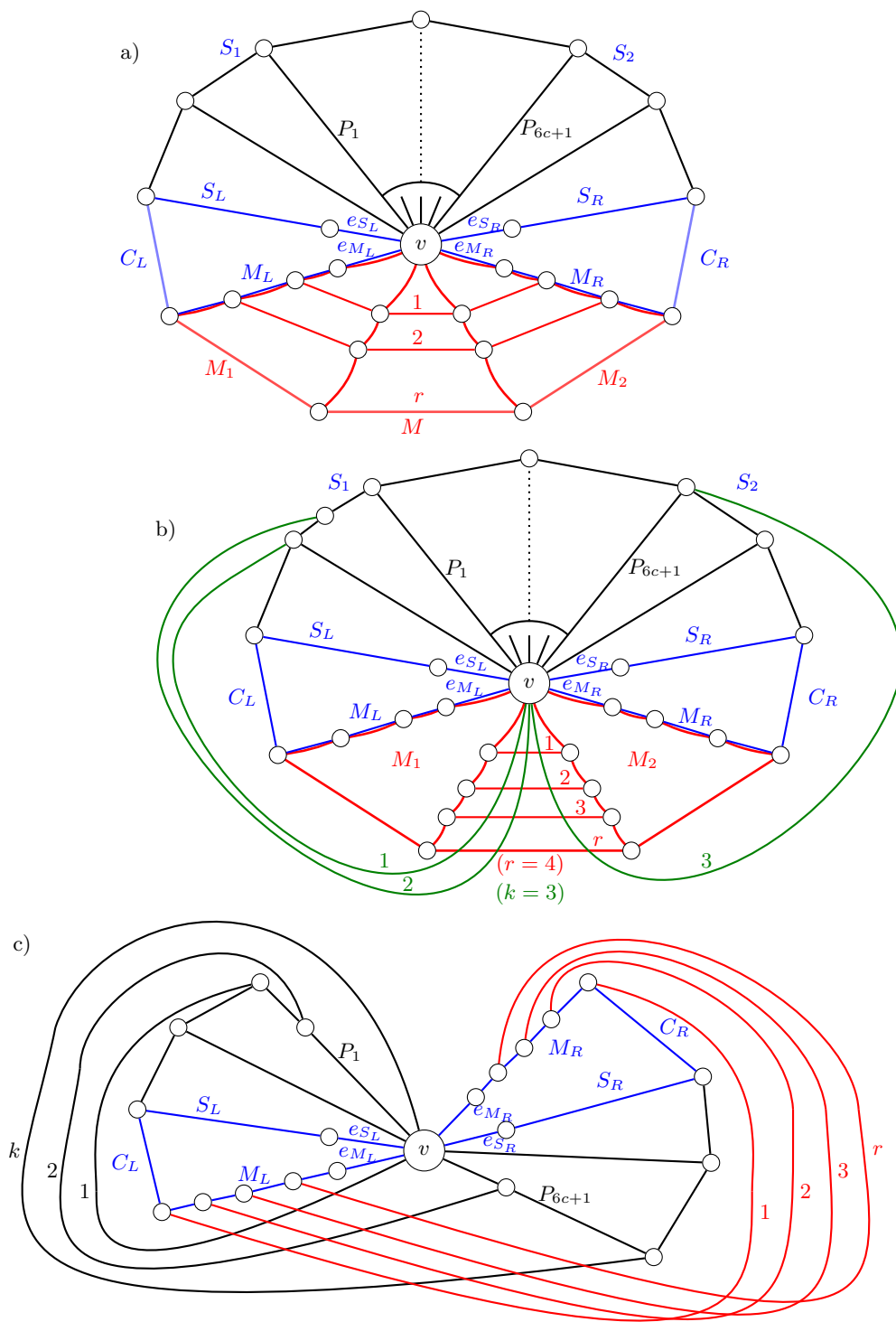
**Proof.** Consider an optimal drawing of  $G$ . Let  $P_1, \dots, P_{6c+1}$  be paths obtained using Corollary 2.5 and  $v$  their common end vertex. For  $2 \leq i \leq 6c - 1$ , let  $T_i$  denote the 2-connected block of  $G - ((V(P_{i-1}) \cup V(P_{i+2})) \setminus \{v\})$  containing  $P_i$  and  $P_{i+1}$ , and let  $C_i$  denote the cycle bounding the face of  $T_i$  containing  $P_{i-1}$ . Note that if  $2 \leq i$  and  $i + 3 \leq j \leq 6c - 1$ , then  $G - V(T_i \cup T_j)$  has at most three components: one containing  $P_{i+2} - v$ , one containing  $P_1 - v$ , and one containing  $P_{6c+1} - v$ , where the latter two components can be the same.

Let  $e$  be the edge of  $P_{3c+1}$  incident with  $v$  and let  $G'$  be an optimal drawing of  $G - e$ . Since  $G$  is  $c$ -crossing-critical,  $G'$  has at most  $c - 1$  crossings. Hence, there exist indices  $i_1$  and  $i_2$  such that  $2 \leq i_1 \leq 3c - 1$ ,  $3c + 2 \leq i_2 \leq 6c - 1$ , and none of the edges of  $T_{i_1}$  and  $T_{i_2}$  is crossed. Let us set  $L = T_{i_1}$ ,  $C_L = C_{i_1}$ ,  $R = T_{i_2}$ , and  $C_R = C_{i_2}$ . Let  $M, S_1$ , and  $S_2$  denote the subgraphs of  $G$  consisting of the components of  $G - V(L \cup R)$  containing  $P_{3c+1} - v$ ,  $P_1 - v$ , and  $P_{6c+1} - v$ , respectively, together with the edges from these components to the rest of  $G$  and their incident vertices (where possibly  $S_1 = S_2$ ). Let  $S_L$  and  $M_L$  be subpaths of  $C_L$  of length at least one intersecting in  $v$  such that  $V(S_1 \cap C_L) \subseteq V(S_L)$  and  $V(M \cap C_L) \subseteq V(M_L)$ . Analogously, let  $S_R$  and  $M_R$  be subpaths of  $C_R$  of length at least one intersecting in  $v$  such that  $V(S_2 \cap C_R) \subseteq V(S_R)$  and  $V(M \cap C_R) \subseteq V(M_R)$ . See Figure 1.

We can assume without loss of generality (by circle inversion of the plane if necessary) that neither  $C_L$  nor  $C_R$  bounds the outer face of  $C_L \cup C_R$  in the drawings inherited from  $G$  and from  $G'$ . Let  $e_{M_L}, e_{S_L}, e_{S_R}, e_{M_R}$  be the clockwise cyclic order of the edges of  $C_L \cup C_R$  incident with  $v$  in the drawing  $G$ , where  $e_Q \in E(Q)$  for every  $Q \in \{M_L, S_L, S_R, M_R\}$ . By the same argument, we can assume that the clockwise cyclic order of these edges in the drawing of  $G'$  is either the same or  $e_{M_L}, e_{S_L}, e_{M_R}, e_{S_R}$ .

In  $G$ ,  $L$  is drawn in the closed disk bounded by  $C_L$ ,  $R$  is drawn in the closed disk bounded by  $C_R$ , and  $M, S_1$ , and  $S_2$  together with all the edges joining them to  $v$  are drawn in the outer face of  $C_L \cup C_R$ . Since  $C_L$  and  $C_R$  are not crossed in the drawing  $G'$ , we can if necessary rearrange the drawing of  $G'$  without creating any new crossings<sup>1</sup> so that the same holds for the drawings of  $L, R, M, S_1$ , and  $S_2$  in  $G'$ . Let  $r \geq 1$  denote the maximum number of pairwise

<sup>1</sup> As  $G$  is not 3-connected, it is possible that some 2-connected components or some edges of  $L, R$  are drawn in the exterior of the disk bounded by  $C_L, C_R$ . However, these can be flipped into the interior of  $C_L, C_R$ , and after such rearranging,  $C_L, C_R$  bound the outer face of the drawings of  $L, R$ . Similarly, if  $S_1 \neq S_2$ , either of them could be in the interior of  $C_L, C_R$ , and we flip them into the exterior, so that the interior of  $C_L, C_R$  contains only drawings of  $L, R$ , respectively.



■ **Figure 1** An illustration of the proof of Lemma 3.1. a) The original optimal drawing of  $G$ , with subdrawings of  $M_1$  and  $M_2$  (red) that will be glued into the drawing of  $G_0$  from an optimal drawing of  $G - e$ . b) A drawing of  $G$  with at most  $c - 1$  crossings, obtained from  $G_0$  (black, blue, green) and  $M_1, M_2$  (red). c) A drawing of  $G$  with at most  $\binom{c-1-kr+k}{2}$  crossings, obtained from  $G_0$  (black, blue) and  $M_1, M_2$  (red).

edge-disjoint paths in  $M - v$  from  $V(M \cap C_L - v)$  to  $V(M \cap C_R - v)$ . By Menger's theorem,  $M - v$  has disjoint induced subgraphs  $M'_1$  and  $M'_2$  such that  $V(M - v) = V(M'_1) \cup V(M'_2)$ ,  $V(M \cap C_L - v) \subseteq V(M'_1)$ ,  $V(M \cap C_R - v) \subseteq V(M'_2)$ , and  $G$  contains exactly  $r$  edges with one end in  $M'_1$  and the other end in  $M'_2$ . For  $i \in \{1, 2\}$ , let  $M_i$  be the subgraph of  $M$  induced by  $V(M'_i) \cup \{v\}$ . Let  $F$  be a path in  $M - v$  from  $V(M \cap C_L - v)$  to  $V(M \cap C_R - v)$  that has in the drawing  $G'$  the smallest number of intersections with the edges of  $S_1 \cup S_2$ , and let  $k$  denote the number of such intersections. Let  $G_0$  denote the drawing  $G' - (V(M) \setminus V(M_L \cup M_R))$ . Since  $M - v$  contains  $r$  pairwise edge-disjoint paths from  $V(M \cap C_L - v)$  to  $V(M \cap C_R - v)$  and each of them crosses  $S_1 \cup S_2$  at least  $k$  times, we conclude that  $G'$  has at least  $kr$  crossings (and thus  $kr \leq c - 1$ ) and  $G_0$  has at most  $c - 1 - kr$  crossings.

Suppose first that edges of  $C_L \cup C_R$  incident with  $v$  are in  $G'$  drawn in the same clockwise cyclic order as in  $G$ . We construct a new drawing of the graph  $G$  in the following way: Start with the drawing of  $G_0$ . Take the plane drawings of  $M_1$  and  $M_2$  as in  $G$ , "squeeze" them and draw them very close to  $M_L$  and  $M_R$ , respectively, so that they do not intersect any edges of  $G_0$ . Finally, draw the  $r$  edges between  $M_1$  and  $M_2$  very close to the curve tracing  $F$  (as drawn in  $G'$ ), so that each of them is crossed at most  $k$  times. This gives a drawing of  $G$  with at most  $(c - 1 - kr) + kr < c$  crossings, contradicting the assumption that  $G$  is  $c$ -crossing-critical.

Hence, we can assume that the edges of  $C_L \cup C_R$  incident with  $v$  are in  $G'$  drawn in the clockwise order  $e_{M_L}, e_{S_L}, e_{M_R}, e_{S_R}$ . If  $r = 1$ , then proceed analogously to the previous paragraph, except that a mirrored version <sup>2</sup> of the drawing of  $M_2$  is inserted close to  $M_R$ ; as there is only one edge between  $M_1$  and  $M_2$ , this does not incur any additional crossings, and we again conclude that the resulting drawing of  $G$  has fewer than  $c$  crossings, a contradiction. Therefore,  $r \geq 2$ .

Consider the drawing  $G'$ , and let  $q$  be a closed curve passing through  $v$ , following  $M_L$  slightly outside  $C_L$  till it meets  $F$ , then following  $F$  almost till it hits  $M_R$ , then following  $M_R$  slightly outside  $C_R$  till it reaches  $v$ . Note that  $q$  only crosses  $G_0$  in  $v$  and in relative interiors of the edges, and it has at most  $k$  crossings with the edges. Shrink and mirror the part of the drawing of  $G_0$  drawn in the open disk bounded by  $q$ , keeping  $v$  at the same spot and the parts of edges crossing  $q$  close to  $q$ ; then reconnect these parts of the edges with their parts outside of  $q$ , creating at most  $\binom{k}{2}$  new crossings in the process. Observe that in the resulting re-drawing of  $G_0$ , the path  $M_L \cup M_R$  is contained in the boundary of a face (since  $q$  is drawn close to it and nothing crosses this part of  $q$ ), and thus we can add  $M$  planarly (as drawn in  $G$ ) to the drawing without creating any further crossings. Therefore, the resulting drawing has at most  $c - 1 - kr + \binom{k}{2}$  crossings. ◀

It is now easy to prove Theorem 1.1.

**Proof of Theorem 1.1.** We prove by induction on  $c$  that, for every positive integer  $c \leq 12$ , there exists an integer  $\Delta_c$  such that every  $c$ -crossing-critical graph has maximum degree at most  $c$ . The only 1-crossing-critical graphs are subdivisions of  $K_5$  and  $K_{3,3}$ , and thus we can set  $\Delta_1 = 4$ . Suppose now that  $c \geq 2$  and the claim holds for every smaller value. We define  $\Delta_c = \max(2\Delta_{c-1}, f_{2.5}(c, 6c + 1))$ . Let  $G$  be a  $c$ -crossing-critical graph and suppose for a contradiction that  $\Delta(G) > \Delta_c$ .

<sup>2</sup> *Mirrored version of a drawing* is the drawing obtained by reversing the vertex rotations of edges around every vertex and every crossing, and embedding the edges and the vertices accordingly. The name explains that this is homeomorphic to the original drawing seen in a mirror.

If  $G$  is not 2-connected, then it contains induced subgraphs  $G_1$  and  $G_2$  such that  $G_1 \neq G \neq G_2$ ,  $G = G_1 \cup G_2$ , and  $G_1$  intersects  $G_2$  in at most one vertex. Then  $c \leq \text{cr}(G) = \text{cr}(G_1) + \text{cr}(G_2)$ , and for every edge  $e \in E(G_1)$  we have  $c > \text{cr}(G - e) = \text{cr}(G_1 - e) + \text{cr}(G_2)$ . Hence,  $\text{cr}(G_1) \geq c - \text{cr}(G_2)$  and  $\text{cr}(G_1 - e) < c - \text{cr}(G_2)$  for every edge  $e \in E(G_1)$ , and thus  $G_1$  is  $(c - \text{cr}(G_2))$ -crossing-critical. Similarly,  $G_2$  is  $(c - \text{cr}(G_1))$ -crossing-critical. Since  $\text{cr}(G_1) \geq 1$  and  $\text{cr}(G_2) \geq 1$ , it follows by the induction hypothesis that  $\Delta(G_i) \leq \Delta_{c-1}$  for  $i \in \{1, 2\}$ , and thus  $\Delta(G) \leq \Delta_c$ , which is a contradiction.

Hence,  $G$  is 2-connected. By Lemma 3.1, there exist integers  $r \geq 2$  and  $k \geq 0$  such that  $kr \leq c - 1$  and  $c - 1 - kr + \binom{k}{2} \geq c$ , and thus  $\binom{k}{2} \geq kr + 1 \geq 2k + 1$ . This inequality is only satisfied for  $k \geq 6$ , and thus the first inequality implies  $c \geq kr + 1 \geq 13$ . This is a contradiction. Hence, the maximum degree of  $G$  is at most  $\Delta_c$ . ◀

#### 4 Explicit 13-crossing-critical graphs with large degree

We define the following family of graphs, which is illustrated in Figure 2. To simplify the terminology and the pictures, we introduce “thick edges”: for a positive integer  $t$ , we say that  $uv$  is a  $t$ -thick edge, or an edge of thickness  $t$ , if there is a bunch of  $t$  parallel edges between  $u$  and  $v$ . Naturally, if a  $t_1$ -thick edge crosses a  $t_2$ -thick edge, then this counts as  $t_1 t_2$  ordinary crossings. By routing every parallel bunch of edges along the “cheapest” edge of the bunch, we get the following important folklore claim:

▷ Claim 4.1. For every graph  $G$ , there exists an optimal drawing  $\mathcal{D}$  of  $G$ , such that every bunch of parallel edges is drawn as one thick edge in  $\mathcal{D}$ .

► **Definition 4.2** (Critical family  $\{G_{13}^k\}$ ). Let  $k \geq 2$  be an integer. Let  $C_u$  be a 6-cycle on the vertex set  $\{x, u_1, u_2, u_3, u_4, u_5\}$  with (thick) edges  $xu_1, u_1u_2, u_2u_3, u_3u_4, u_4u_5, u_5x$  which are of thickness 7, 5, 4, 4, 4, 1 in this order. Analogously, let  $C_v$  be a 6-cycle on the vertex set  $\{x, v_1, v_2, v_3, v_4, v_5\}$  isomorphic to  $C_u$  in this order of vertices. We denote by  $B$  the graph obtained from the union  $C_u \cup C_v$  (identifying their vertex  $x$ ) by adding edges  $u_2v_3$  and  $u_3v_2$ , and 2-thick edges  $u_1v_4$  and  $u_4v_1$ .

Let  $D_i$ , for  $i \in \{1, \dots, k\}$ , denote the graph on the vertex set  $\{x, w_1^i, w_2^i, w_3^i, w_4^i\}$  with the edges  $xw_1^i, xw_4^i, w_1^i w_4^i, w_2^i w_3^i$  and the 2-thick edges  $w_1^i w_2^i$  and  $w_3^i w_4^i$ . From the union  $B \cup D_1 \cup \dots \cup D_k$  (again identifying their vertex  $x$ ), we obtain the graph  $G_{13}^k$  via

- identifying  $u_5$  with  $w_2^1$  and  $w_3^k$  with  $v_5$ , and
- for  $i = 2, 3, \dots, k$ , identifying  $w_3^{i-1}$  with  $w_2^i$ .

This definition is illustrated in Figure 2. For reference, we will call the graph  $B$  the bowtie of  $G_{13}^k$ , and the graph  $D_i$  the  $i$ -th wedge of  $G_{13}^k$ .

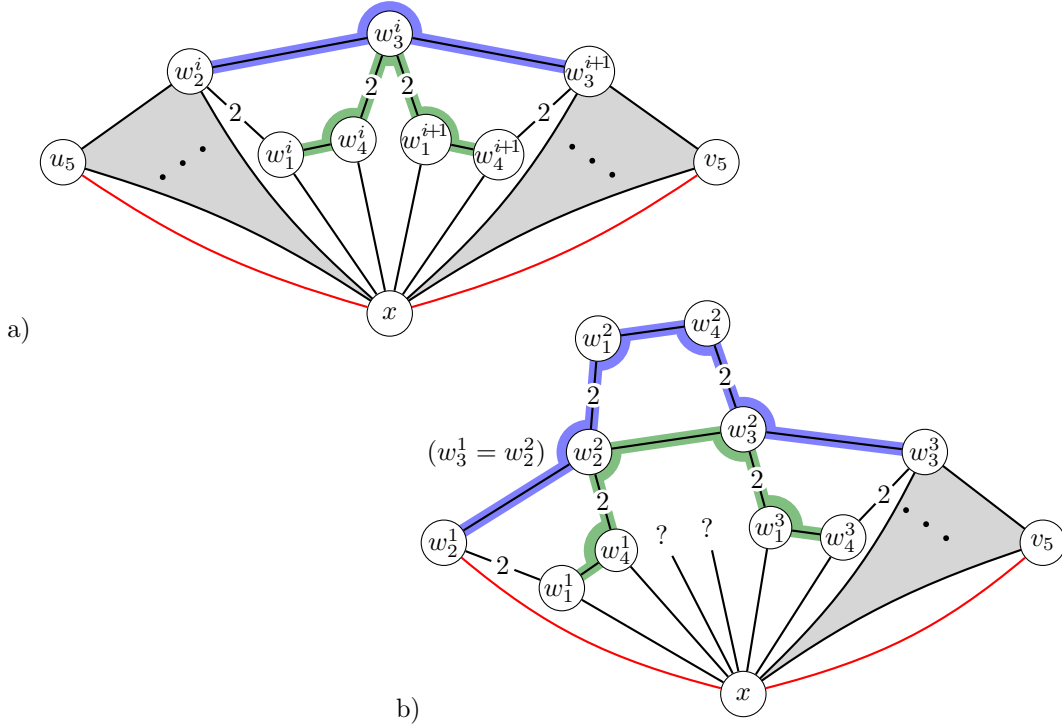
► **Observation 4.3.**

- a) For every  $k \geq 2$ , the graph  $G_{13}^k$  is a 3-connected and non-planar.
- b) The degree of the vertex  $x$  in  $G_{13}^k$  equals  $2k + 16$ .
- c) There exists an automorphism of  $G_{13}^k$  exchanging  $u_i$  with  $v_i$ , for  $i = 1, 2, 3, 4, 5$ .

In order to prove Theorem 1.2, e.g. for  $G(d) = G_{13}^{\lfloor d/2 \rfloor}$ , it suffices to show two claims; that  $\text{cr}(G_{13}^k) \geq 13$  for  $k \geq 2$ , and that, for every edge  $e$  of  $G_{13}^k$ , we get  $\text{cr}(G_{13}^k - e) \leq 12$ . (We also remark that  $\text{cr}(G_{13}^1) \leq 12$ , and for this reason we assume  $k \geq 2$ .)

► **Lemma 4.4.**  $\text{cr}(G_{13}^2) = \text{cr}(G_{13}^3) = 13$ .





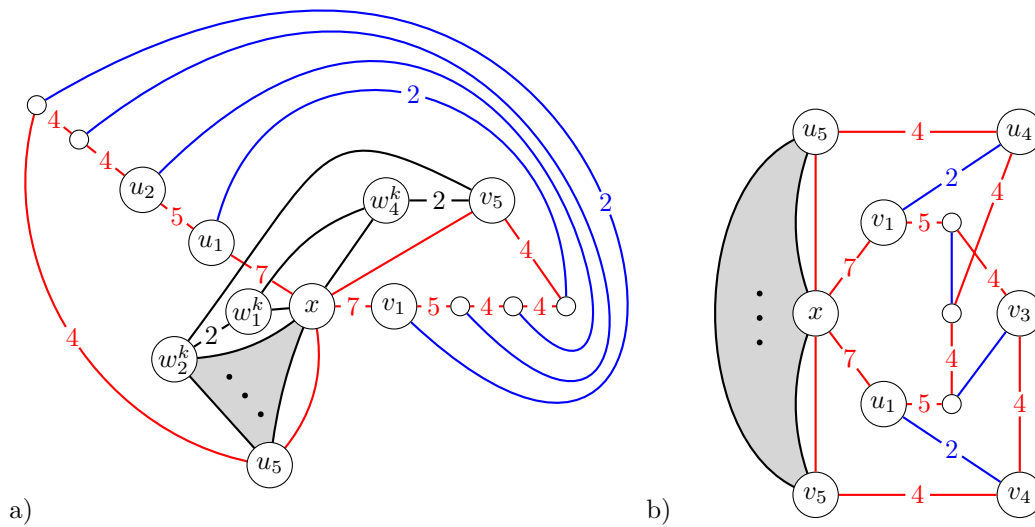
■ **Figure 3** Two cases of the induction step in the proof of Lemma 4.5. (a) We “shrink” two wedges into one by drawing new edges  $w_1^i w_4^{i+1}$  (green) and  $w_2^i w_3^{i+1}$  (blue) along the depicted paths. (b) We likewise “shrink” three wedges into one by drawing the depicted new edges  $w_1^1 w_4^3$  and  $w_2^1 w_3^3$  (this picture does not specify how  $w_1^2$  and  $w_4^2$  connect to  $x$  since it is not important for us).

- There exists  $i \in \{1, \dots, k-1\}$ , such that the edges incident to  $w_3^i = w_2^{i+1}$ , in a small neighbourhood of  $w_3^i$ , have the cyclic order  $w_3^i w_4^i, w_3^i w_1^{i+1}, w_3^i w_3^{i+1}, w_3^i w_2^i$ . See in Figure 3 a, where this cyclic order is anti-clockwise. In this case, we draw a new edge  $w_1^i w_4^{i+1}$  along the path  $(w_1^i, w_4^i, w_3^i, w_1^{i+1}, w_4^{i+1})$ , and another new edge  $w_2^i w_3^{i+1}$  along the path  $(w_2^i, w_3^i, w_3^{i+1})$  (both new edges are of thickness 1). Then we delete the vertices  $w_4^i, w_3^i, w_1^{i+1}$  together with incident edges. The resulting drawing represents a graph which is clearly isomorphic to  $G_{13}^{k-1}$  – the wedges  $i$  and  $i+1$  have been replaced with one wedge.

Moreover, thanks to the assumption, we can avoid crossing between  $w_1^i w_4^{i+1}$  and  $w_2^i w_3^{i+1}$  in the considered neighbourhood of former  $w_3^i$ . Therefore, every crossing of the new drawing (including possible crossings of each of the new edges  $w_1^i w_4^{i+1}$  and  $w_2^i w_3^{i+1}$  among themselves or with other edges) existed already in the original drawing of  $G_{13}^k$ , and so  $\text{cr}(G_{13}^{k-1}) \leq c$ . However,  $\text{cr}(G_{13}^{k-1}) \geq 13$  by the induction assumption, and so  $c \geq 13$  holds true in this case.

- The same proof as above works if the cyclic order around  $w_3^i$  is  $w_3^i w_4^i, w_3^i w_1^{i+1}, w_3^i w_2^i, w_3^i w_3^{i+1}$ .
- For all  $i \in \{1, \dots, k-1\}$ , in a small neighbourhood of  $w_3^i$ , the edges incident to  $w_3^i = w_2^{i+1}$  have (up to orientation reversal) the cyclic order  $w_3^i w_4^i, w_3^i w_3^{i+1}, w_3^i w_1^{i+1}, w_3^i w_2^i$ . See Figure 3 b. We will use this assumption only for  $i = 1, 2$  as follows.

We draw a new edge  $w_1^1 w_4^3$  along the path  $(w_1^1, w_4^1, w_3^1, w_3^3, w_1^3, w_4^3)$ , and another new edge  $w_2^1 w_3^3$  along the path  $(w_2^1, w_3^1, w_2^2, w_4^2, w_3^2, w_3^3)$  (both new edges are of thickness 1). Then we delete the vertices  $w_4^1, w_3^1, w_2^2, w_4^2, w_3^2, w_1^3$  together with incident edges. The resulting



■ **Figure 4** Two drawings of the graph  $G_{13}^k$ , having (a) 14 and (b) 16 crossings. These drawings are used to argue criticality of some of the bowtie (red) edges of  $G_{13}^k$ . The grey areas span the crossing-free wedges of  $G_{13}^k$  which are not detailed in the pictures, similarly as in Figure 2.

drawing represents a graph which is now isomorphic to  $G_{13}^{k-2}$  – the wedges 1, 2 and 3 have been replaced with one wedge.

As in the previous case, we can avoid crossing between  $w_1^1 w_4^3$  and  $w_2^1 w_3^3$  in the considered neighbourhoods of former  $w_3^1$  and  $w_3^2$ . Therefore, analogously,  $\text{cr}(G_{13}^{k-2}) \leq c$ . However,  $k - 2 \geq 2$  and so  $\text{cr}(G_{13}^{k-2}) \geq 13$  by the induction assumption, and hence  $c \geq 13$  holds true also in this case.

By induction, for every integer  $k \geq 2$ ,  $\text{cr}(G_{13}^k) = c \geq 13$  holds. ◀

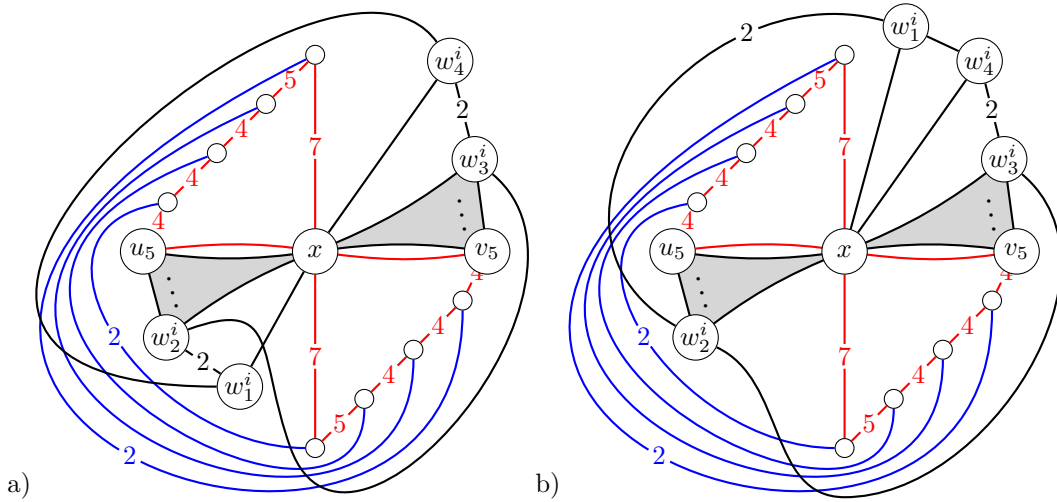
► **Lemma 4.6.** *For every edge  $e$  of  $G_{13}^k$ , we have  $\text{cr}(G_{13}^k - e) \leq 12$ .*

**Proof.** On a high level, our proof strategy can be described as follows. We provide a collection of drawings of  $G_{13}^k$ , such that each edge of  $G_{13}^k$  in some of the drawings, when deleted, exhibits a “drop” of the crossing number below 13. Note that, for thick edges, we are deleting only one edge of the multiple bunch.

In particular, the drawing of Figure 2 proves the claim for  $e \in \{u_1 v_4, u_2 v_3, u_3 v_2, u_4 v_1\}$  (the blue edges of the bowtie subgraph); whenever any one of these edges is removed, we save at least one crossing from the optimal number 13. Likewise, Figure 2 proves the claim for  $e \in \{x v_5, v_4 v_5\}$  (see the dotted routings of the edge  $v_1 u_4$ ), and hence also for  $e \in \{x u_5, u_4 u_5\}$  by symmetry (the automorphism from Observation 4.3).

To proceed with the remaining edges  $e$  of the bowtie graph  $B \subseteq G_{13}^k$  (the red edges), we resort to drawings that may have more than 13 crossings but still provide a drawing of  $G_{13}^k - e$  with at most 12 crossings. Consider first the drawing obtained from the one of Figure 2 by “flipping” the right-hand part of the picture along the line through  $x$  and  $u_5$ . See Figure 4 a. In this drawing (with 14 crossings), the only crossings occur between the 7-thick edge  $x u_1$  and the edges  $w_1^k w_4^k, w_2^k v_5$ . By a slight shift of the latter two edges, we obtain another drawing with only 14 crossing between the 5- and 2-thick edges  $u_1 u_2, u_1 v_4$  and again the edges  $w_1^k w_4^k, w_2^k v_5$ . Removing an edge of the bunch  $x u_1$  (respectively, one edge of  $u_1 u_2$ ) drops the crossing number down to 12. Second, there is a drawing with 16 crossing, which occur only between the 4-thick edges  $v_2 v_3$  and  $u_3 u_4$ ; see Figure 4 b. Deleting any one





■ **Figure 5** Two drawings of the graph  $G_{13}^k$ , having (a) 13 and (b) 18 crossings. These drawings are used to argue criticality of edges of the  $i$ -th wedge. The grey areas span the crossing-free wedges of  $G_{13}^k$  which are not detailed in the pictures, similarly as in Figure 4.

edge of  $v_2v_3, u_3u_4$  again drops the crossing number down to at most 12. Consequently, our claim holds for  $e \in \{xu_1, u_1u_2, v_2v_3, u_3u_4\}$  and, by symmetry, for  $e \in \{xv_1, v_1v_2, u_2u_3, v_3v_4\}$ .

We are left with the last, and perhaps most interesting, cases in which  $e$  is an edge in the  $i$ -th wedge  $D_i$ . Imagine we “disconnect”  $D_i$  by removing vertices  $w_1^i$  and  $w_4^i$  and the edge  $w_2^i w_3^i$ , and then twist the bowtie graph  $B$  together with the adjacent strips of wedges  $D_1 \cup \dots \cup D_{i-1}$  and  $D_{i+1} \cup \dots \cup D_k$ , removing all crossings. If we introduce the vertices and edges of  $D_i$  back, we get a drawing in Figure 5 a with 13 crossings which are only between the edges  $xw_1^i, w_2^i w_3^i, w_1^i w_4^i$  and the six blue bowtie edges. For every edge  $e \in \{xw_1^i, w_2^i w_3^i, w_1^i w_4^i\}$  its removing from  $G_{13}^k$  thus decreases the crossing number to at most 12. Lastly, we may move the vertex  $w_1^i$  to obtain another drawing as in Figure 5 b. The latter drawing has 18 crossings between the edges  $w_1^i w_2^i, w_2^i w_3^i$  and the six blue bowtie edges. Removing one edge of  $w_1^i w_2^i$  thus drops the crossing number down to 12, again. With help of symmetry, the claim is thus proved also for every edge  $e \in E(D_i)$ , for each  $i \in \{1, 2, \dots, k\}$ . ◀

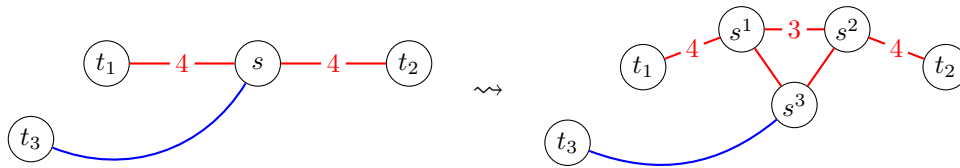
For  $d \geq 4$ , Theorem 1.2 is then established with  $G(d) := G_{13}^{\lfloor d/2 \rfloor}$ .

## 5 Extended crossing-critical constructions

In the previous section, we have constructed an infinite family of 13-crossing-critical graphs with unbounded maximum degree. There are two further natural questions to be asked; (a) what about analogous  $c$ -crossing-critical families for  $c > 13$ , and (b) what about constructing  $c$ -crossing-critical graphs with more than one high-degree vertex?

Clearly, disjoint union of a graph  $G_{13}^k$  with  $c-13$  disjoint copies of  $K_{3,3}$  yields a (disconnected)  $c$ -crossing-critical graph with maximum degree greater than  $d$ , for every  $c \geq 14$ . Similarly, concerning (b), we can consider disjoint union of  $t$  copies of  $G_{13}^k$  to get a  $13t$ -crossing-critical graph with  $t$  vertices of arbitrarily high degree. Though, our aim is to preserve also the 3-connectivity property of the resulting graphs.

First, to motivate the coming construction, we recall that the zip product of Definition 2.3 requires a vertex of degree 3 in the considered graphs. However, the graph  $G_{13}^k$  of Definition 4.2 has no such vertex, and so we come with the following modification.



■ **Figure 6** An illustration of the 4-to-3 expansion of the vertex  $s$  in a graph  $G$ . Clearly, for every optimal drawing of  $G$  respecting Claim 4.1, this “split” construction can be performed in a small neighbourhood of  $s$  without introducing additional crossings.

► **Definition 5.1** (Critical family  $\{H_{13}^k\}$ ). Let  $G$  be a graph and  $s \in V(G)$  be a vertex incident exactly with two 4-thick edges  $st_1, st_2$ , and one ordinary edge  $st_3$ . We call a 4-to-3 expansion of  $s$  the following operation: in  $G$ , remove the vertex  $s$  with its incident edges, and add three new vertices  $s^1, s^2, s^3$ , two 4-thick edges  $s^1t_1, s^2t_2$ , one 3-thick edge  $s^1s^2$ , and three ordinary edges  $s^3t_1, s^3t_2, s^3t_3$ . See the sketch in Figure 6.

Now, for  $k \geq 2$ , the graph  $H_{13}^k$  is constructed from the graph  $G_{13}^k$  of Definition 4.2 by a 4-to-3 expansion of the vertex  $v_3$  and of the vertex  $u_3$  (cf. Figure 2).

The proof of the following technical Lemma is present in the full paper.

► **Lemma 5.2.** Let  $G$  be a 13-crossing-critical graph, and let  $s \in V(G)$  be a vertex incident exactly with two 4-thick edges and one ordinary edge in  $G$ . If  $G_1$  is a graph obtained by a 4-to-3 expansion of  $s$ , then  $G_1$  is also 13-crossing-critical.

► **Remark 5.3.** The number 13 of crossings in Lemma 5.2 is rather special and the claim cannot be easily generalized to other numbers of crossings. For instance, one can construct a graph of crossing number 14, such that one of its 4-to-3 expansions has crossing number only 13.

► **Corollary 5.4.** For every  $k \geq 2$ , the graph  $H_{13}^k$  is 13-crossing-critical.

**Proof.** We start with Theorem 1.2, and apply Lemma 5.2 to the vertices  $v_3$  and  $u_3$  of  $G_{13}^k$ . ◀

**Proof of Corollary 1.3.** For  $c = 13$ , we set  $G(13, d) := H_{13}^{\lfloor d/2 \rfloor}$  from Corollary 5.4. Note that  $G(13, d)$  contains two vertices of degree 3. For  $c > 13$ , we proceed by induction, assuming that we have already constructed the graph  $G(c - 1, d)$  and it contains a vertex of degree 3. Theorem 2.4 establishes that  $G(c, d)$ , as a zip product of  $G(c - 1, d)$  with  $K_{3,3}$ , is  $c$ -crossing-critical (and it still contains the same vertex  $x$  of high degree as  $H_{13}^k$  does). Furthermore,  $G(c, d)$  contains a vertex of degree 3 coming from the  $K_{3,3}$  part. ◀

**Proof of Corollary 1.4.** The proof proceeds in a manner similar to the proof of Corollary 1.3. This time we inductively zip together (cf. Theorem 2.4)  $i \leq c/13$  copies of the graph  $H_{13}^{\lfloor d/2 \rfloor}$  and  $c - 13i$  copies of  $K_{3,3}$ , which results in a  $c$ -crossing-critical graph with  $i$  vertices (one per each copy) of degree greater than  $d$ . Note that we never “run out” of degree-3 vertices in the construction since each copy of  $H_{13}^{\lfloor d/2 \rfloor}$  has two such vertices. ◀

## 6 Concluding remarks and open problems

While our contribution closes the questions related to the validity of the bounded maximum degree conjecture, the following problems remain open:

► **Problem 6.1.** For each  $c \leq 12$ , determine the least integer  $D(c)$  bounding maximum degree of  $c$ -crossing-critical graphs.

► **Problem 6.2.** Develop a theory of wedges that parallels the theory of tiles for constructively establishing  $c$ -crossing-criticality of graphs with large maximum degrees.

Furthermore, the requirement in Corollary 1.4 that the number  $i$  of large degree vertices is at most  $c/13$  could possibly be weakened. The following is a natural question. For an integer  $i \geq 0$ , let  $c_i$  denote the largest possible positive integer  $c$  for which there exists an integer  $D$  such that every 3-connected  $c$ -crossing-critical graph has at most  $i$  vertices of degree larger than  $D$ . From the previous, we easily see that  $12 \leq c_i \leq 13i + 12$ , and we have exactly determined the value  $c_0 = 12$  in Theorems 1.1 and 1.2.

► **Problem 6.3.** Determine  $c_i$  as a function of  $i$ .

---

### References

- 1 Miklós Ajtai, Vašek Chvátal, Monroe M. Newborn, and Endre Szemerédi. Crossing-Free Subgraphs. In Peter L. Hammer, Alexander Rosa, Gert Sabidussi, and Jean Turgeon, editors, *Theory and Practice of Combinatorics*, volume 60 of *North-Holland Mathematics Studies*, pages 9–12. North-Holland, 1982. doi:10.1016/S0304-0208(08)73484-4.
- 2 Drago Bokal. Infinite families of crossing-critical graphs with prescribed average degree and crossing number. *J. Graph Theory*, 65(2):139–162, 2010. doi:10.1002/jgt.20470.
- 3 Drago Bokal, Mojca Bračič, Marek Dernár, and Petr Hliněný. On Degree Properties of Crossing-Critical Families of Graphs. *Electron. J. Comb.*, 26(1):#P1.40, 2019.
- 4 Drago Bokal, Markus Chimani, and Jesús Leaños. Crossing number additivity over edge cuts. *Eur. J. Comb.*, 34(6):1010–1018, 2013. doi:10.1016/j.ejc.2013.02.002.
- 5 Drago Bokal, Bogdan Oporowski, R. Bruce Richter, and Gelasio Salazar. Characterizing 2-crossing-critical graphs. *Advances in Appl. Math.*, 74:23–208, 2016. doi:10.1016/j.aam.2015.10.003.
- 6 Markus Chimani and Tilo Wiedera. An ILP-based Proof System for the Crossing Number Problem. In *ESA 2016*, LIPIcs 57, pages 29:1–29:13, 2016. doi:10.4230/LIPIcs.ESA.2016.29.
- 7 Zdeněk Dvořák, Petr Hliněný, and Bojan Mohar. Structure and Generation of Crossing-Critical Graphs. In *SoCG 2018*, LIPIcs 99, pages 33:1–33:14, 2018. doi:10.4230/LIPIcs.SoCG.2018.33.
- 8 Zdeněk Dvořák and Bojan Mohar. Crossing-critical graphs with large maximum degree. *J. Comb. Theory, Series B*, 100(4):413–417, 2010. doi:10.1016/j.jctb.2009.11.003.
- 9 César Hernández-Vélez, Gelasio Salazar, and Robin Thomas. Nested cycles in large triangulations and crossing-critical graphs. *J. Comb. Theory, Series B*, 102(1):86–92, 2012. doi:10.1016/j.jctb.2011.04.006.
- 10 Petr Hliněný. Crossing-number critical graphs have bounded path-width. *J. Comb. Theory, Series B*, 88(2):347–367, 2003. doi:10.1016/S0095-8956(03)00037-6.
- 11 Martin Kochol. Construction of crossing-critical graphs. *Discrete Math.*, 66(3):311–313, 1987. doi:10.1016/0012-365X(87)90108-7.
- 12 Jesús Leaños and Gelasio Salazar. On the additivity of crossing numbers of graphs. *Journal of Knot Theory and Its Ramifications - JKTR*, 17:1043–1050, 2008. doi:10.1142/S0218216508006531.
- 13 Frank T. Leighton. *Complexity Issues in VLSI: Optimal Layouts for the Shuffle-exchange Graph and Other Networks*. MIT Press, Cambridge, MA, USA, 1983.
- 14 Bojan Mohar, Richard J. Nowakowski, and Douglas B. West. Research problems from the 5th Slovenian Conference (Bled, 2003). *Discrete Math.*, 307(3-5):650–658, 2007. doi:10.1016/j.disc.2006.07.013.

- 15 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, 2001. URL: <https://jhupbooks.press.jhu.edu/content/graphs-surfaces>.
- 16 Benny Pinontoan and R. Bruce Richter. Crossing numbers of sequences of graphs II: Planar tiles. *J. Graph Theory*, 42(4):332–341, 2003. doi:10.1002/jgt.10097.
- 17 R. Bruce Richter and Carsten Thomassen. Minimal Graphs with Crossing Number at Least  $k$ . *J. Comb. Theory, Series B*, 58(2):217–224, 1993. doi:10.1006/jctb.1993.1038.
- 18 Jozef Širáň. Infinite families of crossing-critical graphs with a given crossing number. *Discrete Math.*, 48(1):129–132, 1984. doi:10.1016/0012-365X(84)90140-7.



# Preconditioning for the Geometric Transportation Problem

**Andrey Boris Khesin**

University of Toronto, Toronto, Ontario, Canada

**Aleksandar Nikolov**

University of Toronto, Toronto, Ontario, Canada

<http://www.cs.toronto.edu/~anikolov/>

anikolov@cs.toronto.edu

**Dmitry Paramonov**

University of Toronto, Toronto, Ontario, Canada

---

## Abstract

In the geometric transportation problem, we are given a collection of points  $P$  in  $d$ -dimensional Euclidean space, and each point is given a supply of  $\mu(p)$  units of mass, where  $\mu(p)$  could be a positive or a negative integer, and the total sum of the supplies is 0. The goal is to find a flow (called a transportation map) that transports  $\mu(p)$  units from any point  $p$  with  $\mu(p) > 0$ , and transports  $-\mu(p)$  units into any point  $p$  with  $\mu(p) < 0$ . Moreover, the flow should minimize the total distance traveled by the transported mass. The optimal value is known as the transportation cost, or the Earth Mover's Distance (from the points with positive supply to those with negative supply). This problem has been widely studied in many fields of computer science: from theoretical work in computational geometry, to applications in computer vision, graphics, and machine learning.

In this work we study approximation algorithms for the geometric transportation problem. We give an algorithm which, for any fixed dimension  $d$ , finds a  $(1 + \varepsilon)$ -approximate transportation map in time nearly-linear in  $n$ , and polynomial in  $\varepsilon^{-1}$  and in the logarithm of the total supply. This is the first approximation scheme for the problem whose running time depends on  $n$  as  $n \cdot \text{polylog}(n)$ . Our techniques combine the generalized preconditioning framework of Sherman, which is grounded in continuous optimization, with simple geometric arguments to first reduce the problem to a minimum cost flow problem on a sparse graph, and then to design a good preconditioner for this latter problem.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Network flows

**Keywords and phrases** Earth Mover Distance, Transportation Problem, Minimum Cost Flow

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.15

**Related Version** A full version is available at <https://arxiv.org/abs/1902.08384>.

**Funding** *Aleksandar Nikolov*: Supported by an NSERC Discovery Grant (RGPIN-2016-06333).

## 1 Introduction

In the *Geometric Transportation problem*, we are given a set  $P$  of  $n$  points in  $d$ -dimensional Euclidean space, and a function  $\mu : P \rightarrow \mathbb{Z}$  that assigns a (positive or negative integer) weight to each point, so that  $\sum_{p \in P} \mu(p) = 0$ . We call  $\mu$  the supply function. We can think of each point  $p \in P$  as either a pile of earth, or a hole, and  $\mu(p)$  gives, respectively, the amount of earth (if positive) or the size of the hole (if negative). The constraint on  $\mu$  means that the total space in the holes equals the total amount of earth in the piles. Our goal is to find the most efficient way to transport the earth to the holes, where the cost of transporting a unit of mass from  $p$  to  $q$  equals the distance  $\|p - q\|_2$  it must travel, measured in the Euclidean norm.



© Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov;  
licensed under Creative Commons License CC-BY

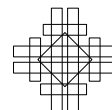
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 15; pp. 15:1–15:14

Leibniz International Proceedings in Informatics



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



## 15:2 Preconditioning for Transportation

More formally, let  $P_+ = \{p \in P : \mu(p) \geq 0\}$  be the “piles” and  $P_- = \{q \in P : \mu(q) < 0\}$  be the “holes”. We want to solve the minimum cost flow problem given by the following linear program

$$\text{Minimize } \sum_{p \in P_+, q \in P_-} f_{pq} \|p - q\|_2$$

subject to

$$\forall p \in P_+ : \sum_{q \in P_-} f_{pq} = \mu(p),$$

$$\forall q \in P_- : \sum_{p \in P_+} f_{pq} = \mu(q),$$

$$\forall p \in P_+, q \in P_- : f_{pq} \geq 0.$$

Above, the constraints enforce that all demands are satisfied, i.e. all the earth goes into holes, and no hole is overfilled. A vector  $f \in \mathbb{R}^{P_+ \times P_-}$  that satisfies these constraints is called a *transportation map* for  $(P, \mu)$ . The cost of an optimal transportation map is called the *transportation cost* of  $(P, \mu)$ , or also the Earth Mover’s Distance, or 1-Wasserstein distance between the measures  $\phi = \mu|_{P_+}$  and  $\psi = -\mu|_{P_-}$ , and is denoted  $\text{COST}(P, \mu)$ .

The geometric transportation problem is a discretized version of the continuous optimal mass transportation problem of Monge and Kantorovich. This is a classical problem in mathematics, with many beautiful connections to partial differential equations, geometry, and probability theory, among others [19]. The discrete problem above has also found a large number of applications, for example to shape matching [9, 10] and image retrieval [15, 13] in computer vision, and interpolation between distributions in graphics [6]. It is also of intrinsic interest as a natural problem in computational geometry.

The geometric transportation problem can be modeled as a minimum cost flow problem in the complete bipartite graph with bi-partition  $P_+ \cup P_-$ . This graph is dense, and solving the minimum cost flow problem exactly using the algorithm of Lee and Sidford would take time  $O(n^{2.5} \text{polylog}(U))$ ,<sup>1</sup> where  $U = \sum_{p \in P_+} \mu(p)$ . If instead we settle for an  $(1+\varepsilon)$ -approximation, then the recent algorithm of Sherman [18] gives running time  $O(n^{2+o(1)}\varepsilon^{-2})$ . However, there exist faster (in certain parameter regimes) approximation algorithms which exploit the geometric structure of the problem: after a long line of work, Sharathkumar and Agarwal [17] gave an algorithm that computes a  $(1+\varepsilon)$ -approximation in time  $O(n\sqrt{U} \text{polylog}(U, \varepsilon, n))$ , and, recently, Agarwal et al. [1] gave several algorithms with different trade-offs, among them an  $(1+\varepsilon)$ -approximation algorithm running in time  $O(n^{3/2}\varepsilon^{-d} \text{polylog}(U, n))$ . A nearly linear time  $(1+\varepsilon)$ -approximation algorithm was given by Sharathkumar and Agarwal [16] in the special case of unit supplies; their algorithm runs in time  $O(n\varepsilon^{-O(d)} \text{polylog}(n))$ . Until now, no such algorithm was known for general supply functions.

A related line of work focuses on estimating the transportation cost, without necessarily computing a transportation map. An influential paper in this direction was that of Indyk [11], whose algorithm gives a constant factor approximation to the transportation cost in the case of unit supplies, in time  $O(n \text{polylog}(n))$ . This result was extended to arbitrary supplies and to approximation factor  $1+\varepsilon$  by Andoni et al. [4], whose algorithm runs in time  $O(n^{1+o(1)})$ , with the  $o(1)$  factor in the exponent hiding dependence on  $\varepsilon$ . The result of Sherman [18], mentioned above, together with the existence of sparse Euclidean spanners, implies an

<sup>1</sup> We allow the implicit constants in the asymptotic notation to depend on the dimension  $d$ , except when the asymptotic notation is used in an exponent.



$O(n^{1+o(1)}\varepsilon^{-O(d)})$  time algorithm to estimate the transportation cost.<sup>2</sup> It is not immediately clear, however, how to use the flow in  $G$  to construct a transportation map of comparable cost in nearly linear time.

There is a related line of work [14, 2, 3] that studies the transportation problem when the cost of transporting mass from  $p$  to  $q$  is  $\|p - q\|_2^r$ , giving the  $r$ -Wasserstein distance. This appears to be a more challenging problem, and we do not address it further.

## 1.1 Our results and methods

Let us recall that the aspect ratio (or spread) of a pointset  $P$  is defined as the ratio of its diameter to the smallest distance between two distinct points. Our main result is a nearly linear time  $(1 + \varepsilon)$ -approximation algorithm for the geometric transportation problem, as captured by the following theorem.

► **Theorem 1.** *There exists a randomized algorithm that on input an  $n$ -point set  $P \subset \mathbb{Q}^d$  with aspect ratio  $\Delta$ , and supply function  $\mu : P \rightarrow \mathbb{Z}$ , runs in time  $O(n\varepsilon^{-O(d)} \log(\Delta)^{O(d)} \log n)$ , and with probability at least  $1/2$  finds a transportation map with cost at most  $(1 + \varepsilon) \cdot \text{COST}(P, \mu)$ .*

*There also exists a randomized algorithm that on input an  $n$ -point set  $P \subset \mathbb{Q}^d$  and supply function  $\mu : P \rightarrow \mathbb{Z}$  such that  $U = \sum_{p \in P} \mu(p)$ , runs in time  $O(n\varepsilon^{-O(d)} \log(U)^{O(d)} \log(n)^2)$ , and with probability at least  $1/2$  finds a transportation map with cost at most  $(1 + \varepsilon) \cdot \text{COST}(P, \mu)$ .*

In constant dimension, the dependence of the running time on the aspect ratio  $\Delta$  or total supply  $U$  is polylogarithmic, and the dependence on the approximation  $\varepsilon$  is polynomial. The dependence on  $n$  is just  $O(n \log n)$  (respectively  $O(n \log(n)^2)$ ). This is in contrast with prior work which either had a much larger dependence on  $n$ , or a polynomial dependence on  $U$ .

In the proof of this result, we employ a combination of geometric tools, and tools from continuous optimization developed for the general minimum cost flow problem. As a first step, we reduce the transportation problem to an (uncapacitated) minimum cost flow problem on a random sparse graph. This construction is closely related to the prior work of Sharathkumar and Agarwal in the case of unit capacities [16], and also to the estimation algorithm for the transportation cost in [4]. In particular, the sparse graph and minimum cost flow instance we construct are a simplification of the minimum cost flow instance in [4]. Together with the recent work of Sherman [18], this reduction is enough to get a  $O(n^{1+o(1)}\varepsilon^{-O(d)})$  time algorithm to estimate the transportation cost. As mentioned above, this running time can also be achieved using a Euclidean spanner, and the random sparse graph we use can, in fact, be seen as a randomized spanner. Our graph, however, has a nice hierarchical structure not shared by other spanner constructions. In particular, there is a quadtree of the input point set such that any edge leaving a cell of the quadtree goes to either a sibling cell or a parent cell. This property is useful both for improving the running time further, and for computing a transportation map.

To further improve the running time, we open up Sherman's elegant framework, and combine it with classical geometric constructions. Sherman showed that repeatedly finding a flow which is approximately optimal, and approximately feasible gives a fast algorithm, as long as the problem being solved is sufficiently well conditioned. Unfortunately, most minimum cost flow problems are not well-conditioned in their natural formulations, and we need to construct a *preconditioner*. We exploit ideas from the known embeddings of

<sup>2</sup> We are indebted to an anonymous reviewer for this observation.

transportation cost into  $\ell_1$  [7, 12] to construct a preconditioner with condition number that depends polynomially on the approximation factor and the logarithm of the aspect ratio. The insight that these simple and well-known techniques can be repurposed to give high-quality preconditioners is one of our main conceptual contributions. Finally, we also give a simple method to extract a transportation map from a low-cost flow in our random sparse graph.

## 2    **Notation and basic notions**

We use the notation  $\|x\|_p$  for the  $\ell_p$  norm of a vector  $x \in \mathbb{R}^d$ :  $\|x\|_p = \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$ . We will assume that the input  $P \subset \mathbb{Q}^d$  to our problem lies in  $[0, \Delta]^d$ , and that the smallest Euclidean distance between any two points is at least 1. Since any shifts, rotations, or dilations of the points do not change the problem, this assumption is equivalent to assuming that the aspect ratio of  $P$ , i.e. the ratio between the diameter of  $P$  and the smallest distance between two distinct points, is bounded between  $\Delta$  and  $\sqrt{d}\Delta$ .

In fact, at the cost of a small increase in the approximation factor, we can reduce to this case of bounded aspect ratio. The reduction produces a point set  $P$  for which  $\Delta$  is bounded by a factor that depends on  $U = \sum_{p \in P_+} \mu(p)$ .

► **Lemma 2** ([11, 4]). *Suppose that there exists a function  $T$ , increasing in all its parameters, and an algorithm which, for any  $\varepsilon, \delta < 1$ , on input of size  $n$  in  $[0, \Delta]^d \cap \mathbb{Z}^d$ , runs in time  $O(nT(\Delta, \varepsilon, \delta))$ , and with probability  $1 - \delta$  computes a  $1 + \varepsilon$  approximation to the geometric transportation problem. Then there exists an algorithm that takes any input  $P \subset \mathbb{R}^d$  of size  $n$  and  $\mu : P \rightarrow \mathbb{Z}$  such that  $U = \sum_{p \in P_+} \mu(p)$ , runs in time  $O(nT(c\Delta U/\varepsilon^2, \varepsilon, \delta n))$  for an absolute constant  $c$ , and, with probability  $1 - \delta$ , achieves an approximation of  $1 + O(\varepsilon)$  for the geometric transportation problem on  $P$  and  $\mu$ .*

With this lemma, the second algorithm in Theorem 1 follows from the first one. For this reason, we can focus our presentation on the case of bounded aspect ratio.

For a set  $V$ , we call  $f \in \mathbb{R}^{V \times V}$  a *flow* if it is anti-symmetric, i.e.  $f_{uv} = -f_{vu}$  for every  $u, v \in V$ . Intuitively, we think of  $f_{uv} > 0$  as flow going in the direction from  $u$  to  $v$ . For a graph  $G = (V, E)$ , we define a flow on  $G$  to be a flow  $f$  supported on  $E$ , i.e. one such that  $f_{uv} = 0$  for any  $(u, v) \notin E$ . The *divergence* of a flow  $f$  at  $u$  is the quantity  $\sum_{v \in V} f_{uv}$ , i.e. the excess of the flow leaving  $u$  over the flow entering  $u$ . When the divergence at  $u$  is 0, we say that *flow conservation* is satisfied at  $u$ .

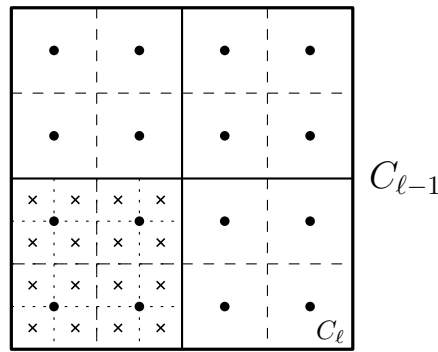
In general, we assume that our graphs are stored in the adjacency list representation, and that flow values are stored with the adjacency list.

## 3    **Reduction to minimum cost flow on a sparse graph**

The first step in our algorithm is to reduce the geometric transportation problem, which is naturally modeled as a minimum cost flow problem on a complete bipartite graph, to another minimum cost flow on a sparse random graph. The following construction is a simplified version of one used in [4].

### 3.1    **Graph construction**

Recall that  $P \subset [0, \Delta]^d$ . We start by constructing a grid containing all the points in  $P$  as follows. We sample a uniformly random point  $x \in [0, \Delta]^d$ , and define the cell  $C_0 = [-\Delta, \Delta]^d + x$ . Note that all points in  $P$  are in  $C_0$ . We say that  $C_0$  is on *level 0*. The set of



■ **Figure 1** A cell  $C_{\ell-1}$  on level  $\ell - 1$  and the four cells on level  $\ell$  contained in it. The subcells of  $C_{\ell-1}$  are shown with dashed lines, and the corresponding net points in  $N_{\ell-1}$  with black dots. The subcells of one of the cells on level  $\ell$  are shown with dotted lines, and the corresponding net point in  $N_\ell$  with crosses.

cells on level  $\ell$  is labelled  $\mathcal{C}_\ell$  and is constructed by taking each cell  $C$  in  $\mathcal{C}_{\ell-1}$  and dividing it into  $2^d$  equally-sized cubes of side length half the side length of  $C$ , and retaining those that contain points in  $P$ . We say that  $C$  is the *parent* of the cells into which it was divided. Conversely, those  $2^d$  cells are called the *children* of  $C$ . This division is continued until level  $L$ , where all points in  $P$  lie in different cells in  $\mathcal{C}_L$ . With probability 1, no point of  $P$  lands on the boundary of any cell, so we will ignore this event.

Since the side length of any cell on level  $\ell$  is half of the side length of a cell on level  $\ell - 1$ , any cell in  $\mathcal{C}_\ell$  has side length  $2^{1-\ell}\Delta$ , and we refer to this length as  $\Delta_\ell$ . Also note that, since the closest two points in  $P$  are at least distance 1 apart, we have  $L \leq \log_2(2\sqrt{d}\Delta)$ , so  $L = O(\log \Delta)$ . Moreover, any point  $p \in P$  lies in at most  $L + 1$  cells, one per level, and, since in  $\mathcal{C}_\ell$  we only retain cells that contain points in  $P$ , we have that  $|\mathcal{C}_0 \cup \dots \cup \mathcal{C}_L| \leq n(L + 1) = O(n \log \Delta)$ .

Let  $\varepsilon_0$  be a small number such that  $\varepsilon_0^{-1}$  is an *even* integer. This  $\varepsilon_0$  is not the same as the value used when we speak of a  $(1 + \varepsilon)$ -approximation, although the two are related, as we will see. Next, for each  $\ell \in \{0, \dots, L\}$ , we take each cell  $C \in \mathcal{C}_\ell$ , and we divide it into  $\varepsilon_0^{-d}$  *subcells*, each of side length  $\varepsilon_0\Delta_\ell$ . The set of all such subcells of all  $C \in \mathcal{C}_\ell$  is denoted  $\tilde{\mathcal{C}}_\ell$ . To each subcell we associate a *net point* at its centre and we denote the set of all net points on level  $\ell$  as  $N_\ell$ . See Figure 1 for an illustration. Note that  $|N_\ell| = |\mathcal{C}_\ell|\varepsilon_0^{-d}$ , so  $|N_0 \cup \dots \cup N_L| \leq n(L + 1)\varepsilon_0^{-d} = O(n\varepsilon_0^{-d} \log \Delta)$ .

If  $p$  is a point in  $C_0$ , let  $C_\ell(p) \in \tilde{\mathcal{C}}_\ell$  be the unique subcell on level  $\ell$  that contains  $p$ . Similarly, let  $N_\ell(p)$  be the net point on level  $\ell$  closest to  $p$ . Equivalently,  $N_\ell(p)$  is the center of  $C_\ell(p)$ . We say that  $N_\ell(p)$  and  $N_{\ell-1}(p)$  are each other's *child* and *parent*, respectively.

We are now ready to construct our graph  $G = (V, E)$ . We set  $V = P \cup N_0 \cup N_1 \cup \dots \cup N_L$ , the set of all of the points in the original problem and all of the netpoints we have constructed. By the previous discussion,  $|V| = O(n\varepsilon_0^{-d} \log \Delta)$ . We build our graph using three types of edges. The first set,  $E_1$ , connects points  $p$  in  $P$  to their closest net point on level  $L$ , i.e.  $E_1 = \{(p, N_L(p)) : p \in P\}$ . The size of this set is  $|E_1| = n$ . The second set of edges,  $E_2$ , connects all net points of subcells of a given cell pairwise. Thus,

$$E_2 = \{(u, v) : \ell \in \{0, 1, \dots, L\}, C \in \mathcal{C}_\ell, u, v \in N_\ell \cap C \text{ s.t. } u \neq v\}.$$

Since  $|\mathcal{C}_\ell| \leq n$ , and, for any  $C \in \mathcal{C}_\ell$ ,  $|N_\ell \cap C| = \varepsilon_0^{-d}$ , we have that  $|E_2| \leq (L + 1)n\varepsilon_0^{-2d} = O(n\varepsilon_0^{-2d} \log \Delta)$ . The last set of edges,  $E_3$ , connects net points to their parent net points, i.e.

$$E_3 = \{(N_{\ell-1}(u), u) : \ell \in \{1, 2, \dots, L\}, u \in N_\ell\}.$$

## 15:6 Preconditioning for Transportation

The size of  $E_3$  is less than the total number of net points, so  $|E_3| = O(n\varepsilon_0^{-d} \log \Delta)$ . Then, the set of edges is defined as  $E = E_1 \cup E_2 \cup E_3$ , and this completes the description of  $G = (V, E)$ . The number of edges is dominated by the size of  $E_2$ , and is in  $O(n\varepsilon_0^{-2d} \log \Delta)$ . We will be calculating distances between points along paths in this graph, and we define the distance/cost of any edge to equal the Euclidean distance between the two endpoints of the edge.

For the rest of the paper we will use  $G$  to refer to the random graph constructed above. Having stored the random shift of  $C_0$ , we can enumerate the vertices in time  $O(n\varepsilon_0^{-d} \log \Delta)$  by going through each point in  $P$  and checking which cells it lies in. We can then store the vertices in a static dictionary [8]. Similarly, we can construct the adjacency lists of the vertices in time  $O(n\varepsilon_0^{-2d} \log \Delta)$ .

### 3.2 Preserving Euclidean distance

Let  $\text{dist}_G(u, v)$  be the shortest path distance between two nodes  $u$  and  $v$  of  $G$ , i.e.

$$\text{dist}_G(u, v) = \min \sum_{i=1}^k \|u_i - u_{i-1}\|_2,$$

with the minimum taken over paths  $u = u_0, \dots, u_k = v$  connecting  $u$  and  $v$  in  $G$ . Our goal for this section is to show that, for any two  $p$  and  $q$  in  $P$ , the expected value of  $\text{dist}_G(u, v)$  is close to the Euclidean distance between them. In other words, we want to show that our random graph construction provides a randomized embedding of Euclidean distance into the shortest path metric of a sparse graph. This is similar to the embeddings used in [4] and [17, 16], and can be seen as a randomized spanner construction, in which the graph  $G$  only needs to have low stretch in expectation.

For two points  $p$  and  $q$  in  $P$ , we define  $\ell(p, q)$  to be the level of the smallest cell containing both  $p$  and  $q$ , which can depend on the random shift of the grid. This means that  $C_{\ell(p, q)}(p) = C_{\ell(p, q)}(q)$ , but  $C_{\ell(p, q)+1}(p) \neq C_{\ell(p, q)+1}(q)$ . The following definition will be useful.

► **Definition 3.** *Given two points  $p, q \in P$ , the canonical path between  $p$  and  $q$  in the random graph  $G$  consists of the edges*

$$\begin{aligned} & \{(N_\ell(p), N_{\ell-1}(p)) : \ell(p, q) + 1 \leq \ell \leq L\} \cup \{(N_{\ell(p, q)}(p), N_{\ell(p, q)}(q))\} \\ & \cup \{(N_\ell(q), N_{\ell-1}(q)) : \ell(p, q) + 1 \leq \ell \leq L\} \cup \{(N_L(p), p), (N_L(q), q)\}. \end{aligned}$$

The canonical path could be much longer than  $\|p - q\|_2$  because, if  $p$  and  $q$  are two nearby points that are separated by a grid line, the canonical path will be much larger than the Euclidean distance. However, when  $p$  and  $q$  are very close together, the likelihood that such a grid line will fall between them is low. To formalize this intuition, we need the following well-known bound on the probability that  $p$  and  $q$  will be split at a certain level of the tree, which is elementary, and goes back at least to Arora's work on Euclidean TSP [5].

► **Lemma 4.** *Over the random shift of  $C_0$ ,  $\forall p, q \in P$ ,*

$$\mathbb{P}(C_\ell(p) \neq C_\ell(q)) \leq \frac{\|p - q\|_1}{\Delta_\ell} \leq \frac{\sqrt{d}\|p - q\|_2}{\Delta_\ell}.$$

Using this standard lemma, we can prove the bound on the distortion between Euclidean distance and the expected shortest path distance in  $G$ . The proof, which is standard, is presented in the full version of the paper.

► **Lemma 5.** *Let  $p, q \in P$ . Then, for any shift of  $C_0$ ,  $\|p - q\|_2 \leq \text{dist}_G(p, q)$ . Moreover, over the random shift of  $C_0$ ,  $\mathbb{E}[\text{dist}_G(p, q)] \leq (1 + O(\varepsilon_0 L))\|p - q\|_2$ .*

### 3.3 Approximating the transportation cost with minimum cost flow

We are now ready to translate the transportation problem to a minimum cost flow problem in  $G$ , captured by the following linear program.

$$\text{Minimize } \sum_{(u,v) \in E} f_{uv} \|u - v\|_2 \tag{1}$$

subject to

$$\forall u, v \in V : f_{uv} = -f_{vu} \tag{2}$$

$$\forall p \in P : \sum_{u \in V} f_{pu} = \mu(p), \tag{3}$$

$$\forall u \in V \setminus P : \sum_{v \in V} f_{uv} = 0. \tag{4}$$

Let us denote by  $\text{COST}(G, \mu)$  the value of the optimal solution to the problem (1)–(4). Note that, since  $G$  is a random graph,  $\text{COST}(G, \mu)$  is also a random variable.

The next theorem shows that that  $\text{COST}(G, \mu)$  approximates the transportation cost  $\text{COST}(P, \mu)$ . The proof is given in the full version of the paper.

► **Theorem 6.** *For any shift of  $C_0$ ,  $\text{COST}(P, \mu) \leq \text{COST}(G, \mu)$ . Moreover, over the random shift of  $C_0$ ,*

$$\mathbb{E}[\text{COST}(G, \mu)] \leq (1 + O(\varepsilon_0 L))\text{COST}(P, \mu).$$

## 4 Solving the minimum cost flow problem

In this section we describe the generalized preconditioning framework of Sherman [18], and describe how to use it to solve the minimum cost flow problem (1)–(4) in nearly linear time.

### 4.1 Generalized preconditioning framework

We take a short detour to describe the generalized preconditioning framework of Sherman. Suppose that  $\|\cdot\|$  is a norm on  $\mathbb{R}^m$ , and we are given inputs  $A \in \mathbb{R}^{n \times m}$  and  $b \in \mathbb{R}^n$ . The framework is concerned with solving the following optimization problem over the variables  $f \in \mathbb{R}^m$ :

$$\text{Minimize } \|f\| \tag{5}$$

$$\text{subject to } Af = b, \tag{6}$$

This formulation can capture many important problems, including, crucially for us, the uncapacitated minimum cost flow problem,<sup>3</sup> as we explain later.

A central definition in the framework is that of the nonlinear condition number. Before we state it, let us recall that the norm of a linear map  $T : X \rightarrow Y$ , where  $(X, \|\cdot\|_X)$  and  $(Y, \|\cdot\|_Y)$  are two finite dimensional normed vector spaces, is defined as

$$\|T\| = \sup_{x \neq 0} \frac{\|T(x)\|_Y}{\|x\|_X}.$$

<sup>3</sup> Uncapacitated minimum cost flow is commonly known as the transportation problem. We do not use this terminology, in order to avoid confusion with the geometric transportation problem.

## 15:8 Preconditioning for Transportation

We adopt the same definition for non-linear maps, as well. For an  $n \times m$  matrix  $A$ , and norms  $\|\cdot\|_X$  on  $X = \mathbb{R}^n$  and  $\|\cdot\|_Y$  on the column span  $Y$  of  $A$ , we use the notation  $\|A\|_{\|\cdot\|_X \rightarrow \|\cdot\|_Y}$  to refer to the norm of the linear map from  $X$  to  $Y$  represented by  $A$ .

Having recalled these notions, we proceed to define the condition number.

► **Definition 7.** Let  $(X, \|\cdot\|_X)$  and  $(Y, \|\cdot\|_Y)$  be two finite dimensional normed vector spaces. The non-linear condition number  $\kappa$  of a linear map  $T : X \rightarrow Y$  is defined by

$$\kappa(T) = \inf_S \|T\| \|S\|,$$

where  $S : Y \rightarrow X$  ranges over all (not necessarily linear) maps such that for all  $x \in X$  we have  $T(S(Tx)) = Tx$ .

For an  $n \times m$  matrix  $A$ , and norms  $\|\cdot\|_X$  on  $X = \mathbb{R}^m$  and  $\|\cdot\|_Y$  on the column span  $Y$  of  $A$ , we define  $\kappa_{\|\cdot\|_X \rightarrow \|\cdot\|_Y}(A)$  as  $\kappa(T)$ , where  $T : X \rightarrow Y$  is the linear map represented by the matrix  $A$ .

This definition generalizes the standard condition number, which is the special case in which both norms are taken to be Euclidean.

The generalized preconditioning framework is based on composing rough approximation algorithms to get a high-quality approximation. The rough approximation algorithms are further allowed to violate the constraints of the problem slightly. Thus, they achieve a bi-criteria approximation, captured by the following definition.

► **Definition 8.** Let  $\|\cdot\|_Y$  be a norm defined on the range of the matrix  $A$  in (6). Let  $f^*$  be an optimal solution to the problem (5)–(6). Then  $f \in \mathbb{R}^m$  is called an  $(\alpha, \beta)$ -solution (with respect to  $\|\cdot\|_Y$ ) to (5)–(6) if  $\|f\| \leq \alpha \|f^*\|$  and, moreover,

$$\|Af - b\|_Y \leq \beta \|A\|_{\|\cdot\| \rightarrow \|\cdot\|_Y} \|f^*\|.$$

An algorithm that, when given inputs  $A$  and  $b$ , outputs an  $(\alpha, \beta)$ -solution  $f$  is called an  $(\alpha, \beta)$ -solver (with respect to  $\|\cdot\|_Y$ ) for (5)–(6).

We use the next result of Sherman, which gives a solver for (5)–(6) with running time controlled by condition number of the constraint matrix.

► **Theorem 9 ([18]).** Let  $\varepsilon, \beta > 0$ , and suppose that the norm in (5) is the  $\ell_1^m$  norm. Let  $\kappa = \kappa_{\|\cdot\|_1 \rightarrow \|\cdot\|_1}(A)$ , and let  $M$  be an upper bound on the time necessary to compute matrix-vector products with  $A$  and  $A^\top$ . Then there exists a  $(1 + \varepsilon, \beta)$ -solver with respect to  $\|\cdot\|_1$  for the problem (5)–(6) with running time bounded by

$$O(\kappa^2(m + n + M) \log(m)(\varepsilon^{-2} + \log(1/\beta))).$$

This algorithm is based on repeatedly applying an  $(\alpha, \beta)$  solver with much worse dependence on  $\beta$ ; in fact, at the cost of a slightly worse dependence on  $\kappa$ , for the latter one can use a simple solver based on the multiplicative weights update method.

Rescaling  $f$  coordinatewise gives us the following easy corollary, whose proof is omitted.

► **Corollary 10.** Let  $\varepsilon, \beta > 0$ , and let  $c \in \mathbb{R}_{>0}^m$ . Suppose that the norm in (5) is given by  $\|f\|_c = \sum_{i=1}^m c_i f_i$ . Let  $\kappa = \kappa_{\|\cdot\|_c \rightarrow \|\cdot\|_1}(A)$ , and let  $M$  be the time necessary to compute matrix-vector products with  $A$  and  $A^\top$ . Then there exists a  $(1 + \varepsilon, \beta)$ -solver with respect to  $\|\cdot\|_1$  for the problem (5)–(6) with running time bounded by  $O(\kappa^2(m + n + M) \log(m)(\varepsilon^{-2} + \log(1/\beta)))$ .

While Theorem 9 and Corollary 10 allow us to find solutions which are very close to being feasible, they do not give an exactly feasible solution. The final result we use from Sherman allows us to use a solver which exactly satisfies the constraints, but only achieves a large approximation ratio, to round an approximately feasible solution to an exactly feasible one.

To state the result, we need to define the composition of two solvers  $\mathcal{F}$  and  $\mathcal{F}'$ . The composed algorithm  $\mathcal{F}' \circ \mathcal{F}$  first calls  $\mathcal{F}$  on  $A$  and  $b$  to get  $f$ ; then it calls  $\mathcal{F}'$  on  $A$  and the residual vector  $b - Af$  to get a solution  $f'$ ; finally it outputs  $f + f'$ .

► **Theorem 11** ([18]). *Let  $C \geq 1$ , and  $\varepsilon, \beta > 0$ . Let  $\kappa = \kappa_{\|\cdot\| \rightarrow \|\cdot\|_Y}(A)$ , where  $\|\cdot\|$  is the norm in (5), and  $\|\cdot\|_Y$  is some norm on the column span of  $A$ . Then, if  $\mathcal{F}$  is a  $(1 + \varepsilon, \varepsilon\beta/\kappa)$ -solver, and  $\mathcal{F}'$  is a  $(C, 0)$ -solver, the composition  $\mathcal{F}' \circ \mathcal{F}$  is a  $(1 + \varepsilon + C\varepsilon\beta, 0)$ -solver.*

## 4.2 Preconditioner construction

The formulation (5)–(6) captures the (uncapacitated) minimum cost flow problem with arbitrary demands. We will explain this for the graph  $G = (V, E)$  defined in Section 3.2 and the minimum cost flow problem (1)–(4). Let us pick an arbitrary orientation on the edges  $E$ , and call the directed edges  $\vec{E}$ . Then take  $A$  to be the directed vertex by edge incidence matrix of  $G$ : it is indexed by  $V \times \vec{E}$ , and for any node  $u$ , and any directed edge  $e = (v, w)$ , set  $A_{u,e} = 1$  if  $u = v$ ,  $A_{u,e} = -1$  if  $u = w$ , and  $A_{u,e} = 0$  otherwise. We represent a flow  $f$  by its restriction to  $\vec{E}$ , seen as a vector in  $\mathbb{R}^{\vec{E}}$ . Slightly abusing notation, we will use the letter  $f$  both for this vector, and for the flow, which is defined both for  $(u, v) \in \vec{E}$ , and for  $(v, u)$ , with  $f_{vu} = -f_{uv}$ . For a flow vector  $f$ , the product  $Af$  gives us the vector of divergences, i.e. for any  $u \in V$

$$(Af)_u = \sum_{v:(u,v) \in \vec{E}} f_{uv} - \sum_{v:(v,u) \in \vec{E}} f_{vu} = \sum_{v:(u,v) \in E} f_{uv}.$$

We define the vector  $b \in \mathbb{R}^V$  to encode the supplies, i.e. for  $p \in P$  we set  $b_p = \mu(p)$ , and for  $u \in V \setminus P$  we set  $b_u = 0$ . It follows that the constraint  $Af = b$  encodes (3)–(4). Finally, let us denote the cost of an edge  $e = (u, v) \in \vec{E}$  by  $c(e) = \|u - v\|_2$ . Then, for the norm in the objective (5), we choose  $\|f\|_c = \sum_{e \in E} c(e)|f_e|$ . With these choices of  $A$ ,  $b$ , and  $\|\cdot\| = \|\cdot\|_c$ , an optimal solution to (5)–(6) gives an optimal solution to (1)–(4). For the rest of this section we will fix  $A$ ,  $b$ ,  $c$ , and  $\|\cdot\|_c$  to be as just defined.

Unfortunately, we cannot directly use Corollary 10 to get the running time we are aiming for, since the condition number of the matrix  $A$  could be large. We address this by designing a preconditioner: another matrix  $B$ , of full column rank, which can be applied quickly, and has the property that  $BA$  has small condition number. This allows us to apply Corollary 10 to the problem (5)–(6) with the modified, but equivalent, constraint  $BAf = Bb$ , and get a fast algorithm for the minimum cost flow problem in  $G$ .

In our construction of the preconditioner  $B$  we will use the following lemma, which was implicit in [18]. We omit the proof, which follows from the arguments in [18].

► **Lemma 12.** *Let  $\|\cdot\|_X$  be a norm on  $\mathbb{R}^n$ , and let  $H$  be an  $n \times m$  matrix. Suppose that there exists a  $\gamma > 0$  such that for any  $h$  in the column span  $Y$  of  $H$ , the norm  $\|\cdot\|_Y$  satisfies*

$$\|h\|_Y \leq \min\{\|f\|_X : f \in \mathbb{R}^n, Hf = h\} \leq \gamma \|h\|_Y.$$

*Then  $\kappa_{\|\cdot\|_X \rightarrow \|\cdot\|_Y}(H) \leq \gamma$ .*

We will design a preconditioner matrix  $B$  such that  $\|\tilde{b}\|_1$  approximates the cost of the minimum cost flow with supply vector  $\tilde{b}$ . Then the fact that  $BA$  has small condition number will follow from Lemma 12. The following lemma captures the construction of  $B$ , which is inspired by embedding of the Earth Mover Distance in  $\ell_1$  [7, 12].



## 15:10 Preconditioning for Transportation

► **Lemma 13.** *There exists a matrix  $B \in \mathbb{R}^{V \times V}$  of full column rank with at most  $O(|V| \log \Delta)$  nonzero entries, such that the following holds. For any  $\tilde{b} \in \mathbb{R}^V$  such that  $\sum_{u \in V} \tilde{b}_u = 0$ , we have*

$$\|B\tilde{b}\|_1 \leq \min\{\|f\|_c : f \in \mathbb{R}^{\tilde{E}}, Af = \tilde{b}\} \leq \gamma \|B\tilde{b}\|_1,$$

for  $\gamma = O(\log(\Delta)/\varepsilon_0)$ . Moreover, a flow  $f$  satisfying  $Af = \tilde{b}$  of cost at most  $\gamma \|B\tilde{b}\|_1$  can be constructed in time  $O(n\varepsilon_0^{-d} \log \Delta)$  given  $B\tilde{b}$ .

**Proof.** The matrix  $B$  has one row associated with each vertex  $u \in V$ . For  $p \in P$ , we let  $B_{p,p} = \|p - N_L(p)\|_2$  and  $B_{p,u} = 0$  for any  $u \neq p$ . Every other vertex  $u \in V \setminus P$  is a net point. Suppose that  $u$  is in  $N_\ell$ ; then for any  $v \in V$  such that  $N_\ell(v) = u$ ,  $B_{u,v} = \frac{\varepsilon_0 \Delta_\ell}{4(L+1)}$ , and for all other  $v$ ,  $B_{u,v} = 0$ . This defines the matrix  $B$ . Notice that each vertex  $u \in V$  contributes to at most  $L+2$  nonzero entries of  $B$ : one for each  $\ell \in \{0, \dots, L\}$  corresponding to  $N_\ell(u)$ , and one more when  $u \in P$ . It is easy to verify directly that  $B$  has full column rank.

The definition of  $B$  guarantees that

$$\|B\tilde{b}\|_1 = \sum_{p \in P} |\tilde{b}_p| \|p - N_L(p)\|_2 + \frac{1}{L+1} \sum_{\ell=0}^L \frac{\varepsilon_0 \Delta_\ell}{4} \sum_{\tilde{C} \in \tilde{\mathcal{C}}_\ell} \left| \sum_{u \in \tilde{C} \cap V} \tilde{b}_u \right|. \quad (7)$$

The remainder of the proof is broken down into several claims. For the first claim, let us extend the definition of  $\text{COST}(G, \cdot)$  to supplies which can be non-zero on net points as well by

$$\text{COST}(G, \tilde{b}) = \min\{\|f\|_c : f \in \mathbb{R}^{\tilde{E}}, Af = \tilde{b}\}.$$

► **Claim 14.**  $\|B\tilde{b}\|_1 \leq \text{COST}(G, \tilde{b})$ .

**Proof.** In any feasible flow  $f$ , the total cost of the flow on edges incident to points  $p \in P$  must equal the first term in (7). Then, we just need to show that the remaining terms are a lower bound on the total cost of the flow on the remaining edges. In fact we will show that, for each  $\ell$ ,

$$\frac{\varepsilon_0 \Delta_\ell}{4} \sum_{\tilde{C} \in \tilde{\mathcal{C}}_\ell} \left| \sum_{u \in \tilde{C} \cap V} \tilde{b}_u \right| \leq \text{COST}(G, \tilde{b}) - \sum_{p \in P} |\tilde{b}_p| \|p - N_L(p)\|_2. \quad (8)$$

It follows that the average of the term on the left over all  $\ell$  is also a lower bound on the right hand side, which proves the claim. To establish inequality (8) notice that, in any feasible  $f$ , flow of value at least  $\left| \sum_{u \in \tilde{C} \cap V} \tilde{b}_u \right|$  must enter or leave each  $\tilde{C} \in \tilde{\mathcal{C}}_\ell$ , and every edge  $e$  from a vertex in  $\tilde{C}$  to a vertex outside of  $\tilde{C}$  has cost at least  $c(e) \geq \varepsilon_0 \Delta_\ell / 2$  by the definition of  $G$ . Therefore, the total cost of flow leaving or entering  $\tilde{C}$  is at least  $\frac{\varepsilon_0 \Delta_\ell}{2} \left| \sum_{u \in \tilde{C} \cap V} \tilde{b}_u \right|$ . Adding up these terms over all  $\tilde{C} \in \tilde{\mathcal{C}}_\ell$  accounts for the cost of any edge of  $G$  at most twice, and (8) follows. ◁

► **Claim 15.**  $\text{COST}(G, \tilde{b}) \leq \gamma \|B\tilde{b}\|_1$  for  $\gamma = O(L/\varepsilon_0) = O(\log(\Delta)/\varepsilon_0)$ .

**Proof.** We prove the claim by constructing an appropriate feasible flow  $f$ . We proceed to construct  $f$  by levels, from the bottom up. The construction will be efficient, also proving the claim after “moreover” in the statement of the lemma. On level  $L$ , for any  $p \in P$ , we set

$f_{pu} = \tilde{b}_p$ , where  $u = N_L(p)$ . Then, for levels  $\ell = L - 1, L - 2, \dots, 1$ , we execute the following procedure in every subcell  $\tilde{C} \in \tilde{\mathcal{C}}_{\ell-1}$ . Let us define, for any node  $u$  and the current flow  $f$ , the surplus

$$\delta(u, f) = \sum_{v:(u,v) \in \tilde{E}} f_{uv} - \sum_{v:(v,u) \in \tilde{E}} f_{vu} - \tilde{b}_u = \sum_{v:(u,v) \in E} f_{uv} - \tilde{b}_u.$$

I.e. this is how much more flow leaves  $u$  than should, as prescribed by  $\tilde{b}$ . Pick any two  $u$  and  $v$  in  $N_\ell \cap \tilde{C}$  such that  $\delta(u, f) > 0 > \delta(v, f)$ , and add  $\min\{|\delta(u, f)|, |\delta(v, f)|\}$  units of flow to  $f_{vu}$ . Continue until we have that for all  $u \in N_\ell \cap \tilde{C}$  the surpluses  $\delta(u, f)$  have the same sign. Since at every step of this procedure we make the surplus of at least one node 0, and we always decrease the absolute value of the surplus at any node, we will stop after at most  $|N_\ell \cap \tilde{C}|$  steps. Finally, for the node  $u \in N_{\ell-1}$  which is the center of  $\tilde{C}$ , and for all nodes  $v \in N_\ell \cap \tilde{C}$  for which  $\delta(v, f) \neq 0$ , we set  $f_{uv} = \delta(v, f)$ . After this final step, every node  $u \in N_\ell \cap \tilde{C}$  has surplus 0.

For  $\ell = 0$ , we perform essentially the same procedure, but in the entire cell  $C_0$ . I.e. we pick any two  $u$  and  $v$  in  $N_\ell \cap C_0$  such that  $\delta(u, f) > 0 > \delta(v, f)$ , and add  $\min\{|\delta(u, f)|, |\delta(v, f)|\}$  units of flow to  $f_{vu}$ . Once again, after at most  $|N_0|$  steps all surpluses will have the same sign, and, as we show below, will in fact be 0, so  $f$  will be feasible.

An easy induction argument shows that, once we have processed levels  $L, L - 1, \dots, \ell$ , for any  $\ell > 0$ , the surplus at any node  $u \in N_L \cup \dots \cup N_\ell$  is 0, and the same is true for the surplus at any  $p \in P$ . To show that the flow  $f$  is feasible, it is enough to show that after processing the cell  $C_0$  on level 0, all nodes have surplus 0. Notice that, while processing any subcell  $\tilde{C}$ , we do not change the total surplus  $\sum_{u \in V \cap \tilde{C}} \delta(u, f)$ . This means that, for any  $\ell > 0$ , after having processed a subcell  $\tilde{C} \in \tilde{\mathcal{C}}_{\ell-1}$  with center  $u$ , we have

$$\delta(u, f) = - \sum_{v \in \tilde{C} \cap V} \tilde{b}_v = - \sum_{v: N_{\ell-1}(v)=u} \tilde{b}_v. \tag{9}$$

In particular, we have that before we start processing  $C_0$ ,  $\sum_{u \in N_0} \delta(u, f) = - \sum_{u \in V} \tilde{b}_u = 0$ . Since, once again, every step we make during the processing of  $C_0$  preserves the total surplus, and we stop when all surpluses have the same sign, then at the time we are done it must be the case that every node has surplus 0, and, therefore,  $f$  is a feasible flow.

It remains to bound the cost of the flow  $f$  constructed above. We can charge the cost of the subcells processed on each level while constructing  $f$  to one of the terms of the right hand side of (7), and this completes the proof of the claim. The details of this calculation appear in the full version of the paper.  $\triangleleft$

The two claims finish the proof of the lemma, together with the observation that the construction of the flow  $f$  can be implemented recursively in time  $O(n\varepsilon_0^{-d} \log \Delta)$ .  $\blacktriangleleft$

Our main result for solving the minimum cost flow problem in  $G$  follows. The proof, which at this point is straightforward, is given in the full version of the paper.

► **Theorem 16.** *A flow  $f$  feasible for the problem (1)–(4), with cost at most  $1 + O(\varepsilon)$  factor larger than the optimal cost, can be computed in time*

$$O \left( \frac{\log(\Delta)^2}{\varepsilon_0^2} (|E| + |V| \log(\Delta)) \log(|E|) \left( \frac{1}{\varepsilon^2} + \log \left( \frac{\log(\Delta)}{\varepsilon_0} \right) \right) \right).$$

## 5

 Generating a transportation map

Theorem 16 guarantees we can obtain an approximately optimal flow in our graph  $G$  in nearly linear time. We wish to turn this flow into a transportation map on  $P$ . To accomplish this, we will repeatedly transform our flow into other flows, without increasing the cost, and ending at a flow which is also a transportation map. We will no longer keep the flow supported on the edges of  $E$ , and will instead allow positive flow between arbitrary pairs of points. We will gradually make sure that there is positive flow only between points in  $P$  (as opposed to net points).

We first begin by defining a notion we call uniform flow parity.

► **Definition 17.** Given a flow  $f \in \mathbb{R}^{V \times V}$ , and a vertex  $u \in V$ ,  $f$  is said to satisfy, or have uniform flow parity at  $u$  if for all remaining  $v \in V$ ,  $f_{uv}$  has the same sign. In other words, either there is flow going out of  $v$  or there is flow going into  $v$ , but there is no flow passing through  $v$ . The flow  $f$  is said to satisfy uniform flow parity if  $f$  has uniform flow parity on every  $v \in V$ .

The next easy lemma shows that a flow that satisfies uniform flow parity is supported on edges between vertices with non-zero divergence.

► **Lemma 18.** Suppose that a flow  $f \in \mathbb{R}^{V \times V}$  satisfies uniform flow parity at a vertex  $u \in V$ , and that  $\sum_{v \in V} f_{uv} = 0$ . Then  $f_{uv} = 0$  for all  $v \in V$ .

**Proof.** Assume, towards contradiction, that  $f$  has 0 divergence at  $u$  but  $f_{uv} \neq 0$  for some  $v \in V$ . Then there must be some  $w \in V$  such that  $f_{uw}$  has the opposite sign to  $f_{uv}$ , or  $\sum_{v \in V} f_{uv} = 0$  would not hold. This contradicts uniform flow parity. ◀

We apply Lemma 18 through the following corollary of it.

► **Corollary 19.** Let  $f$  be a flow on  $V \times V$ . Suppose that for any  $u \in V \setminus P$ ,  $\sum_{v \in V} f_{uv} = 0$  and for any  $p \in P$ ,  $\sum_{v \in V} f_{pv} = \mu(p)$ . Then, if  $f$  satisfies uniform flow parity, it is a transportation map for  $P$  and  $\mu$ .

**Proof.** By Lemma 18,  $f$  is supported on  $P \times P$ . Moreover, since for any  $p \in P_+$  we have  $\sum_{q \in P} f_{pq} = \mu(p) > 0$ , and  $f$  satisfies uniform flow parity, it must be the case that  $f_{pq} \geq 0$  for all  $q \in P$ . Similarly, for any  $p \in P_-$  it must be the case that  $f_{pq} \leq 0$  for all  $q \in P$ , or, equivalently,  $f_{qp} \geq 0$  for all  $q \in P$ . It follows that  $f$  is in fact supported on  $P_+ \times P_-$ , and is, therefore, a transportation map. ◀

Let us take a flow  $f$  which is an approximately optimal solution to (1)–(4). We can extend  $f$  to  $V \times V$  by setting  $f_{uv} = 0$  for  $(u, v) \notin E$ . If we can transform  $f$  into another flow  $f'$  without increasing its cost, so that  $f'$  satisfies uniform flow parity, then we can use  $f'$  as an approximately optimal transportation map. Towards this goal, we define the Cancellation Procedure  $\text{CANCEL-VERTEX}(f, u)$ , given in Algorithm 1.

---

**Algorithm 1** The Cancellation Procedure.

---

```

1: procedure CANCEL-VERTEX( $f, u$ )
2:   while  $\exists v, w : f_{vu} > 0 > f_{wu}$  do
3:      $x = \min\{f_{vu}, f_{uw}\}$ 
4:      $f_{vu} \leftarrow f_{vu} - x$ 
5:      $f_{uw} \leftarrow f_{uw} - x$ 
6:      $f_{vw} \leftarrow f_{vw} + x$ 
7:   end while
8: end procedure

```

---

The essential properties of  $\text{CANCEL-VERTEX}(f, u)$  are collected in the following lemma, whose proof is omitted from this extended abstract.

► **Lemma 20.** *Let  $f$  be a flow on  $V \times V$ , and  $u \in V$ . The while loop in  $\text{CANCEL-VERTEX}(f, u)$  makes at most  $|\{v : f_{uv} \neq 0\}|$  many iterations, and, letting  $f'$  be the flow after the procedure is called, we have the following properties:*

1. *All divergences are preserved, i.e.  $\sum_{w \in V} f'_{vw} = \sum_{w \in V} f_{vw}$  for all  $v \in V$ .*
2. *The flow  $f'$  satisfies uniform flow parity at  $u$ .*
3. *If  $f$  satisfies uniform flow parity at some vertex  $v \in V$ , then so does  $f'$ .*
4. *The size of the support of  $f$  is at most that of  $f'$ .*
5. *The cost of  $f'$  with respect to the cost function  $c(u, v) = \|u - v\|_2$  is at most the cost of  $f$ .*

Suppose that we maintain  $f$  in a sparse representation; namely, we keep an adjacency list of the edges on which  $f$  is not zero, with the corresponding flow values. Then, after preprocessing the adjacency list of  $u$  in linear time to find the edges with positive and negative flow, every iteration of the while loop in the Cancellation Procedure can be executed in constant time, and the total running time, by Lemma 20, is bounded by  $O(\deg_f(u))$ , where  $\deg_f(u) = |\{v : f_{uv} \neq 0\}|$ .

Corollary 19 and Lemma 20 imply that if we apply the Cancellation Procedure to the flow  $f$  and every vertex of the graph  $G$ , then the resulting flow has cost no greater than that of  $f$ , is supported on a set of edges of size bounded by  $|E|$ , and is a transportation map. In the next theorem, which is our main result for constructing a transportation map from a flow  $f$  in  $G$ , we show that if we apply the procedure first to netpoints in  $N_L$ , then to  $N_{L-1}$ , etc., then the total running time is nearly linear. The proof, which is deferred to the full version of the paper, simply accounts for the current degree of any node to which we apply the Cancellation Procedure.

► **Theorem 21.** *There exists an algorithm running in time  $O(n\varepsilon^{-d} \log(\Delta) + \varepsilon^{-2d} \log(\Delta))$  that, given as input a flow  $f$  which is feasible for the minimum cost flow problem defined in Section 3.3, outputs a transportation map for  $P$  and  $\mu$  of cost no larger than that of  $f$ .*

Combining Theorem 6, used with  $\varepsilon_0$  set to a sufficiently small multiple of  $\varepsilon/L$ , and Theorems 16 and 21 gives the first claim of Theorem 1, but with the approximation holding in expectation. At the cost of increasing  $\varepsilon$  by a factor of 2, Markov's inequality shows that the approximation also holds with probability at least  $1/2$ . Then the second statement in Theorem 1 follows from the first one, and Lemma 2.

---

## References

- 1 Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster Algorithms for the Geometric Transportation Problem. In *Symposium on Computational Geometry*, volume 77 of *LIPICs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.
- 2 Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *STOC*, pages 555–564. ACM, 2014.
- 3 Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Weed. Approximating the Quadratic Transportation Metric in Near-Linear Time. *CoRR*, abs/1810.10046, 2018.
- 4 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *STOC*, pages 574–583. ACM, 2014.
- 5 Sanjeev Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *J. ACM*, 45(5):753–782, 1998.

- 6 Nicolas Bonneel, Michiel van de Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement Interpolation Using Lagrangian Mass Transport. *ACM Trans. Graph.*, 30(6):158:1–158:12, December 2011. doi:10.1145/2070781.2024192.
- 7 Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388. ACM, 2002.
- 8 Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with  $O(1)$  worst case access time. *J. Assoc. Comput. Mach.*, 31(3):538–544, 1984. doi:10.1145/828.1884.
- 9 Panos Giannopoulos and Remco C Veltkamp. A pseudo-metric for weighted point sets. In *European Conference on Computer Vision*, pages 715–730. Springer, 2002.
- 10 Kristen Grauman and Trevor Darrell. Fast contour matching using approximate earth mover’s distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.
- 11 Piotr Indyk. A near linear time constant factor approximation for Euclidean bichromatic matching (cost). In *SODA*, pages 39–42. SIAM, 2007.
- 12 Piotr Indyk and Nitin Thaper. Fast Image Retrieval via Embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, 2003.
- 13 Qin Lv, Moses Charikar, and Kai Li. Image Similarity Search with Compact Data Structures. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM ’04*, pages 208–217, New York, NY, USA, 2004. ACM. doi:10.1145/1031171.1031213.
- 14 Jeff M. Phillips and Pankaj K. Agarwal. On Bipartite Matching under the RMS Distance. In *CCCG*, 2006.
- 15 Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.
- 16 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time  $\epsilon$ -approximation algorithm for geometric bipartite matching. In *STOC*, pages 385–394. ACM, 2012.
- 17 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *SODA*, pages 306–317. SIAM, 2012.
- 18 Jonah Sherman. Generalized Preconditioning and Undirected Minimum-Cost Flow. In *SODA*, pages 772–780. SIAM, 2017.
- 19 Cédric Villani. *Topics in optimal transportation*, volume 58 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2003. doi:10.1007/b12016.

# The One-Way Communication Complexity of Dynamic Time Warping Distance

**Vladimir Braverman**

Johns Hopkins University, Baltimore MD, USA  
vova@cs.jhu.edu

**Moses Charikar**

Stanford University, Stanford CA, USA  
moses@cs.stanford.edu

**William Kuszmaul**

Massachusetts Institute of Technology, Cambridge MA, USA  
kuszmaul@mit.edu

**David P. Woodruff**

Carnegie Mellon University, Pittsburgh PA, USA  
dwoodruf@cs.cmu.edu

**Lin F. Yang**

Princeton University, Princeton NJ, USA  
lin.yang@princeton.edu

---

## Abstract

We resolve the randomized one-way communication complexity of Dynamic Time Warping (DTW) distance. We show that there is an efficient one-way communication protocol using  $\tilde{O}(n/\alpha)$  bits for the problem of computing an  $\alpha$ -approximation for DTW between strings  $x$  and  $y$  of length  $n$ , and we prove a lower bound of  $\Omega(n/\alpha)$  bits for the same problem. Our communication protocol works for strings over an arbitrary metric of polynomial size and aspect ratio, and we optimize the logarithmic factors depending on properties of the underlying metric, such as when the points are low-dimensional integer vectors equipped with various metrics or have bounded doubling dimension. We also consider linear sketches of DTW, showing that such sketches must have size  $\Omega(n)$ .

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Communication complexity

**Keywords and phrases** dynamic time warping, one-way communication complexity, tree metrics

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.16

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.03520>.

**Funding** *Vladimir Braverman*: Supported by NSF CAREER grant 1652257, ONR Award N00014-18-1-2364, and the Lifelong Learning Machines program from DARPA/MTO.

*Moses Charikar*: Supported by NSF grant CCF-1617577 and a Simons Investigator Award.

*William Kuszmaul*: Supported by an MIT Akamai Fellowship and a Fannie & John Hertz Foundation Fellowship. Also supported by NSF grants 1314547 and 1533644.

*David P. Woodruff*: Partially supported by NSF Big Data grant 1447639.

*Lin F. Yang*: This work was done while the author was visiting IBM in 2017, hosted by David Woodruff, and supported by NSF CAREER grant 1652257.

**Acknowledgements** David P. Woodruff would like to thank the Simons Institute for the Theory of Computing where part of this work was done.



© Vladimir Braverman, Moses Charikar, William Kuszmaul, David P. Woodruff, and Lin F. Yang;

licensed under Creative Commons License CC-BY

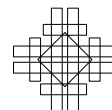
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 16; pp. 16:1–16:15

Leibniz International Proceedings in Informatics



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



## 1 Introduction

The Dynamic Time Warping (DTW) distance is a widely used distance measure between time series. It is particularly flexible in dealing with temporal sequences that vary in speed. To measure the distance between two sequences, each sequence is “warped” non-linearly in the time dimension (i.e., portions of each sequence are stretched by varying amounts) and the warped sequences are compared by summing up distances between corresponding elements. DTW was popularized in the speech recognition community by Sakoe and Chiba [34]. It was introduced in the data mining community for mining time series by Berndt and Clifford [9]. Its many applications include phone authentication [14], signature verification [29], speech recognition [28], bioinformatics [1], cardiac medicine [11], and song identification [38]. Several techniques and heuristics have been developed to speed up natural dynamic programming algorithms for it [19, 34, 25, 26, 24, 6, 33]. We refer the reader also to Section 2 of [4] for more references.

Distance measures on sequences and time series have been extensively studied in the literature. Given two sequences  $x = x_1, x_2, \dots, x_m$  and  $y = y_1, y_2, \dots, y_n$  of points in  $\mathbb{R}^d$  (or a metric space), one seeks to “match the points up” as closely as possible. One way of doing this is to define a “correspondence”  $(\bar{x}, \bar{y})$  between  $x, y$  by considering expansions of  $x$  and  $y$  to produce sequences of equal length, i.e., we duplicate each point  $x_i$  some number  $m_i$  times (to produce  $\bar{x}$ ) and each point  $y_j$  some  $n_j$  times (to produce  $\bar{y}$ ), so that  $\sum_{i=1}^m m_i = \sum_{j=1}^n n_j$ . Now, we define a vector  $z$  with  $z_i = d(\bar{x}_i, \bar{y}_i)$ , for some underlying distance function  $d$  and choose the correspondence which minimizes a certain function of  $z$ . For example, minimizing  $\sum z_i$  leads to the Dynamic Time Warping distance. Minimizing  $\max_i z_i$  leads to the discrete Fréchet distance. The edit distance between strings can be similarly cast in this framework. One unusual aspect of DTW (in contrast to its close cousins, edit distance and Fréchet distance) is that it does not satisfy the triangle inequality.

Edit distance and Fréchet distance have received a lot of attention in the theory community. Fundamental questions such as exact and approximation algorithms, nearest neighbor search, sketching, and communication complexity have been intensively studied. However, there are relatively few results about DTW. Similar to edit distance, DTW can be computed by a quadratic-time dynamic program. Recently, it was shown that there is no strongly subquadratic-time algorithm for DTW unless the Strong Exponential Time Hypothesis is false [10, 2]; approximation algorithms for DTW were obtained under certain assumptions about properties of the input strings [3, 37]; and slightly subquadratic algorithms for DTW have also been obtained [18]. DTW was studied in the context of LSH [15] and nearest neighbor search [35, 16]. To the best of our knowledge, until now, there has been no study of the communication complexity of this basic distance measure on sequences.

In this paper, we study the *one-way communication complexity* of DTW. For a distance measure  $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$  such as DTW, the goal in the one-way communication model is to define a randomized function  $S$  and an estimation procedure  $E$  so that for any  $x, y \in \mathcal{X}$ , given  $S(x)$  and  $y$ , the output  $E(S(x), y) \approx d(x, y)$  with large probability. There are various notions of approximation, but a natural one is that  $d(x, y) \leq E(S(x), y) < \alpha d(x, y)$  for an approximation factor  $\alpha > 1$ . The challenge is to understand how large  $S(x)$  needs to be (for sequences of length  $n$ ) in order to obtain approximation factor  $\alpha$ . A closely related notion is that of sketching, where the estimation procedure takes  $S(x)$  and  $S(y)$  and we require that  $E(S(x), S(y)) \approx d(x, y)$  with large probability. This one-way communication complexity question has been studied previously for edit distance, in the context of document exchange [7, 8, 20]. This model captures a number of applications, e.g., lower bounds in it apply to data



stream algorithms and to sketching protocols. Upper bounds in it are appropriate for nearest neighbor search; indeed, the natural thing to do here is a lookup table, so a (one-way) sketch of size  $b$  bits creates a table of size  $2^b$  (see e.g., [5]). One-way communication is one of the simplest and most natural settings in which one can study communication complexity, and it has rich connections to areas such as information theory, coding theory, on-line computing, and learning theory [27].

## 1.1 Our results

Our main result is a tight  $\tilde{\Theta}(n/\alpha)$  bound, up to logarithmic factors, on the one-way communication complexity of computing an  $\alpha$ -approximation to DTW. The results are discussed in more detail below.

We present a communication protocol using  $\tilde{\Theta}(n/\alpha)$  bits which works for DTW on any underlying metric space of polynomial size and aspect ratio (Theorem 8). We optimize the logarithmic factors in the important case when the points are natural numbers and the distance  $d(a, b) = |a - b|$ , as well as more generally when the points are low-dimensional integer vectors equipped with various metrics (Theorem 9); we also optimize for the important case where the underlying metric has small doubling dimension (Theorem 9). At the cost of an extra logarithmic factor in complexity, all of our protocols are also time-efficient, in that Alice and Bob each run in polynomial time.

Next, we turn to lower bounds. Our communication protocol is non-linear, and we show that in general linear sketches must have size  $\Omega(n)$  (Theorem 12). Moreover, we prove that our upper bounds are within a polylogarithmic factor of tight, establishing a randomized one-way communication lower bound of  $\Omega(n/\alpha)$  for any underlying metric space of size at least three, for one-way communication algorithms which succeed with constant success probability (Theorem 11). We optimize this in several ways: (1) when the underlying metric is generalized Hamming space over a point set of polynomial size  $n^{1+\Omega(1)}$ , we improve the lower bound to  $\Omega(n/\alpha \cdot \log n)$  for algorithms which succeed with probability  $1 - 1/n$ , and show this is optimal (Theorem 10); (2) for the natural numbers, we improve the lower bound to  $\Omega(n/\alpha \cdot \log(\min(\alpha, |\Sigma|)))$  for algorithms which succeed with probability at least  $1 - 1/\min(\alpha, |\Sigma|)$  (see the extended paper [36]). We note that our lower bound of  $\Omega(n/\alpha)$  applies even to approximating DTW in the low distance regime (i.e., distinguishing  $\text{DTW}(x, y) \leq 1$  versus  $\text{DTW}(x, y) > \alpha$  with constant probability), and that in this regime the edit distance admits a much smaller sketching complexity [7, 21]. To the best of our knowledge, our result provides the first separation between the DTW and the edit distance.

We summarize our results in Table 1. The layout of the paper is as follows: We present preliminaries in Section 2. We give a detailed overview of our techniques and results in Section 3. Then in Section 4 we give a complete treatment of several of the core results. A full presentation of all of the technical results appears in the extended paper [36].

## 2 Preliminaries

As a convention, we say an event occurs with *high probability* if it happens with probability at least  $1 - \frac{1}{\text{poly}(n)}$  for a polynomial of our choice. Throughout the paper, we use  $(\Sigma, d)$  to denote a finite metric space. We denote by  $\Sigma^n$  the set of strings of length  $n$  over  $\Sigma$  and by  $\Sigma^{\leq n}$  the set of strings of length at most  $n$  over  $\Sigma$ . An important property of  $\Sigma$  will be its *aspect ratio*, which is defined as the ratio between the diameter of  $\Sigma$  and the smallest distance between distinct points in  $\Sigma$ .

■ **Table 1** Summary of results on computing  $\alpha$  multiplicative approximation of  $\text{DTW}_n$  over a metric space  $\Sigma$  with aspect ratio  $\text{poly}(n)$ . \*These upper bounds are also time efficient. Inefficient protocols can remove an additional  $\log \alpha$  factor in the communication complexity. †These lower bounds hold for protocols that are correct with probability  $1 - 1/n$  or  $1 - 1/\min(\alpha, |\Sigma|)$ .

Model	Metric Space	Communication Bounds	Theorem
One-way	Finite	$O(n/\alpha \cdot \log \alpha \cdot \log^3 n)^*$	8
	Natural Numbers	$O(n/\alpha \cdot \log \alpha \cdot \log^2 n \cdot \log \log \log n)^*$	9
	$\ell_p^d$	$O_{p,d}(n/\alpha \cdot \log \alpha \cdot \log^2 n \cdot \log \log \log n)^*$	9
	doubling constant $\lambda$	$O(\log \lambda \cdot n/\alpha \cdot \log \alpha \cdot \log^2 n \cdot \log \log \log n)^*$	9
	Finite	$\Omega(n/\alpha)$	11
	Generalized Hamming	$\Theta(n/\alpha \cdot \log n)^\dagger$	10
Linear Sketch	Finite	$\Omega(n)$	12

### Dynamic Time Warping Distance

We study the *dynamic warping distance* (DTW) of strings  $x, y \in \Sigma^{\leq n}$ . Before we formally define the DTW distance, we first introduce the notion of an *expansion* of a string.

► **Definition 1.** *The runs of a string  $x \in \Sigma^{\leq n}$  are the maximal substrings consisting of a single repeated letter. Any string obtained from  $x$  by extending  $x$ 's runs is an expansion of  $x$ .*

For example, the runs of  $aabbcccd$  are  $aa$ ,  $bbb$ ,  $cc$ , and  $d$ . Given a string  $x$ , we can *extend* a run in  $x$  by further duplicating the letter which populates the run. For example, the second run in  $aabbcccd$  can be extended to obtain  $aabbbcccd$ , and we say the latter string is an expansion of the first.

Using the notion of an expansion, we can now define dynamic time warping.

► **Definition 2.** *Consider two strings  $x, y \in \Sigma^{\leq n}$ . A correspondence<sup>1</sup> between  $x$  and  $y$  is a pair  $(\bar{x}, \bar{y})$  of equal-length expansions of  $x$  and  $y$ . The edges in a correspondence are the pairs of letters  $(\bar{x}_i, \bar{y}_i)$ , and the cost of an edge is given by  $d(\bar{x}_i, \bar{y}_i)$ . The cost of a correspondence is the sum  $\sum_i d(\bar{x}_i, \bar{y}_i)$  of the costs of the edges between the two expansions. A correspondence between  $x$  and  $y$  is said to be optimal if it has the minimum attainable cost, and the resulting cost is called the dynamic time warping distance  $\text{DTW}(x, y)$ .*

When discussing a correspondence  $(\bar{x}, \bar{y})$ , the following terms will be useful.

► **Definition 3.** *A run in  $\bar{x}$  overlaps a run in  $\bar{y}$  if there is an edge between them. A letter  $x_i$  is matched to a letter  $y_j$  if the extended run containing  $x_i$  overlaps the extended run containing  $y_j$ .*

Note that any minimum-length optimal correspondence between strings  $x, y \in \Sigma^{\leq n}$  will be of length at most  $2n$ . In particular if in an optimal correspondence a run  $r_1$  in  $x$  and a run  $r_2$  in  $y$  have both been extended and overlap by at least one letter, then there is a shorter optimal correspondence in which the length of each run is reduced by one. Thus any minimum-length optimal correspondence has the property that every edge  $(\bar{x}_i, \bar{y}_i)$  contains at least one letter from a run that has not been extended, thereby limiting the length of the correspondence to at most  $2n$ .

<sup>1</sup> A related concept, *traversal*, is sometimes used in the literature. A traversal can be viewed as the set of matching edges of a correspondence.

DTW can be defined over an arbitrary metric space  $(\Sigma, d)$ , and is also well-defined when  $d$  is a distance function not satisfying the triangle inequality.

Throughout our proofs, we will often refer to DTW over generalized Hamming space, denoted by  $\text{DTW}_0(x, y)$ . As a convention, regardless of what metric space the strings  $x$  and  $y$  are initially taken over,  $\text{DTW}_0(x, y)$  is defined to be the DTW-distance between  $x$  and  $y$  obtained by redefining the distance function  $d(\cdot, \cdot)$  to return 1 on distinct inputs.

### One-Way Communication Complexity

In this paper, we focus on the one-way communication model. In this model, Alice is given an input  $x$ , Bob is given an input  $y$ , and Bob wishes to recover a valid solution to a problem with some solution-set  $f(x, y) \subseteq \mathbb{R}$ . (For convenience, we will refer to the problem by its solution set  $f(x, y)$ .) Alice is permitted to send Bob a single message  $\text{sk}(x)$ , which may be computed in a randomized fashion using arbitrarily many public random bits. Bob must then use Alice's message  $\text{sk}(x)$  in order to compute some  $F(\text{sk}(x), y)$ , which he returns as his proposed solution to  $f(x, y)$ .

The pair  $(\text{sk}, F)$  is a *p-accurate one-way communication protocol* for the problem  $f(\cdot, \cdot)$  if for all  $x$  and  $y$ , the probability  $\Pr[F(\text{sk}(x), y) \in f(x, y)]$  that Bob returns a correct answer to  $f(x, y)$  is at least  $p$ . The protocol is said to have *bit complexity* at most  $m$  if Alice's message  $\text{sk}(x)$  is guaranteed not to exceed  $m$  in length. Moreover, the protocol is said to be *efficient* if both  $\text{sk}$  and  $F$  can be evaluated in time polynomial in the length of  $x$  and  $y$ .

Fix a parameter  $p \in (0, 1]$ , the randomized *one-way communication complexity*  $\text{CC}_p(f)$  of the problem  $f$  is the minimum attainable bit complexity of a  $p$ -accurate one-way communication protocol for  $f$ . The focus of this paper is on the one-way communication complexity of the  $\alpha$ -DTW problem, defined as follows:

► **Definition 4** ( $\alpha$ -DTW). *The  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) problem is parameterized by an approximation parameter  $1 \leq \alpha \leq n$ . The inputs are a string  $x \in \Sigma^{\leq n}$  and a string  $y \in \Sigma^{\leq n}$ . The goal is recover an  $\alpha$ -approximation for  $\text{DTW}(x, y)$ . In particular, the set of valid solutions is*

$$\{t \mid \text{DTW}(x, y) \leq t < \alpha \cdot \text{DTW}(x, y)\}.$$

One can also consider the decision version of this problem, in which one wishes to distinguish between distances at most  $r$  and distances at greater than  $r\alpha$ :

► **Definition 5** (DTEP). *The Decision Threshold Estimation Problem  $\text{DTEP}_r^\alpha(\Sigma^{\leq n})$ , is parameterized by a positive threshold  $r > 0$  and an approximation parameter  $1 \leq \alpha \leq n$ . The inputs to the problem are a string  $x \in \Sigma^{\leq n}$  and a string  $y \in \Sigma^{\leq n}$ . An output of 0 is a valid solution if  $\text{DTW}(x, y) \leq r\alpha$ , and an output of 1 is a valid solution of  $\text{DTW}(x, y) > r$ .*

Notice that any algorithm for  $\alpha$ -DTW immediately gives an solution for  $\text{DTEP}_r^\alpha$  for any  $r > 0$ . Conversely, any lower bound for the communication complexity of  $\text{DTEP}_r^\alpha$  gives a lower bound for the communication complexity of  $\alpha$ -DTW. For both of the above two definitions, we may omit the sequence space  $\Sigma^{\leq n}$  if it is clear from the context.

## 3 Technical overview

In this section, we present the statements and proof overviews of our main results.

### Complexity Upper Bounds

Our starting point is the following: suppose that  $x, y \in \Sigma^n$  for a metric space  $\Sigma$  of polynomial size and aspect ratio, and further that the distances between points are always either 0 or at least 1. Alice and Bob wish to construct a  $2/3$ -accurate one-way protocol for  $\alpha$ -DTW.

**Collapsing Repeated Points.** Consider the strings  $c(x)$  and  $c(y)$ , formed by reducing each run of length greater than one in  $x$  and  $y$  to the same run of length one. If we define  $l$  to be the length of the longest run in  $x$  or  $y$ , then  $\text{DTW}(x, y) \leq l \cdot \text{DTW}(c(x), c(y))$ . Indeed, any correspondence  $(c(x), c(y))$  between  $c(x)$  and  $c(y)$  gives rise to a correspondence  $(\bar{x}, \bar{y})$  between  $x$  and  $y$  obtained by duplicating each coordinate in  $\bar{c(x)}$  and  $\bar{c(y)}$  a total of  $l$  times. Moreover, since any correspondence  $(\bar{x}, \bar{y})$  between  $x$  and  $y$  is also a correspondence between  $c(x)$  and  $c(y)$ , it follows that  $\text{DTW}(c(x), c(y)) \leq \text{DTW}(x, y)$ .

**Inefficient Protocol via Hashing.** Suppose Alice and Bob are guaranteed that  $\text{DTW}(x, y) \leq n/\alpha$ , and that the maximum run-length  $l$  satisfies  $l < \alpha$ . Then it suffices for Alice and Bob to compute  $\text{DTW}(c(x), c(y))$ ; and for this it suffices for Bob to be able to reconstruct  $c(x)$ . The claim is that from a random hash of  $c(x)$  of length  $O(n/\alpha \log n)$  bits, given  $c(y)$ , Bob can reconstruct  $c(x)$ . Indeed, given that  $\text{DTW}(c(x), c(y)) \leq n/\alpha$ , and given that the runs in  $c(x)$  and  $c(y)$  are all of length one, one can verify that there must be an optimal correspondence  $(\bar{c(x)}, \bar{c(y)})$  between  $c(x)$  and  $c(y)$  such that  $\bar{c(y)}$  is obtained from  $c(y)$  by extending at most  $n/\alpha$  runs. Since there are  $n^{O(n/\alpha)}$  ways to choose which runs in  $c(y)$  are extended, and since there are then  $n^{O(n/\alpha)}$  ways to choose the new lengths to which those runs are extended, it follows that there are only  $n^{O(n/\alpha)}$  options for  $\bar{c(y)}$ . Moreover, because  $\bar{c(x)}$  and  $\bar{c(y)}$  differ in at most  $n/\alpha$  positions, for a given option of  $\bar{c(y)}$  there are only  $n^{O(n/\alpha)} \cdot |\Sigma|^{O(n/\alpha)} = n^{O(n/\alpha)}$  options for  $\bar{c(x)}$  and thus for  $c(x)$ . Since starting from  $c(y)$ , there are only  $n^{O(n/\alpha)}$  options for  $c(x)$ , meaning that a  $O(n/\alpha \log n)$ -bit hash allows Bob to recover  $c(x)$  with high probability.

**Efficiency via Edit Distance Sketch.** In addition to requiring that  $\text{DTW}(x, y) \leq n/\alpha$  and  $l < \alpha$ , the above protocol is inefficient since Bob needs to enumerate over all possibilities of  $c(x)$  and compute the hash value of each. Exploiting the fact that  $c(x)$  and  $c(y)$  contain only runs of length one, we prove that  $\text{DTW}(c(x), c(y))$  is within a constant factor of the edit distance between  $c(x)$  and  $c(y)$ . This means that Alice can instead invoke the edit-distance communication protocol of [21] of size  $O(n/\alpha \log n \log \alpha)$ , which allows Bob to efficiently recover  $c(x)$  using the fact that the edit distance between  $c(x)$  and  $c(y)$  is  $O(n/\alpha)$ .

**Handling Heavy Hitters.** The arguments presented so far require that  $x$  and  $y$  contain no runs of length greater than  $\alpha$ . We call such runs *heavy hitters*. To remove this restriction, a key observation is that there can be at most  $n/\alpha$  heavy hitters. Therefore Alice can communicate to Bob precisely which runs are heavy hitters in  $x$  using  $O(n/\alpha \log n)$  bits. The players then proceed as before: Alice collapses her input  $x$  to  $c(x)$  by removing consecutive duplicates, and Bob collapses his input  $y$  to  $c(y)$  by removing consecutive duplicates. We still have  $\text{DTW}(c(x), c(y)) \leq \text{DTW}(x, y)$  since any correspondence between  $x$  and  $y$  is a correspondence between  $c(x)$  and  $c(y)$ . Thus, as before, Bob can reconstruct  $c(x)$  whenever  $\text{DTW}(x, y) \leq n/\alpha$ . Now, though, it could be that  $\text{DTW}(x, y) > \alpha \text{DTW}(c(x), c(y))$  because of the positions in  $c(x)$  and  $c(y)$  that occur more than  $\alpha$  times. However, Bob uses his knowledge of the locations and values of the heavy hitters, together with  $c(x)$ , to create a string  $x'$  formed from  $x$  by collapsing runs of length less than  $\alpha$ , and not doing anything to runs of length at least  $\alpha$ . Now by computing  $\text{DTW}(x', y)$ , Bob obtains a  $\alpha$ -approximation for  $\text{DTW}(x, y)$ , since any correspondence between  $x'$  and  $y$  gives rise to a correspondence between  $x$  and  $y$  by duplicating each letter  $\alpha$  times.

Having handled the heavy hitters, the only remaining requirement by our protocol is that the distances between letters in  $x$  and  $y$  be zero and one. Thus we arrive at the following:

► **Proposition 6** (Protocol over Hamming Space). *Consider DTW over a metric space  $\Sigma$  of polynomial size with distances zero and one. Then for  $p = 1 - \text{poly}(n^{-1})$ , there is an efficient  $p$ -accurate one-way communication protocol for  $\alpha$ -DTW over  $\Sigma^{\leq n}$  which uses  $O(n\alpha^{-1} \cdot \log \alpha \cdot \log n)$  bits. Moreover, for any  $\delta \in (0, 1)$ , there is an inefficient  $(1 - \delta)$ -accurate protocol for  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) using space  $O(n\alpha^{-1} \cdot \log n + \log \delta^{-1})$  for any  $\delta \in (0, 1)$ .*

Note that our protocol is constructive in that it actually allows for  $y$  to build a correspondence between  $x$  and  $y$  satisfying the desired approximation bounds.

In generalizing to DTW over arbitrary metric spaces, we will use our protocol over Hamming Space as a primitive. Moreover, we will exploit the fact that it can be used to solve a slightly more sophisticated problem which we call *bounded  $\alpha$ -DTW*:

► **Definition 7** (Bounded  $\alpha$ -DTW). *In the bounded  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) problem, Alice and Bob are given strings  $x$  and  $y$  in  $\Sigma^{\leq n}$ . The goal for Bob is:*

- If  $\text{DTW}_0(x, y) \leq n/\alpha$ , solve  $\alpha$ -DTW on  $(x, y)$ .
- If  $\text{DTW}_0(x, y) > n/\alpha$ , either solve  $\alpha$ -DTW on  $(x, y)$ , or return “Fail”.

A crucial observation is that Proposition 6 continues to hold without modification if the alphabet  $\Sigma$  has arbitrary distances and our goal is to solve the bounded  $\alpha$ -DTW problem.

**Extending Distance Range via HSTs.** The result for the bounded  $\alpha$ -DTW problem allows for Bob to either determine an  $\alpha$ -approximation for  $\text{DTW}(x, y)$ , or to determine that  $\text{DTW}(x, y) > n/\alpha$ . As a result the algorithm can be used to distinguish between  $\text{DTW}(x, y) \leq n/\alpha$  and  $\text{DTW}(x, y) > n$ . One issue though is that the argument cannot distinguish between larger distances, such as for example between the cases  $\text{DTW}(x, y) \leq n$  and  $\text{DTW}(x, y) > n\alpha$ . A key idea for resolving this issue is to first consider the DTW problem over a 2-hierarchically well-separated tree metric (HST), and then use the embedding of [17] to embed an arbitrary finite metric of polynomial size and aspect ratio into such a metric. A 2-hierarchically well-separated tree metric is defined as the shortest path metric on a tree whose nodes are elements of  $\Sigma$  and whose edges have positive weights for which on any root-to-leaf path, the weights are geometrically decreasing by a factor of 2. Since the weights decrease geometrically, for convenience we define pairwise distances in the tree metric to be the maximum edge length on the tree path between the nodes, a notion of distance which coincides with the sum of edge lengths up to a constant factor.

Suppose the points in  $\Sigma$  correspond to a 2-hierarchically well-separated tree metric and we wish to distinguish between whether  $\text{DTW}(x, y) \leq nr/\alpha$  or  $\text{DTW}(x, y) > nr$ . A crucial idea is what we call the  $r$ -simplification  $s_r(x)$  of a string  $x$ , which replaces each character  $p_i$  in  $x$  with its highest ancestor in the tree reachable via edges of weight at most  $r/4$ . A key property is that  $\text{DTW}(s_r(x), s_r(y)) \leq \text{DTW}(x, y)$ , since for two points  $\ell_1, \ell_2$  in  $x, y$ , respectively, either they each get replaced with the same point in the  $r$ -simplifications of  $x$  and  $y$ , or the maximum-length edge on a path between  $\ell_1$  and  $\ell_2$  is the same before and after  $r$ -simplification. Notice that if a point in  $s_r(x)$  is not equal to a point in  $s_r(y)$ , then their distance is at least  $r/4$ , by the definition of an  $r$ -simplification. Combining the preceding two observations, if  $\text{DTW}(x, y) \leq nr/\alpha$ , then  $\text{DTW}(s_r(x), s_r(y)) \leq nr/\alpha$  and there is a correspondence for which  $s_r(x)$  and  $s_r(y)$  disagree in at most  $4n/\alpha$  positions. On the other hand, since we only “collapse” edges of weight at most  $r/4$ , we have that if  $\text{DTW}(x, y) > nr$ , then  $\text{DTW}(s_r(x), s_r(y)) > nr/2$ , since the optimal correspondence has length at most  $2n$ .

It follows that the cases of  $\text{DTW}(x, y) \leq nr/\alpha$  and  $\text{DTW}(x, y) > nr$ , correspond with the cases of  $\text{DTW}(s_r(x), s_r(y)) \leq nr/\alpha$  and  $\text{DTW}(s_r(x), s_r(y)) > nr/2$ , and moreover that when  $\text{DTW}(s_r(x), s_r(y)) \leq nr/\alpha$ , there is an optimal correspondence for which  $s_r(x)$  and

$s_r(y)$  disagree in at most  $4n/\alpha$  positions. Thus we can use our protocol for the  $\alpha$ -bounded DTW problem to figure out which case we are in, for a given  $r$ . This gives a protocol for distinguishing between whether  $\text{DTW}(x, y) \leq nr/\alpha$  or  $\text{DTW}(x, y) > nr$ .

In order to obtain an  $\alpha$ -approximation for  $\text{DTW}(x, y)$ , the rough idea now is to run the above protocol multiple times in parallel as  $r$  varies in powers of 2, and then to find the smallest value of  $r$  for which the protocol declares  $\text{DTW}(x, y) \leq nr$ . This works as long as points are taken from a 2-hierarchically well-separated tree metric. In order to extend the result to hold over arbitrary finite metrics of polynomial size and aspect ratio, the final piece is the embedding  $\phi$  of [17], which embeds any polynomial size metric  $\Sigma$  into a 2-hierarchically well-separated tree metric for which for all  $a, b \in \Sigma$ ,  $d(a, b) \leq d(\phi(a), \phi(b))$  and  $\mathbf{E}(d(\phi(a), \phi(b))) = O(\log n)d(a, b)$ . This “lopsided” guarantee is sufficient for us since it ensures in any correspondence the sum of distances after performing the embedding will not shrink, while for a single fixed optimal correspondence, by a Markov bound the sum of distances after performing the embedding will not increase by more than an  $O(\log n)$  factor with constant probability. Putting the pieces together we are able to obtain an efficient 2/3-accurate one-way communication protocol for  $\alpha$ -DTW using  $O(n/\alpha \log \alpha \log^3 n)$  bits. Formally, we arrive at the following theorem:

► **Theorem 8 (Main Upper Bound).** *Let  $\Sigma$  be a metric space of size and aspect ratio polynomial in  $n$ . Then there is an efficient 2/3-accurate one-way communication protocol for  $\alpha$ -DTW over  $\Sigma$  with space complexity  $O(n\alpha^{-1} \cdot \log \alpha \cdot \log^3 n)$  and an inefficient 2/3-accurate one-way protocol with complexity  $O(n\alpha^{-1} \cdot \log^3 n)$ .*

**Optimizing in the Case of Natural Numbers.** We can further optimize the logarithmic factors in our upper bound when the underlying alphabet  $\Sigma$  is, for example, the natural numbers and  $d(a, b) = |a - b|$ . We handle the case  $\text{DTW}(x, y) \leq n/\alpha$  as before. However, for larger values of  $\text{DTW}(x, y)$ , we take a different approach.

We first explain the case of distinguishing  $\text{DTW}(x, y) \leq n$  versus  $\text{DTW}(x, y) > \alpha n$ . The idea is to impose a randomly shifted grid of side length  $\alpha/4$ , and to round each point in  $x$  and  $y$  down to the nearest smaller grid point, resulting in strings  $x'$  and  $y'$ . Define a *short edge* in a correspondence to be an edge of cost at most  $\alpha/4$ , and otherwise call the edge a *long edge*. We assume w.l.o.g. that any correspondence has length at most  $2n$ .

Suppose first  $\text{DTW}(x, y) \leq n$ , and consider an optimal correspondence. We will show that the effect of rounding is such that with probability at least 2/3,  $\text{DTW}(x', y') \leq O(n)$ . First we consider what effect rounding has on the short edges. The expected number of short edges with endpoints that get rounded to different grid points is at most

$$\sum_{\text{short edge length } l} \frac{l}{\alpha/4} \leq \frac{4 \text{DTW}(x, y)}{\alpha}.$$

Each such edge has its length increased by at most  $\alpha/4$  after rounding, and so the expected contribution of short edges to the correspondence after rounding is at most  $O(\text{DTW}(x, y))$ . Since each long edge has its length increase by at most an additive  $\alpha/4$ , and its original length is at least  $\alpha/4$ , its contribution changes by at most a constant factor, so the total contribution of long edges after rounding is  $O(\text{DTW}(x, y))$ . Hence, when  $\text{DTW}(x, y) \leq n$ , with probability at least 2/3 after rounding, we have  $\text{DTW}(x', y') = O(n)$ .

Next suppose  $\text{DTW}(x, y) > n\alpha$ , and consider any correspondence. The total change in the cost of the correspondence that can result from the rounding procedure is at most  $2n \cdot \alpha/4$ , since there are at most  $2n$  edges in total. Consequently the effect of rounding is such that  $\text{DTW}(x', y') > n\alpha/2$ .



It follows that when comparing the cases of  $\text{DTW}(x, y) \leq n$  and  $\text{DTW}(x, y) > n\alpha$ , there is an  $\Omega(\alpha)$ -factor gap between  $\text{DTW}(x', y')$  in the two cases. Further, after rounding to grid points, all non-equal points have distance at least  $\alpha/4$ , and so if  $\text{DTW}(x', y') \leq n$ , then there is a correspondence on which they differ in at most  $O(n/\alpha)$  positions. Thus our protocol for bounded  $\alpha$ -DTW can be applied to distinguish between the two cases. A similar approach can be used to distinguish between  $\text{DTW}(x, y) \leq rn/\alpha$  and  $\text{DTW}(x, y) > rn$  in general, and this can then be used to solve  $\alpha$ -DTW similarly as for 2-hierarchically well-separated tree metrics above. We save roughly a  $\log n$  factor here because we do not incur the  $\log n$  factor distortion of embedding an arbitrary metric into a tree metric.

We remark that our algorithm in the 1-dimensional natural number case uses a similar grid snapping as used in [15] for their nearest neighbor search algorithm for Frechét distance. Recently, Bringmann (personal communication) obtained a sketch for Frechét distance which builds upon the ideas in [15] and uses  $O(n/\alpha)$  bits. To the best of our knowledge, these techniques do not yield nontrivial results for Dynamic Time Warping, however.

**A Unified Approach.** To unify the argument for 2-hierarchically well-separated tree metrics and the natural numbers, we recall the definition of a  $\sigma$ -separable metric space. A  $\delta$ -bounded partition of a metric space  $(\Sigma, d)$  is a partition such that the diameter of each part is at most  $\delta$ . A distribution over partitions is then called  $\sigma$ -separating if for all  $x, y \in \Sigma$ , the probability that  $x$  and  $y$  occur in different parts of the partition is at most  $\sigma \cdot d(x, y)/\delta$ . We say  $\Sigma$  is  $\sigma$ -separable if for every  $\delta > 0$ , there exists a  $\sigma$ -separating probability distribution over  $\delta$ -bounded partitions of  $\Sigma$ . One can also define an efficient notion of this, whereby the distribution over partitions is efficiently sampleable.

By adapting our argument for the natural numbers to  $\sigma$ -separable metrics of polynomial size and aspect ratio, we obtain an efficient  $2/3$ -accurate protocol for  $\alpha$ -DTW with bit complexity  $O(\sigma n/\alpha \log^3 n \log \log \log n)$ , where the  $\log \log \log n$  comes from minor technical subtleties. For general metrics, it is known that  $\sigma = O(\log n)$ , while for the natural numbers,  $\sigma = O(1)$ . Consequently, our result for  $\sigma$ -separable metrics captures both the result obtained using HSTs (up to a factor of  $\log \log \log n$ ) as well as the optimization for the natural numbers. Moreover, the theorem allows for space savings over many additional metrics, such as low-dimensional integer vectors equipped with  $\ell_p$ -norms, metrics with bounded doubling dimension, etc., all of which have  $\sigma \ll O(\log n)$  [13, 30, 31]. The general result we arrive at is captured formally in the following theorem:

► **Theorem 9** (Extended Main Upper Bound). *Let  $(\Sigma, d)$  be a metric space of size and aspect ratio  $\text{poly}(n)$ . Suppose that  $(\Sigma, d)$  is efficiently  $\sigma$ -separable for some  $1 \leq \sigma \leq O(\log n)$ . Then there is an efficient  $2/3$ -accurate one-way communication protocol for  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) with space complexity  $O(\sigma n \alpha^{-1} \cdot \log \alpha \cdot \log^2 n \cdot \log \log \log n)$  and an inefficient  $2/3$ -accurate one-way protocol with space complexity  $O(\sigma n \alpha^{-1} \cdot \log^2 n \cdot \log \log \log n)$ .*

The proof closely follows that for the natural numbers, where instead of our randomly shifted grid, we use a random  $\delta$ -bounded partition. If we are trying to distinguish  $\text{DTW}(x, y) \leq nr/\alpha$  versus  $\text{DTW}(x, y) > nr$ , then we set  $\delta = \Theta(r)$ . Just like for the grid, where we “snapped” points to their nearest grid point, we now snap points to a representative point in each part of the partition, obtaining two new sequences  $\tilde{x}$  and  $\tilde{y}$ . By using shared randomness, the representative in each part can be agreed upon without any communication. Just like in the grid case, we show that if  $\text{DTW}(x, y) \leq nr/\alpha$ , then for the optimal correspondence, in expectation its cost increases only by a constant factor after snapping. On the other hand, if  $\text{DTW}(x, y) > nr$ , then we show that for every correspondence, its cost decreases only by a constant factor. The key difference is that now the expected number of short edges with endpoints occurring in different parts of the partition is at most  $\frac{\sigma \cdot \text{DTW}(x, y)}{\delta}$ .



### Complexity Lower Bounds

The simplest of our lower bounds comes from a reduction from a randomized 1-way communication lower bound for indexing over large alphabets [22]. In this problem, Alice is given a string  $s$  in  $\mathcal{U}^r$  for some universe  $\mathcal{U}$  and length parameter  $r$ , and Bob is given a character  $a \in \mathcal{U}$  and an index  $j \in [r]$ . The goal is for Bob to decide if  $s_j = a$  with probability at least  $1 - 1/|\mathcal{U}|$ . It is known if Alice sends a single message to Bob, then there is an  $\Omega(r \log_2 |\mathcal{U}|)$  lower bound. By reducing this large-alphabet indexing problem to  $\alpha$ -DTW when  $r = n/\alpha$ . To perform the reduction, Alice's input string  $s = s_1, \dots, s_{n/\alpha} \in \mathcal{U}^{n/\alpha}$  is mapped to the string  $x = (s_1, 1), (s_2, 2), \dots, (s_{n/\alpha}, n/\alpha)$ . Bob's inputs of  $a \in \mathcal{U}$  and  $j \in [r]$  are mapped to an input string  $y = (a, j), (a, j), \dots$  in which the character  $(a, j)$  is repeated  $n$  times. If  $s_j = a$ , then  $\text{DTW}(x, y) = n/\alpha - 1$  (due to the  $n/\alpha - 1$  characters of  $x$  that do not get matched with an equal-value letter); otherwise  $\text{DTW}(x, y) \geq n$  (due to the fact that none of the letters in  $y$  can be correctly matched). This gives a reduction to  $\alpha$ -DTW as desired. Using this we have an  $\Omega(n/\alpha \cdot \log n)$  lower bound for  $(1 - 1/n)$ -accurate  $\alpha$ -DTW, provided the alphabet size  $|\Sigma|$  is say, at least  $n^2$ . Thus we have the following theorem:

► **Theorem 10 (Tight Bound Over Hamming Space).** *Consider  $1 \leq \alpha \leq n$ , and consider the generalized Hamming distance over a point-set  $\Sigma$  with  $\Sigma$  of polynomial size  $n^{1+\Omega(1)}$ . For  $p \geq 1 - 1/|\Sigma|^{-1}$ ,  $\text{CC}_p(\alpha\text{-DTW}(\Sigma^{\leq n})) = \Theta[n\alpha^{-1} \cdot \log n]$ .*

In order to obtain a nearly tight lower bound for arbitrary finite metric spaces, we construct a more intricate lower bound of  $\Omega(n/\alpha)$  which holds whenever  $|\Sigma| \geq 3$ . For convenience, we describe the argument for the case of  $\Sigma = \{0, 1, 2\}$  below. The lower bound is achieved via a reduction from the Index problem in which Alice has  $s \in \{0, 1\}^t$ , Bob has an  $i \in [t]$ , and Bob would like to output  $s_i$  with probability at least  $2/3$ . The randomized 1-way communication complexity of this problem is  $\Omega(t)$ . We instantiate  $t = \Theta(n/\alpha)$ . For each  $s_j$ , if  $s_j = 1$ , Alice creates a string  $Z(1)$  of length  $3\alpha$  consisting of  $\alpha$  0s, followed by  $\alpha$  1s, followed by  $\alpha$  2s; and if  $s_j = 0$ , Alice creates a string  $Z(0)$  of length  $2\alpha + 1$  consisting of  $\alpha$  0s, followed by a single 1, followed by  $\alpha$  2s. She then concatenates  $Z(s_1), Z(s_2), \dots, Z(s_t)$  into a single string  $x$  of length  $n$ . Bob, who is given an index  $i \in [t]$ , creates the string  $y = (012)^{i-1}(02)(012)^{t-i}$ ; that is, we have the length-3 string 012 repeated  $i - 1$  times, then the string 02, followed by the string 012 repeated  $t - i$  times. (We call each piece of the form (012) and (01) a *block*.) Notice that if  $s_i = 0$ , then  $\text{DTW}(x, y) = 1$ , since the single 1 in  $Z(s_i)$  can match to either the 0 or 2 in the (02) block of Bob's string  $y$ . On the other hand, if  $s_i = 1$ , the entire run of  $\alpha$  1s in  $Z(s_i)$  has to appear somewhere in the correspondence and cannot match to the 0 or 2 in the  $i$ -th piece of Bob's string, without incurring a cost of  $\alpha$ . So these  $\alpha$  1s must either "travel" to blocks  $j > i$  in  $y$  or blocks  $j < i$  in  $y$ . Suppose, without loss of generality, most of these  $\alpha$  1s are matched to a block  $j > i$ . This has a ripple effect, since it causes the  $\alpha$  2s in the  $i$ -th block to also have to travel to a block  $j > i$ . While this is possible, it then means the  $\alpha$  0s in the  $(i + 1)$ -st block must travel to a block even larger than  $j$ , etc. Eventually, we run out of blocks to match the elements in Alice's string to since there are  $t - i$  blocks in her string that need to be matched to fewer than  $t - i$  blocks in Bob's string. This ultimately forces  $\text{DTW}(x, y) \geq \alpha$ , completing the reduction from the Index problem to  $\alpha$ -DTW. The extension of this argument to arbitrary  $\Sigma$  establishes that our upper bound for general metric spaces is optimal up to a polylogarithmic factor:

► **Theorem 11 (General Lower Bound).** *Let  $\Sigma = \{a, b, c\}$  be three letters with a two-point distance function  $d : \Sigma \times \Sigma \rightarrow \mathbb{R}_+$ , not necessarily satisfying the triangle inequality. Consider  $1 \leq \alpha \leq n$ . Then  $\text{CC}_{0.1}[\text{DTEP}_r^\alpha(\Sigma^{\leq n})] = \Omega(n/\alpha)$ .*

Our communication protocols are non-linear, and we conclude our lower bounds by observing that linear sketches must have size  $\Omega(n)$ . (See the extended paper [36].)

► **Theorem 12** (Linear Sketching Lower Bound). *Consider  $1 \leq \alpha \leq n$ . Then any 0.1-error linear sketch for  $\alpha$ -DTW on  $\{0, 1, 2\}^{4n}$  has space complexity  $\Omega(n)$ .*

#### 4 The bounded $\alpha$ DTW problem

Recall that in the Bounded  $\alpha$ -DTW problem, the goal for Bob is: If  $\text{DTW}_0(x, y) \leq n/\alpha$ , solve  $\alpha$ -DTW on  $(x, y)$ ; and if  $\text{DTW}_0(x, y) > n/\alpha$ , either solve  $\alpha$ -DTW on  $(x, y)$ , or return “Fail”. In this section, we formally present the protocol for bounded  $\alpha$ -DTW. We then use this to give tight bounds on the one-way communication complexity of  $\alpha$ -DTW over generalized Hamming Space. The protocols in this section are constructive in that, in addition to estimating  $\text{DTW}(x, y)$ , Bob is also able to build a correspondence between  $x$  and  $y$ . The protocol for Bounded  $\alpha$ -DTW forms the core for the communication protocol over arbitrary metrics. The full protocol over arbitrary metrics is given in the extended paper [36].

In order to design an efficient one-way communication scheme for bounded  $\alpha$ -DTW, we will use what we refer to as the *K-document exchange* problem as a primitive. Here, Alice and Bob are given strings  $x$  and  $y \in \Sigma^n$ . The goal for Bob is: If  $\text{ed}(x, y) \leq K$ , recover the string  $x$ ; and if  $\text{ed}(x, y) > K$ , either recover  $x$  or return “Fail”.

The *K-document exchange* problem has been studied extensively [32, 23, 8, 12, 21]. The one-way communication protocol of [21] efficiently solves *K-document exchange* using  $O(K \log(n/K) \cdot \log n)$  bits with high probability. This can be slightly improved at the cost of being no longer time-efficient using the protocol of [32], which achieves accuracy  $1 - \delta$  for any  $\delta \in (0, 1)$  by having Alice simply hash her string to a  $\Theta(K \cdot \log n + \log \delta^{-1})$ -bits.

The *K-document exchange* problem concerns edit distance rather than DTW. Nonetheless, in designing a sketch for DTW, the *K-document exchange* problem will prove useful due to a convenient relationship between edit distance and DTW over generalized Hamming space.

► **Lemma 13** (DTW<sub>0</sub> Approx. Edit Dist.). *Let  $x, y$  be strings of length at most  $n$  with letters from any metric space, and suppose that neither string contains any runs of length greater than one. Then  $\text{DTW}_0(x, y) \leq \text{ed}(x, y) \leq 3 \text{DTW}_0(x, y)$ .*

**Proof.** We first show that  $\text{DTW}_0(x, y) \leq \text{ed}(x, y)$ . A sequence of edits from  $x$  to  $y$  can be viewed as consisting of insertions in each of  $x$  and  $y$ , as well as substitutions. One can create expansions  $\bar{x}$  and  $\bar{y}$  of  $x$  and  $y$ , respectively, by extending runs by one in each place where the sequence of edits would have performed an insertion. The Hamming distance between  $\bar{x}$  and  $\bar{y}$  is then at most the length of the sequence of edits. Hence  $\text{DTW}_0(x, y) \leq \text{ed}(x, y)$ .

Next we show that  $\text{ed}(x, y) \leq 3 \text{DTW}_0(x, y)$ . Consider an optimal correspondence  $(\bar{x}, \bar{y})$  between  $x$  and  $y$ . Without loss of generality, we may assume that whenever two runs in  $\bar{x}$  and  $\bar{y}$  overlap, at least one of them has length only one. (Indeed, otherwise both runs could have been reduced in size by one at no cost to DTW.) Therefore, any run of length  $k$  in  $\bar{x}$  must overlap  $k$  distinct runs in  $\bar{y}$ , and thus must incur at least  $(k - 1)/2$  Hamming differences. On the other hand, because the run is length  $k$ , the expansion of the run can be simulated by  $k - 1$  insertions. Therefore,  $\bar{x}$  and  $\bar{y}$  can be constructed from  $x$  and  $y$  through at most  $2 \text{DTW}_0(x, y)$  edits. Hence,  $\text{ed}(x, y) \leq 3 \text{DTW}_0(x, y)$ . ◀

We now present an efficient one-way communication scheme for bounded  $\alpha$ -DTW.

## 16:12 One-Way Communication Complexity of DTW

► **Proposition 14** (Protocol for Bounded DTW). *Consider DTW over a metric space  $\Sigma$  of polynomial size. Then for  $p = 1 - \text{poly}(n^{-1})$ , there is an efficient  $p$ -accurate one-way communication protocol for bounded  $\alpha$ -DTW over  $\Sigma^{\leq n}$  which uses  $O(n\alpha^{-1} \cdot \log \alpha \cdot \log n)$  bits. Moreover, for any  $\delta \in (0, 1)$ , there is an inefficient  $(1 - \delta)$ -accurate protocol for bounded  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) using space  $O(n\alpha^{-1} \cdot \log n + \log \delta^{-1})$  for any  $\delta \in (0, 1)$ .*

**Proof.** We assume without loss of generality that  $\alpha$  and  $n/\alpha$  are integers. Let  $x \in \Sigma^{\leq n}$  be a string given to Alice, and  $y \in \Sigma^{\leq n}$  be a string given to Bob. Alice can construct a string  $x'$  by taking each run in  $x$  which is of length less than  $\alpha$  and reducing its length to one. Notice that  $\text{DTW}(x', y) \leq \text{DTW}(x, y)$  trivially and that  $\text{DTW}(x, y) < \alpha \text{DTW}(x', y)$  because any correspondence between  $x'$  and  $y$  can be turned into a correspondence between  $x$  and  $y$  by duplicating every letter in the original correspondence  $\alpha - 1$  times. Thus if Alice could communicate  $x'$  to Bob, then Bob could solve  $\alpha$ -DTW.

In an attempt to communicate  $x'$  to Bob, Alice constructs a list  $L$  consisting of the pairs  $(i, l_i)$  for which the  $i$ -th run in  $x$  is of length  $l_i \geq \alpha$ . Alice then sends  $L$  to Bob. Notice that  $|L| \leq n/\alpha$ , and thus can be communicated with  $O(\frac{n}{\alpha} \log n)$  bits. Moreover, if Alice defines  $x''$  to be  $x$  except with every run reduced to length one, then  $x'$  can be recovered from  $x''$  and  $L$ . Therefore, if Alice could further communicate  $x''$  to Bob, then Bob could solve  $\alpha$ -DTW.

In an attempt to communicate  $x''$  to Bob, Alice invokes the one-way communication protocol of [21] for the  $3n/\alpha$ -document exchange problem. She sends Bob the resulting sketch  $s$  of size  $O(n/\alpha \cdot \log \alpha \log n)$  bits which is correct with probability at least  $p$ . Bob defines  $y''$  to be  $y$  with each run reduced to length one and uses the sketch  $s$  along with  $y''$  in order to try to recover  $x''$ . If Bob is able to use  $s$  to recover a value for  $x''$ , then he can correctly solve  $\alpha$ -DTW with high probability. If Bob is unable to use  $s$  to recover a value for  $x''$ , then Bob may conclude with high probability that  $\text{ed}(x'', y'') > 3n/\alpha$ . Because  $\text{ed}(x'', y'') \leq 3 \text{DTW}_0(x'', y'')$  by Lemma 13 and because  $\text{DTW}_0(x'', y'') \leq \text{DTW}_0(x, y)$ , we have that  $n/\alpha < \text{DTW}_0(x, y)$ . It follows that in this case Bob can correctly return “Fail”.

Rather than using the efficient one-way communication protocol of [21], Alice could instead invoke the protocol of [32] in which she sends Bob a hash of  $x''$  using  $O(n\alpha^{-1} \cdot \log n + \log \delta^{-1})$  and Bob is able to then inefficiently recover  $x''$  correctly with probability at least  $1 - \delta$ . This gives an inefficient  $(1 - \delta)$ -accurate protocol which uses  $O(n\alpha^{-1} \cdot \log n + \log \delta^{-1})$  bits. ◀

We now prove a tight communication bound for  $\alpha$ -DTW over generalized Hamming space.

► **Theorem 15** (Theorem 10 Restated). *Consider  $1 \leq \alpha \leq n$ , and consider the generalized Hamming distance over a point-set  $\Sigma$  with  $\Sigma$  of polynomial size  $n^{1+\Omega(1)}$ . For  $p \geq 1 - 1/|\Sigma|^{-1}$ , the  $p$ -accurate one-way communication complexity of  $\alpha$ -DTW( $\Sigma^{\leq n}$ ) is  $\Theta[n\alpha^{-1} \cdot \log n]$ .*

Proposition 14 implies the desired upper bound (via the inefficient protocol). In order to prove Theorem 15, it therefore suffices to prove the lower bound. To do this, we first introduce a problem with high one-way communication complexity.

► **Lemma 16** ([22], Theorem 3.1). *Define  $(n, \mathcal{S})$ -SET as follows. Alice gets an  $n$ -element set  $S \subseteq \mathcal{S}$  and Bob gets a character  $a \in \mathcal{S}$ . The goal is for Bob to determine whether  $a \in S$ . Let  $p \geq 1 - \frac{1}{|\mathcal{S}|}$ . Then  $\text{CC}_p((n, \mathcal{S})\text{-SET}) \geq \Omega(n \log(|\mathcal{S}|/n))$ .*

**Proof of Theorem 15.** As described above, it suffices to show the lower bound. To this end, we reduce  $(n/\alpha, \Sigma)$ -SET to  $\alpha$ -DTW for strings of length  $n$ . Suppose Alice is given  $S \subseteq \Sigma$  of size  $n/\alpha$  and Bob is given the character  $a \in \Sigma$ . Then Alice can compute  $x$  to be the concatenation of the elements of  $S$  in an arbitrary order. Alice will use the resulting string  $x \in \Sigma^{\leq n}$  as an input for the  $\alpha$ -DTW problem. Bob can then define  $y$  to

be the character  $a$  repeated  $n$  times. Notice that if  $a \in S$  then  $\text{DTW}(x, y) = n/\alpha - 1$ , whereas if  $a \notin S$  then  $\text{DTW}(x, y) = n$ . By Lemma 16, this reduction establishes that for  $p \geq 1 - \frac{1}{|\Sigma|}$  the  $p$ -accurate one-way communication complexity of  $\alpha$ -DTW is at least  $\Omega(n\alpha^{-1} \cdot \log(\alpha|\Sigma|/n)) = \Omega(n\alpha^{-1} \cdot \log n)$ , since  $|\Sigma| = n^{1+\Omega(1)}$ . ◀

---

## References

- 1 John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.
- 2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 59–78. IEEE, 2015.
- 3 Pankaj K Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying. Approximating dynamic time warping and edit distance for a pair of point sequences. *arXiv preprint*, 2015. [arXiv:1512.01876](#).
- 4 Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. Sparsedt看w: A novel approach to speed up dynamic time warping. In *Proceedings of the Eighth Australasian Data Mining Conference-Volume 101*, pages 117–127. Australian Computer Society, Inc., 2009.
- 5 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 324–330. IEEE, 2009.
- 6 Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn J. Keogh. Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, pages 49–58, 2015.
- 7 D. Belazzougui and Q. Zhang. Edit Distance: Sketching, Streaming and Document Exchange. *ArXiv e-prints*, 2016. [arXiv:1607.04200](#).
- 8 Djamal Belazzougui. Efficient deterministic single round document exchange for edit distance. *arXiv preprint*, 2015. [arXiv:1511.09229](#).
- 9 Donald J. Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, July 1994. Technical Report WS-94-03*, pages 359–370, 1994.
- 10 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 79–97. IEEE, 2015.
- 11 EG Caiani, A Porta, G Baselli, M Turiel, S Muzzupappa, F Pieruzzi, C Crema, A Malliani, and S Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In *Computers in Cardiology 1998*, pages 73–76. IEEE, 1998.
- 12 Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 712–725. ACM, 2016.
- 13 Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 379–388. IEEE, 1998.
- 14 Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.
- 15 Anne Driemel and Francesco Silvestri. Locality-Sensitive Hashing of Curves. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 37:1–37:16, 2017.

- 16 Ioannis Z Emiris and Ioannis Psarros. Products of Euclidean metrics and applications to proximity questions among curves. *arXiv preprint*, 2017. [arXiv:1712.06471](#).
- 17 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.
- 18 Omer Gold and Micha Sharir. Dynamic time warping and geometric edit distance: Breaking the quadratic barrier. *arXiv preprint*, 2016. [arXiv:1607.05994](#).
- 19 Daniel S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Commun. ACM*, 18(6):341–343, 1975.
- 20 Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. In *INFOCOM*, pages 1665–1676, 2005.
- 21 Utku Irmak, Svilen Mihaylov, and Torsten Suel. Improved single-round protocols for remote file synchronization. *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, 3:1665–1676 vol. 3, 2005.
- 22 Thathachar S Jayram and David P Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG)*, 9(3):26, 2013.
- 23 Hossein Jowhari. Efficient communication protocols for deciding edit distance. In *European Symposium on Algorithms*, pages 648–658. Springer, 2012.
- 24 Eamonn J. Keogh. Exact Indexing of Dynamic Time Warping. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 406–417, 2002.
- 25 Eamonn J. Keogh and Michael J. Pazzani. Scaling up Dynamic Time Warping to Massive Dataset. In *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD '99, Prague, Czech Republic, September 15-18, 1999, Proceedings*, pages 1–11, 1999.
- 26 Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 285–289, 2000.
- 27 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 596–605. ACM, 1995.
- 28 Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *arXiv preprint*, 2010. [arXiv:1003.4083](#).
- 29 Mario E Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 108–115. IEEE, 1999.
- 30 Assaf Naor. Probabilistic clustering of high dimensional norms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 690–709. SIAM, 2017.
- 31 Ofer Neiman. On Stochastic Decompositions of Metric Spaces.
- 32 Alon Orlitsky. Interactive communication: Balanced distributions, correlated files, and average-case complexity. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 228–238. IEEE, 1991.
- 33 François Petitjean, Germain Forestier, Geoffrey I. Webb, Ann E. Nicholson, Yanping Chen, and Eamonn J. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowl. Inf. Syst.*, 47(1):1–26, 2016.
- 34 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

- 35 Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. FTW: fast similarity search under the time warping distance. In *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 326–337, 2005.
- 36 Braverman Vladimir, Moses Charikar, William Kuszmaul, David P. Woodruff, and Liu F. Yang. The One-Way Communication Complexity of Dynamic time Warping Distance. *arXiv preprint*, 2019. [arXiv:1903.03520](https://arxiv.org/abs/1903.03520).
- 37 Rex Ying, Jiangwei Pan, Kyle Fox, and Pankaj K Agarwal. A simple efficient approximation algorithm for dynamic time warping. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 21. ACM, 2016.
- 38 Yunyue Zhu and Dennis Shasha. Warping indexes with envelope transforms for query by humming. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 181–192. ACM, 2003.





# Walking the Dog Fast in Practice: Algorithm Engineering of the Fréchet Distance

**Karl Bringmann**

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany  
kbringma@mpi-inf.mpg.de

**Marvin Künnemann**

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany  
marvin@mpi-inf.mpg.de

**André Nusser**

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany  
Saarbrücken Graduate School of Computer Science, Saarland Informatics Campus, Germany  
anusser@mpi-inf.mpg.de

---

## Abstract

The Fréchet distance provides a natural and intuitive measure for the popular task of computing the similarity of two (polygonal) curves. While a simple algorithm computes it in near-quadratic time, a strongly subquadratic algorithm cannot exist unless the Strong Exponential Time Hypothesis fails. Still, fast practical implementations of the Fréchet distance, in particular for realistic input curves, are highly desirable. This has even led to a designated competition, the ACM SIGSPATIAL GIS Cup 2017: Here, the challenge was to implement a near-neighbor data structure under the Fréchet distance. The bottleneck of the top three implementations turned out to be precisely the decision procedure for the Fréchet distance.

In this work, we present a fast, certifying implementation for deciding the Fréchet distance, in order to (1) complement its pessimistic worst-case hardness by an empirical analysis on realistic input data and to (2) improve the state of the art for the GIS Cup challenge. We experimentally evaluate our implementation on a large benchmark consisting of several data sets (including handwritten characters and GPS trajectories). Compared to the winning implementation of the GIS Cup, we obtain running time improvements of up to more than two orders of magnitude for the decision procedure and of up to a factor of 30 for queries to the near-neighbor data structure.

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

**Keywords and phrases** curve simplification, Fréchet distance, algorithm engineering

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.17

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1901.01504>.

**Supplement Material** [https://github.com/chaot4/frechet\\_distance](https://github.com/chaot4/frechet_distance)

## 1 Introduction

A variety of practical applications analyze and process trajectory data coming from different sources like GPS measurements, digitized handwriting, motion capturing, and many more. One elementary task on trajectories is to compare them, for example in the context of signature verification [28], map matching [15, 27, 16, 10], and clustering [12, 13]. In this work we consider the Fréchet distance as curve similarity measure as it is arguably the most natural and popular one. Intuitively, the Fréchet distance between two curves is explained using the following analogy. A person walks a dog, connected by a leash. Both walk along



© Karl Bringmann, Marvin Künnemann, and André Nusser;  
licensed under Creative Commons License CC-BY

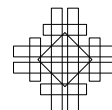
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 17; pp. 17:1–17:21

Leibniz International Proceedings in Informatics



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



their respective curve, with possibly varying speeds and without ever walking backwards. Over all such traversals, we search for the ones which minimize the leash length, i.e., we minimize the maximal distance the dog and the person have during the traversal.

Initially defined more than one hundred years ago [21], the Fréchet distance quickly gained popularity in computer science after the first algorithm to compute it was presented by Alt and Godau [2]. In particular, they showed how to decide whether two length- $n$  curves have Fréchet distance at most  $\delta$  in time  $\mathcal{O}(n^2)$  by full exploration of a quadratic-sized search space, the so-called *free-space* (we refer to Section 3.1 for a definition). Almost twenty years later, it was shown that, conditional on the Strong Exponential Time Hypothesis (SETH), there cannot exist an algorithm with running time  $\mathcal{O}(n^{2-\epsilon})$  for any  $\epsilon > 0$  [6]. Even for realistic models of input curves, such as  $c$ -packed curves [18], exact computation of the Fréchet distance requires time  $n^{2-o(1)}$  under SETH [6]. Only if we relax the goal to finding a  $(1 + \epsilon)$ -approximation of the Fréchet distance, algorithms with near-linear running times in  $n$  and  $c$  on  $c$ -packed curves are known to exist [18, 7].

It is a natural question whether these hardness results are mere theoretical worst-case results or whether computing the Fréchet distance is also hard in practice. This line of research was particularly fostered by the research community in form of the GIS Cup 2017 [25]. In this competition, the 28 contesting teams were challenged to give a fast implementation for the following problem: Given a data set of two-dimensional trajectories  $\mathcal{D}$ , answer queries that ask to return, given a curve  $\pi$  and query distance  $\delta$ , all  $\sigma \in \mathcal{D}$  with Fréchet distance at most  $\delta$  to  $\pi$ . We call this the *near-neighbor problem*.

The three top implementations [5, 11, 19] use multiple layers of heuristic filters and spatial hashing to decide as early as possible whether a curve belongs to the output set or not, and finally use an essentially exhaustive Fréchet distance computation for the remaining cases. Specifically, these implementations perform the following steps:

0. Preprocess  $\mathcal{D}$ .

On receiving a query with curve  $\pi$  and query distance  $\delta$ :

1. Use spatial hashing to identify candidate curves  $\sigma \in \mathcal{D}$ .
2. For each candidate  $\sigma$ , decide whether  $\pi, \sigma$  have Fréchet distance  $\leq \delta$ :
  - a) Use heuristics (*filters*) for a quick resolution in simple cases.
  - b) If unsuccessful, use a *complete decision procedure via free-space exploration*.

Let us highlight the *Fréchet decider* outlined in steps 2a and 2b: Here, *filters* refer to sound, but incomplete Fréchet distance decision procedures, i.e., whenever they succeed to find an answer, they are correct, but they may return that the answer remains unknown. In contrast, a *complete decision procedure via free-space exploration* explores a sufficient part of the free space (the search space) to always determine the correct answer. As it turns out, the bottleneck in all three implementations is precisely Step 2b, the complete decision procedure via free-space exploration. Especially [5] improved upon the naive implementation of the free-space exploration by designing very basic pruning rules, which might be the advantage because of which they won the competition. There are two directions for further substantial improvements over the cup implementations: (1) increasing the range of instances covered by fast filters and (2) algorithmic improvements of the exploration of the reachable free-space.

**Our contribution.** We develop a fast, practical Fréchet distance implementation. To this end, we give a complete decision procedure via free-space exploration that uses a divide-and-conquer interpretation of the Alt-Godau algorithm for the Fréchet distance and optimize it using sophisticated pruning rules. These pruning rules greatly reduce the search space for the realistic benchmark sets we consider – this is surprising given that simple constructions

generate hard instances which require the exploration of essentially the full quadratic-sized search space [6, 8]. Furthermore, we present improved filters that are sufficiently fast compared to the complete decider. Here, the idea is to use adaptive step sizes (combined with useful heuristic tests) to achieve essentially “sublinear” time behavior for testing if an instance can be resolved quickly. Additionally, our implementation is certifying (see [22] for a survey on certifying algorithms), meaning that for every decision of curves being close/far, we provide a short proof (certificate) that can be checked easily; we also implemented a computational check of these certificates. See Section 8 in the full version for details.

An additional contribution of this work is the creation of benchmarks to make future implementations more easily comparable. We compile benchmarks both for the near-neighbor problem (Steps 0 to 2) and for the decision problem (Step 2). For this, we used publicly available curve data and created queries in a way that should be representative for the performance analysis of an implementation. As data sets we use the GIS Cup trajectories [24], a set of handwritten characters called the Character Trajectories Data Set [26] from [17], and the GeoLife data set [3] of Microsoft Research [30, 29, 31]. Our benchmarks cover different distances and also curves of different similarity, giving a broad overview over different settings. We make the source code as well as the benchmarks publicly available to enable independent comparisons with our approach.<sup>1</sup> Additionally, we particularly focus on making our implementation easily readable to enable and encourage others to reuse the code.

**Evaluation.** The GIS Cup 2017 had 28 submissions, with the top three submissions<sup>2</sup> (in decreasing order) due to Bringmann and Baldus [5], Buchin et al. [11], and Dütsch and Vahrenhold [19]. We compare our implementation with all of them by running their implementations on our new benchmark set for the near-neighbor problem and also comparing to the improved decider of [5]. The comparison shows significant speed-ups up to almost a factor of 30 for the near-neighbor problem and up to more than two orders of magnitude for the decider.

**Related work.** The best known algorithm for deciding the Fréchet distance runs in time  $O(n^2 \frac{(\log \log n)^2}{\log n})$  on the word RAM [9]. This relies on the Four Russians technique and is mostly of theoretical interest. There are many variants of the Fréchet distance, e.g., the discrete Fréchet distance [1, 20]. After the GIS Cup 2017, several practical papers studying aspects of the Fréchet distance appeared [4, 14, 23]. Some of this work [4, 14] addressed how to improve upon the spatial hashing step (Step 1) if we relax the requirement of exactness. Since this is orthogonal to our approach of improving the complete decider, these improvements could possibly be combined with our algorithm. The other work [23] neither compared with the GIS Cup implementations, nor provided their source code publicly to allow for a comparison, which is why we have to ignore it here.

<sup>1</sup> Code and benchmarks are available at: [https://github.com/chaot4/frechet\\_distance](https://github.com/chaot4/frechet_distance)

<sup>2</sup> The submissions were evaluated “for their correctness and average performance on a[sic!] various large trajectory databases and queries”. Additional criteria were the following: “We will use the total elapsed wall clock time as a measure of performance. For breaking ties, we will first look into the scalability behavior for more and more queries on larger and larger datasets. Finally, we break ties on code stability, quality, and readability and by using different datasets.”

## 2 Preliminaries

Our implementation as well as the description are restricted to two dimensions, however, the approach can also be generalized to polygonal curves in  $d$  dimensions. Therefore, a *curve*  $\pi$  is defined by its *vertices*  $\pi_1, \dots, \pi_n \in \mathbb{R}^2$  which are connected by straight lines. We also allow continuous indices as follows. For  $p = i + \lambda$  with  $i \in \{1, \dots, n\}$  and  $\lambda \in [0, 1]$ , let

$$\pi_p := (1 - \lambda)\pi_i + \lambda\pi_{i+1}.$$

We call the  $\pi_p$  with  $p \in [1, n]$  the *points* on  $\pi$ . A subcurve of  $\pi$  which starts at point  $p$  on  $\pi$  and ends at point  $q$  on  $\pi$  is denoted by  $\pi_{p\dots q}$ . In the remainder, we denote the *number of vertices* of  $\pi$  (resp.  $\sigma$ ) with  $n$  (resp.  $m$ ) if not stated otherwise. We denote the length of a curve  $\pi$  by  $\|\pi\|$ , i.e., the sum of the Euclidean lengths of its line segments. Additionally, we use  $\|v\|$  for the Euclidean norm of a vector  $v \in \mathbb{R}^2$ . For two curves  $\pi$  and  $\sigma$ , the *Fréchet distance*  $d_F(\pi, \sigma)$  is defined as

$$d_F(\pi, \sigma) := \inf_{\substack{f \in \mathcal{T}_n \\ g \in \mathcal{T}_m}} \max_{t \in [0, 1]} \|\pi_{f(t)} - \sigma_{g(t)}\|,$$

where  $\mathcal{T}_k$  is the set of monotone and continuous functions  $f : [0, 1] \rightarrow [1, k]$ . We define a *traversal* as a pair  $(f, g) \in \mathcal{T}_n \times \mathcal{T}_m$ . Given two curves  $\pi, \sigma$  and a query distance  $\delta$ , we call them *close* if  $d_F(\pi, \sigma) \leq \delta$  and *far* otherwise. There are two problem settings that we consider in this paper:

**Decider Setting.** Given curves  $\pi, \sigma$  and a distance  $\delta$ , decide whether  $d_F(\pi, \sigma) \leq \delta$ . (With such a decider, we can compute the exact distance by using parametric search in theory and binary search in practice.)

**Query Setting.** Given a curve dataset  $\mathcal{D}$ , build a data structure that on query  $(\pi, \delta)$  returns all  $\sigma \in \mathcal{D}$  with  $d_F(\pi, \sigma) \leq \delta$ .

We mainly focus on the decider in this work. To allow for a comparison with previous implementations (which are all in the query setting), we also run experiments with our decider plugged into a data structure for the query setting.

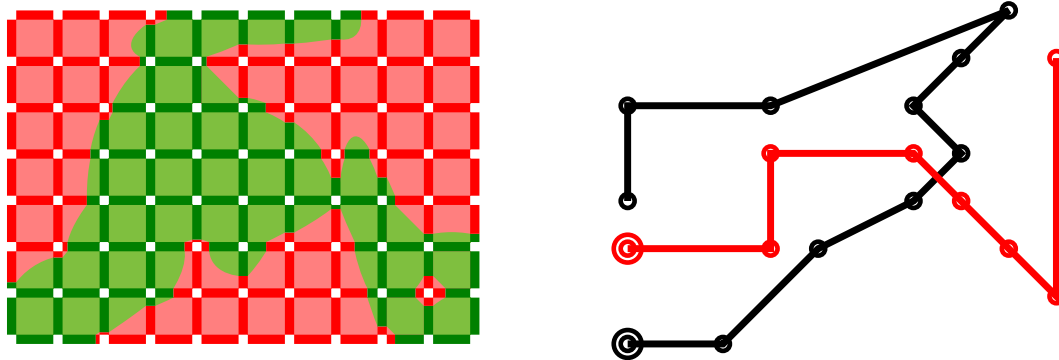
### 2.1 Preprocessing

When reading the input curves we immediately compute additional data which is stored with each curve:

**Prefix Distances.** To be able to quickly compute the curve length between any two vertices of  $\pi$ , we precompute the prefix lengths, i.e., the curve lengths  $\|\pi_{1\dots i}\|$  for every  $i \in \{2, \dots, n\}$ . We can then compute the curve length for two indices  $i < i'$  on  $\pi$  by  $\|\pi_{i\dots i'}\| = \|\pi_{1\dots i'}\| - \|\pi_{1\dots i}\|$ .

**Bounding Box.** We compute the bounding box of all curves, which is a simple coordinate-wise maximum and minimum computation.

Both of these preprocessing steps are extremely cheap as they only require a single pass over all curves, which we anyway do when parsing them. In the remainder of this work we assume that this additional data was already computed, in particular, we do not measure it in our experiments as it is dominated by reading the curves.



■ **Figure 1** Example of a free-space diagram for curves  $\pi$  (black) and  $\sigma$  (red). The doubly-circled vertices mark the start. The free-space, i.e., the pairs of indices of points which are close, is colored green. The non-free areas are colored red. The threshold distance  $\delta$  is roughly the distance between the first vertex of  $\sigma$  and the third vertex of  $\pi$ .

### 3 Complete decider

The key improvement of this work lies in the complete decider via free-space exploration. Here, we use a divide-and-conquer interpretation of the algorithm of Alt and Godau [2] which is similar to [5] where a free-space diagram is built recursively. This interpretation allows us to prune away large parts of the search space by designing powerful *pruning rules* identifying parts of the search space that are irrelevant for determining the correct output. Before describing the details, we formally define the free-space diagram.

#### 3.1 Free-space diagram

The free-space diagram was first defined in [2]. Given two polygonal curves  $\pi$  and  $\sigma$  and a distance  $\delta$ , it is defined as the set of all pairs of indices of points from  $\pi$  and  $\sigma$  that are close to each other, i.e.,

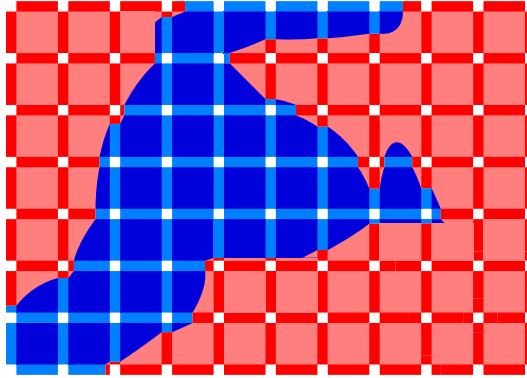
$$F := \{(p, q) \in [1, n] \times [1, m] \mid \|\pi_p - \sigma_q\| \leq \delta\}.$$

For an example see Figure 1. A *path* from  $a$  to  $b$  in the free-space diagram  $F$  is defined as a continuous mapping  $P : [0, 1] \rightarrow F$  with  $P(0) = a$  and  $P(1) = b$ . A path  $P$  in the free-space diagram is *monotone* if  $P(x)$  is component-wise at most  $P(y)$  for any  $0 \leq x \leq y \leq 1$ . The *reachable space* is then defined as

$$R := \{(p, q) \in F \mid \text{there exists a monotone path from } (1, 1) \text{ to } (p, q) \text{ in } F\}.$$

Figure 2 shows the reachable space for the free-space diagram of Figure 1. It is well known that  $d_F(\pi, \sigma) \leq \delta$  if and only if  $(n, m) \in R$ .

This leads us to a simple dynamic programming algorithm to decide whether the Fréchet distance of two curves is at most some threshold distance. We iteratively compute  $R$  starting from  $(1, 1)$  and ending at  $(n, m)$ , using the previously computed values. As  $R$  is potentially a set of infinite size, we have to discretize it. A natural choice is to restrict to cells. The *cell* of  $R$  with coordinates  $(i, j) \in \{1, \dots, n-1\} \times \{1, \dots, m-1\}$  is defined as  $C_{i,j} := R \cap [i, i+1] \times [j, j+1]$ . This is a natural choice as given  $C_{i-1,j}$  and  $C_{i,j-1}$ , we can compute  $C_{i,j}$  in constant time; this follows from the simple fact that  $F \cap [i, i+1] \times [j, j+1]$  is convex [2]. We call this computation of the outputs of a cell the *cell propagation*. This algorithm runs in time  $\mathcal{O}(nm)$  and was introduced by Alt and Godau [2].



■ **Figure 2** Reachable space of the free-space diagram in Figure 1. The reachable part is blue and the non-reachable part is red. Note that the reachable part is a subset of the free-space.

### 3.2 Basic algorithm

For integers  $1 \leq i \leq i' \leq n, 1 \leq j \leq j' \leq m$  we call the set  $B = [i, i'] \times [j, j']$  a *box*. We denote the left/right/bottom/top *boundaries* of  $B$  by  $B_l = \{i\} \times [j, j']$ ,  $B_r = \{i'\} \times [j, j']$ ,  $B_b = [i, i'] \times \{j\}$ ,  $B_t = [i, i'] \times \{j'\}$ . The *left input* of  $B$  is  $B_l^R = B_l \cap R$ , and its *bottom input* is  $B_b^R = B_b \cap R$ . Similarly, the *right/top output* of  $B$  is  $B_r^R = B_r \cap R$ ,  $B_t^R = B_t \cap R$ . A box is a cell if  $i + 1 = i'$  and  $j + 1 = j'$ . We always denote the lower left corner of a box by  $(i, j)$  and the top right by  $(i', j')$ , if not mentioned otherwise.

A recursive variant of the standard free-space decision procedure is as follows: Start with  $B = [1, n] \times [1, m]$ . At any recursive call, if  $B$  is a cell, then determine its outputs from its inputs in constant time, as described by [2]. Otherwise, split  $B$  vertically or horizontally into  $B_1, B_2$  and first compute the outputs of  $B_1$  from the inputs of  $B$  and then compute the outputs of  $B_2$  from the inputs of  $B$  and the outputs of  $B_1$ . In the end, we just have to check  $(n, m) \in R$  to decide whether the curves are close or far. This is a constant-time operation after calculating all outputs.

Now comes the main idea of our approach: we try to avoid recursive splitting by directly computing the outputs for non-cell boxes using certain rules. We call them *pruning rules* as they enable pruning large parts of the recursion tree induced by the divide-and-conquer approach. Our pruning rules are heuristic, meaning that they are not always applicable, however, we show in the experiments that on practical curves they apply very often and therefore massively reduce the number of recursive calls. The detailed pruning rules are described Section 3.3. Using these rules, we change the above recursive algorithm as follows. In any recursive call on box  $B$ , we first try to apply the pruning rules. If this is successful, then we obtained the outputs of  $B$  and we are done with this recursive call. Otherwise, we perform the usual recursive splitting. Corresponding pseudocode is shown in Algorithm 1.

In the remainder of this section, we describe our pruning rules and their effects.

### 3.3 Pruning rules

In this section we introduce the rules that we use to compute outputs of boxes which are above cell-level in certain special cases. Note that we aim at catching special cases which occur often in practice, as we cannot hope for improvements on adversarial instances due to the conditional lower bound of [6]. Therefore, we make no claims whether they are applicable, only that they are sound and fast. In what follows, we call a boundary *empty* if its intersection with  $R$  is  $\emptyset$ .

**Algorithm 1** Recursive Decider of the Fréchet Distance.

---

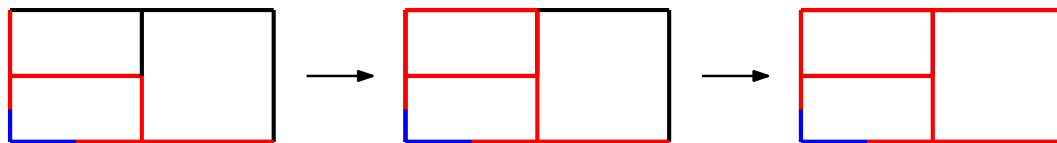
```

1: procedure DECIDEFRÉCHETDISTANCE( $\pi, \sigma$ )
2:   COMPUTEOUTPUTS( $\pi, \sigma, [1, n] \times [1, m]$ )
3:   return  $[(n, m) \in R]$ 

4: procedure COMPUTEOUTPUTS( $\pi, \sigma, B = [i, i'] \times [j, j']$ )
5:   if  $B$  is a cell then
6:     compute outputs by cell propagation
7:   else
8:     use pruning rules I to IV to compute outputs of  $B$ 
9:     if not all outputs have been computed then
10:      if  $j' - j > i' - i$  then ▷ split horizontally
11:         $B_1 = [i, i'] \times [j, \lfloor (j + j')/2 \rfloor]$ 
12:         $B_2 = [i, i'] \times [\lfloor (j + j')/2 \rfloor, j']$ 
13:      else ▷ split vertically
14:         $B_1 = [i, \lfloor (i + i')/2 \rfloor] \times [j, j']$ 
15:         $B_2 = [\lfloor (i + i')/2 \rfloor, i'] \times [j, j']$ 
16:      COMPUTEOUTPUTS( $\pi, \sigma, B_1$ ) and COMPUTEOUTPUTS( $\pi, \sigma, B_2$ )

```

---



**Figure 3** Output computation of a box when inputs are empty. First we can compute the outputs of the top left box and then the outputs of the right box. In this example, we then know that the curves have a Fréchet distance greater than  $\delta$  as  $(n, m)$  is not reachable.

**Rule I: empty inputs**

The simplest case where we can compute the outputs of a box  $B$  is if both inputs are empty, i.e.  $B_b^R = B_l^R = \emptyset$ . In this case no propagation of reachability is possible and thus the outputs are empty as well, i.e.  $B_t^R = B_r^R = \emptyset$ . See Figure 3 for an example.

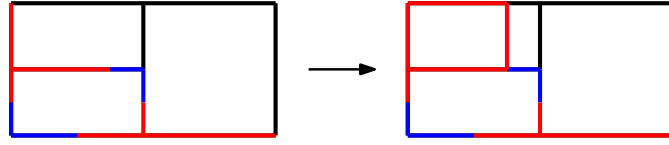
**Rule II: shrink box**

Instead of directly computing the outputs, this rule allows us to shrink the box we are currently working on, which reduces the problem size. Assume that for a box  $B$  we have that  $B_b^R = \emptyset$  and the lowest point of  $B_l^R$  is  $(i, j_{\min})$  with  $j_{\min} > j$ . In this case, no pair in  $[i, i'] \times [j, j_{\min}]$  is reachable. Thus, we can shrink the box to the coordinates  $[i, i'] \times [\lfloor j_{\min} \rfloor, j']$  without losing any reachability information. An equivalent rule can be applied if we swap the role of  $B_b$  and  $B_l$ . See Figure 4 for an example of applying this rule.

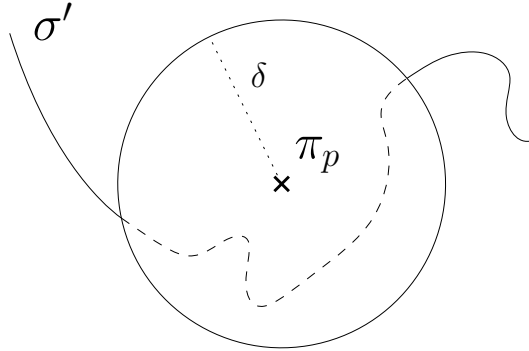
**Rule III: simple boundaries**

*Simple boundaries* are boundaries of a box that contain at most one free section. To define this formally, a set  $\mathcal{I} \subseteq [1, n] \times [1, m]$  is called an *interval* if  $\mathcal{I} = \emptyset$  or  $\mathcal{I} = \{p\} \times [q, q']$  or  $\mathcal{I} = [q, q'] \times \{p\}$  for real  $p$  and an interval  $[q, q']$ . In particular, the four boundaries of a box  $B = [i, i'] \times [j, j']$  are intervals. We say that an interval  $\mathcal{I}$  is *simple* if  $\mathcal{I} \cap F$  is again an interval. Geometrically, we have a free interval of a point  $\pi_p$  and a curve  $\sigma_{q\dots q'}$  (which is the





■ **Figure 4** This is an example of shrinking a box in case one of the inputs is empty and the other one starts with an empty part. In this example the top left box has an empty input on the left and the start of the bottom input is empty as well. Thus, we can shrink the box to the right part.



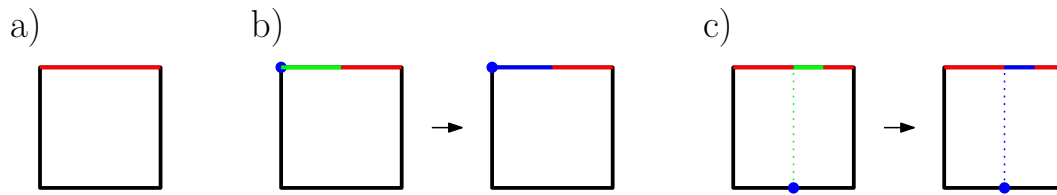
■ **Figure 5** Example of a point  $\pi_p$  and a curve  $\sigma'$  which lead to a simple boundary.

form of a boundary in the free-space diagram) if the circle of radius  $\delta$  around  $\pi_p$  intersects  $\sigma_{q\dots q'}$  at most twice. See Figure 5 for an example. We call such a boundary simple because it is of low complexity, which we can exploit for pruning.

There are three pruning rules that we do based on simple boundaries (see Figure 6 for visualizations). They are stated here for the top boundary  $B_t$ , but symmetric rules apply to  $B_r$ . Later, in Section 3.4, we then explain how to actually compute simple boundaries, i.e., also how to compute  $B_t \cap F$ . The pruning rules are:

- (a) If  $B_t$  is simple because  $B_t \cap F$  is empty then we also know that the output of this boundary is empty. Thus, we are done with  $B_t$ .
- (b) Suppose that  $B_t$  is simple and, more specifically, of the form that it first has a free and then a non-free part; in other words, we have  $(i, j') \in B_t \cap F$ . Due to our recursive approach, we already computed the left inputs of the box and thus know whether the top left corner of the box is reachable, i.e. whether  $(i, j') \in R$ . If this is the case, then we also know the reachable part of our simple boundary: Since  $(i, j') \in R$  and  $B_t \cap F$  is an interval containing  $(i, j')$ , we conclude that  $B_t^R = B_t \cap F$  and we are done with  $B_t$ .
- (c) Suppose that  $B_t$  is simple, but the leftmost point  $(i_{\min}, j')$  of  $B_t \cap F$  has  $i_{\min} > i$ . In this case, we try to certify that  $(i_{\min}, j') \in R$ , because then it follows that  $B_t^R = B_t \cap F$  and we are done with  $B_t$ . To check for reachability of  $(i_{\min}, j')$ , we try to propagate the reachability through the inside of the box, which in this case means to propagate it from the bottom boundary. We test whether  $(i_{\min}, j)$  is in the input, i.e., if  $(i_{\min}, j) \in B_b^R$ , and whether  $\{i_{\min}\} \times [j, j'] \subseteq F$  (by slightly modifying the algorithm for simple boundary computations). If this is the case, then we can reach every point in  $B_t \cap F$  from  $(i_{\min}, j)$  via  $\{i_{\min}\} \times [j, j']$ .

We also use symmetric rules by swapping “top” with “right” and “bottom” with “left”.



**Figure 6** Visualization of the rules for computing outputs using simple boundaries. All three cases are visualized with the top boundary being simple. In a) the boundary is non-free and therefore no point on it can be reachable. In b) the boundary's beginning is free and reachable, enabling us to propagate the reachability to the entire free interval. In c) we can propagate the reachability of a point on the bottom boundary, using a free interval inside the box, to the beginning of the free interval of the top boundary and thus decide the entire boundary.

### Rule IV: boxes at free-space diagram boundaries

The boundaries of a free-space diagram are a special form of boundary which allows us to introduce an additional rule. Consider a box  $B$  which touches the top boundary of the free-space diagram, i.e.,  $B = [i, i'] \times [j, m]$ . Suppose the previous rules allowed us to determine the output for  $B_r^R$ . Since any valid traversal from  $(1, 1)$  to  $(n, m)$  passing through  $B$  intersects  $B_r$ , the output  $B_l^R$  is not needed anymore, and we are done with  $B$ . A symmetric rule applies to boxes which touch the right boundary of the free-space diagram.

### 3.4 Implementation details of simple boundaries

It remains to describe how we test whether a boundary is simple, and how we determine the free interval of a simple boundary. One important ingredient for the fast detection of simple boundaries are two simple heuristic checks that check whether two polygonal curves are close or far, respectively. The former check was already used in [5]. We first explain these heuristic checks, and then explain how to use them for the detection of simple boundaries.

**Heuristic check whether two curves are close.** Given two subcurves  $\pi' := \pi_{i\dots i'}$  and  $\sigma' := \sigma_{j\dots j'}$ , this filter heuristically tests whether  $d_F(\pi', \sigma') \leq \delta$ . Let  $i_c := \lfloor \frac{i+i'}{2} \rfloor$  and  $j_c := \lfloor \frac{j+j'}{2} \rfloor$  be the indices of the midpoints of  $\pi'$  and  $\sigma'$  (with respect to hops). Then  $d_F(\pi', \sigma') \leq \delta$  holds if

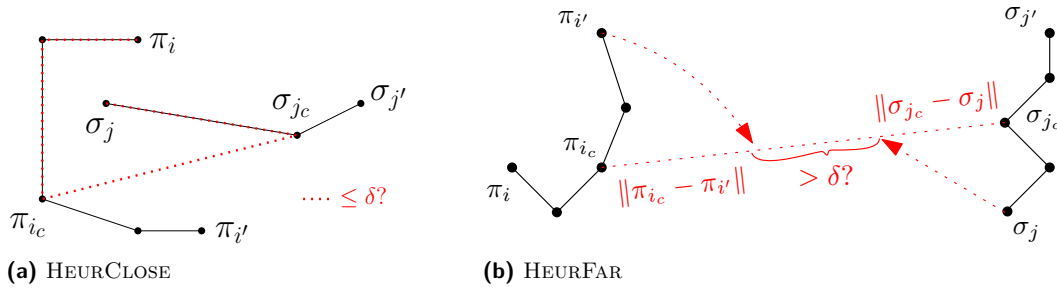
$$\max\{\|\pi_{i\dots i_c}\|, \|\pi_{i_c\dots i'}\|\} + \|\pi_{i_c} - \sigma_{j_c}\| + \max\{\|\sigma_{j\dots j_c}\|, \|\sigma_{j_c\dots j'}\|\} \leq \delta.$$

The triangle equality ensures that this is an upper bound on all distances between two points on the curves. For a visualization, see Figure 7a. Observe that all curve lengths that need to be computed in the above equation can be determined quickly due to our preprocessing, see Section 2.1. We call this procedure  $\text{HEURCLOSE}(\pi', \sigma', \delta)$ .

**Heuristic check whether two curves are far.** Symmetrically, we can test whether all pairs of points on  $\pi'$  and  $\sigma'$  are far by testing

$$\|\pi_{i_c} - \sigma_{j_c}\| - \max\{\|\pi_{i\dots i_c}\|, \|\pi_{i_c\dots i'}\|\} - \max\{\|\sigma_{j\dots j_c}\|, \|\sigma_{j_c\dots j'}\|\} > \delta.$$

We call this procedure  $\text{HEURFAR}(\pi', \sigma', \delta)$ .



■ **Figure 7** Visualizations of heuristic checks HEURCLOSE and HEURFAR.

**Computation of simple boundaries.** Recall that an interval is defined as  $I = \{p\} \times [q, q']$  (intervals of the form  $[q, q'] \times \{p\}$  are handled symmetrically). The naive way to decide whether interval  $I$  is simple would be to go over all the segments of  $\sigma_{q\dots q'}$  and compute the intersection with the circle of radius  $\delta$  around  $\pi_p$ . However, this is too expensive because (i) computing the intersection of a disc and a segment involves taking a square root, which is an expensive operation with a large constant running time, and (ii) iterating over all segments of  $\sigma_{q\dots q'}$  incurs a linear factor in  $n$  for large boxes, while we aim at a logarithmic dependence on  $n$  for simple boundary detection.

We avoid these issues by resolving long subcurves  $\sigma_{j..j+s}$  using our heuristic checks (HEURCLOSE, HEURFAR). Here,  $s$  is an adaptive step size that grows whenever the heuristic checks were applicable, and shrinks otherwise. See Algorithm 2 for pseudocode of our simple boundary detection. It is straightforward to extend this algorithm to not only detect whether a boundary is simple, but also compute the free interval of a simple boundary; we call the resulting procedure SIMPLEBOUNDARY.

### 3.5 Effects of combined pruning rules

All the pruning rules presented above can in practice lead to a reduction of the number of boxes that are necessary to decide the Fréchet distance of two curves. We exemplify this on two real-world curves; see Figure 8 for the curves and their corresponding free-space diagram. We explain in the following where the single rules come into play. For *Box 1* we apply Rule IIIb twice – for the top and right output. The top boundary of *Box 2* is empty and thus we computed the outputs according to Rule IIIa. Note that the right boundary of this box is on the right boundary of the free-space diagram and thus we do not have to compute it according to Rule IV. For *Box 3* we again use Rule IIIb for the top, but we use Rule IIIc for the right boundary – the blue dotted line indicates that the reachability information is propagated through the box. For *Box 4* we first use Rule II to move the bottom boundary significantly up, until the end of the left empty part; we can do this because the bottom boundary is empty and the left boundary is simple, starting with an empty part. After two splits of the remaining box, we see that the two outputs of the leftmost box are empty as the top and right boundaries are non-free, using Rule IIIa. For the remaining two boxes we use Rule I as their inputs are empty.

This example illustrates how propagating through a box (in *Box 3*) and subsequently moving a boundary (in *Box 4*) leads to pruning large parts. Additionally, we can see how using simple boundaries leads to early decisions and thus avoids many recursive steps. In total, we can see how all the explained pruning rules together lead to a free-space diagram

---

**Algorithm 2** Checks if the boundary  $\{p\} \times [q, q']$  in the free-space diagram is simple.

---

```

1: procedure ISIMPLEBOUNDARY( $\pi_p, \sigma_{q\dots q'}$ )
2:   if HEURFAR( $\pi_p, \sigma_{q\dots q'}, \delta$ ) or HEURCLOSE( $\pi_p, \sigma_{q\dots q'}, \delta$ ) then
3:     return ‘simple’

4:    $C \leftarrow \begin{cases} \{\sigma_q\} & , \text{if } \|p - \sigma_q\| \leq \delta \\ \emptyset & , \text{otherwise} \end{cases}$  ▷ set of change points
5:    $s \leftarrow 1, j \leftarrow q$ 
6:   while  $j < q'$  do
7:     if HEURCLOSE( $\pi_p, \sigma_{j\dots j+s}, \delta$ ) then
8:        $j \leftarrow j + s$ 
9:        $s \leftarrow 2s$ 
10:    else if HEURFAR( $\pi_p, \sigma_{j\dots j+s}, \delta$ ) then
11:       $j \leftarrow j + s$ 
12:       $s \leftarrow 2s$ 
13:    else if  $s > 1$  then
14:       $s \leftarrow s/2$ 
15:    else
16:       $P \leftarrow \{j' \in (j, j + 1] \mid \|\pi_p - \sigma_{j'}\| = \delta\}$ 
17:       $C \leftarrow C \cup P$ 
18:       $j \leftarrow j + 1$ 
19:      if  $|C| > 2$  then
20:        return ‘not simple’

21:   return ‘simple’

```

---

with only twelve boxes, i.e., twelve recursive calls, for curves with more than 50 vertices and more than 1500 reachable cells. Figure 9 shows what effects the pruning rules have by introducing them one by one in an example.

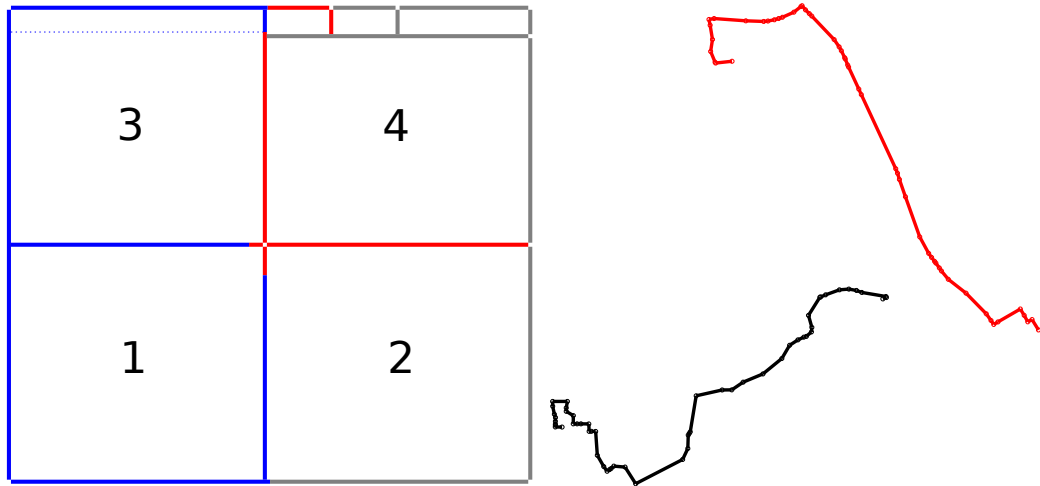
#### 4 Decider with filters

The decider can be divided into two parts:

1. Filters
2. Complete decider via free-space exploration

As outlined in Section 1, we first try to determine the correct output by using fast but incomplete filtering mechanisms and only resort to the slower complete decider presented in the last section if none of the heuristic deciders (*filters*) gave a result.

The speed-ups introduced by our complete decider were already explained in Section 3. A second source for our speed-ups lies in the usage of a good set of filters. Interestingly, since our optimized complete decider via free-space exploration already solves many simple instances very efficiently, our filters have to be extremely fast to be useful – otherwise, the additional effort for an incomplete filter does not pay off. In particular, we cannot afford expensive preprocessing and ideally, we would like to achieve sublinear running times for our filters. To this end, we only use filters that can traverse large parts of the curves quickly. We achieve sublinear-type behavior by making previously used filters work with an adaptive step size (exploiting fast heuristic checks), and designing a new adaptive negative filter. Due to space constraints, the detailed explanation of the filters is in Section 4 of the full version.



■ **Figure 8** A free-space diagram as produced by our final implementation (left) with the corresponding curves (right). The curves are taken from the SIGSPATIAL dataset. We number the boxes in the third level of the recursion from 1 to 4.

## 5 Experiments

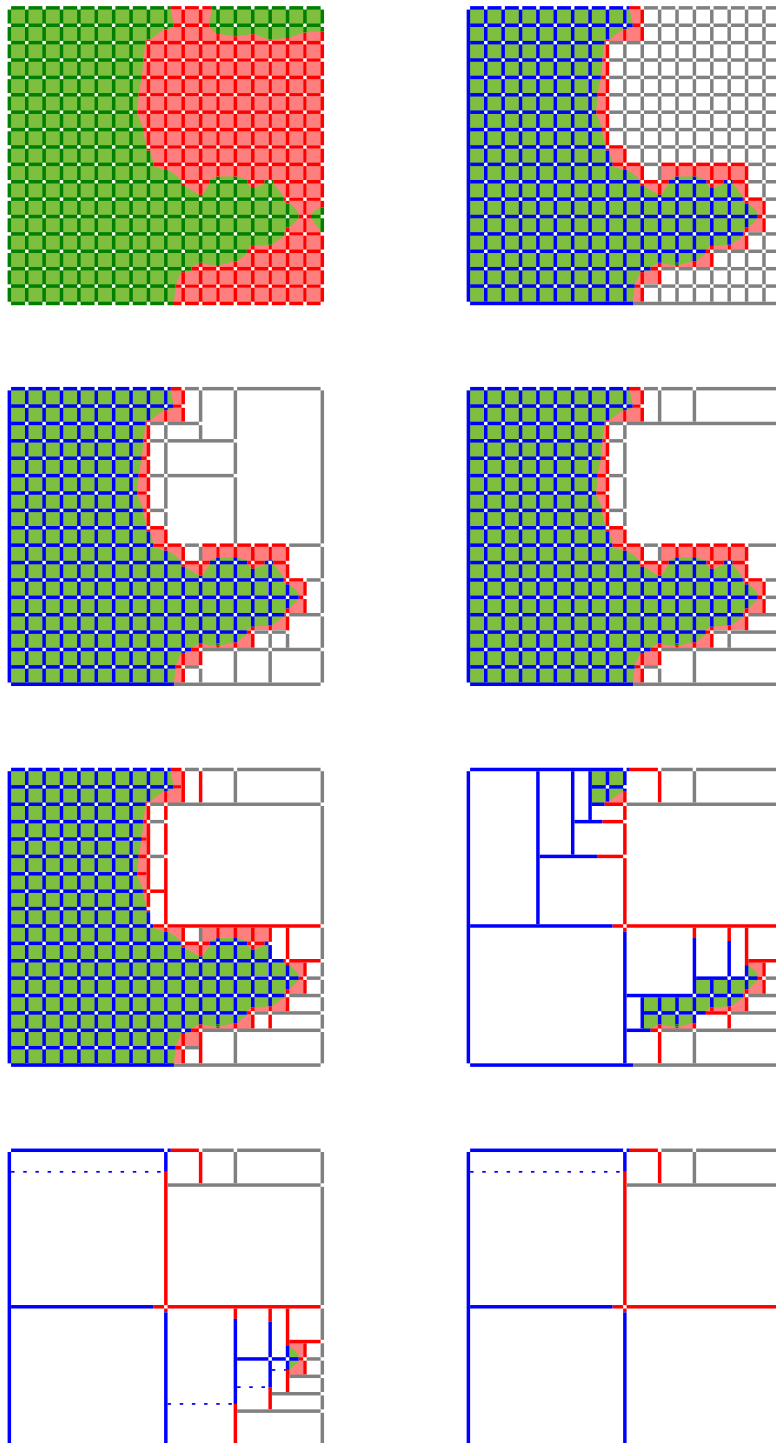
In the experiments, we aim to substantiate the following two claims. First, we want to verify that our main contribution, the decider, actually is a significant improvement over the state of the art. To this end, we compare our implementation with the – to our knowledge – currently fastest Fréchet distance decider, namely [5]. Second, we want to verify that our improvements in the decider setting also carry over to the query setting, also significantly improving the state of the art. To show this, we compare to the top three submissions of the GIS Cup.

We use three different data sets: the GIS Cup set (SIGSPATIAL) [24], the handwritten characters (CHARACTERS) [26], and the GeoLife data set (GEO LIFE) [3]. For all experiments, we used a laptop with an Intel i5-6440HQ processor with 4 cores and 16GB of RAM. See Section 7 of the full version for additional experiments.

### 5.1 Decider setting

In this section we test the running time performance of our new decider algorithm. We first describe our new benchmark using the three data sets, and then discuss our experimental findings, in particular how the performance and improvement over the state of the art varies with the distance and also the “neighbor rank” in the data set.

**Benchmark.** For the decider, we want to specifically test how the decision distance  $\delta$  and how the choice of the second curve  $\sigma$  influences the running time of the decider. To experimentally evaluate this, we create a benchmark for each data set  $\mathcal{D}$  in the following way. We select a random curve  $\pi \in \mathcal{D}$  and sort the curves in the data set  $\mathcal{D}$  by their distance to  $\pi$  in increasing order, obtaining the sequence  $\sigma_1, \dots, \sigma_n$ . For all  $k \in \{1, \dots, \lfloor \log n \rfloor\}$ , we



■ **Figure 9** A decider example introducing the pruning rules one by one. They are introduced from top to bottom and left to right. The images in this order depict: the free-space diagram, the reachable space, after introducing Rule I, Rule II, Rule IIIa, Rule IIIb, Rule IIIc, Rule IV, and finally the free-space diagram with all pruning rules enabled. The curves of this example are shown in Figure 8.

- select a curve  $\sigma \in \{\sigma_{2^k}, \dots, \sigma_{2^{k+1}-1}\}$  uniformly at random<sup>3</sup>,
  - compute the exact distance  $\delta^* := d_F(\pi, \sigma)$ ,
  - for each  $l \in \{-10, \dots, 0\}$ , add benchmark tests  $(\pi, \sigma, (1 - 2^l) \cdot \delta^*)$  and  $(\pi, \sigma, (1 + 2^l) \cdot \delta^*)$ .
- By repeating this process for 1000 uniformly random curves  $\pi \in \mathcal{D}$ , we create 1000 test cases for every pair of  $k$  and  $l$ .

**Running times.** First we show how our implementation performs in this benchmark. In Figure 10 we depict timings for running our implementation on the benchmark for all data sets. We can see that distances larger than the exact Fréchet distance are harder than smaller distances. This effect is most likely caused by the fact that decider instances with positive result need to find a path through the free-space diagram, while negative instances might be resolved earlier as it already becomes clear close to the lower left corner of the free-space diagram that there cannot exist such a path. Also, the performance of the decider is worse for computations on  $(\pi, \sigma_i, \delta)$  when  $i$  is smaller. This seems natural, as curves which are closer are more likely in the data set to actually be of similar shape, and similar shapes often lead to bottlenecks in the free-space diagram (i.e., small regions where a witness path can barely pass through), which have to be resolved in much more detail and therefore lead to a higher number of recursive calls. It follows that the benchmark instances for low  $k$  and  $l$  are the hardest; this is the case for all data sets. In CHARACTERS we can also see that for  $k = 7$  there is suddenly a rise in the running time for certain distance factors. We assume that this comes from the fact that the previous values of  $k$  all correspond to the same written character and this changes for  $k = 7$ .

We also run the original code of the winner of the GIS Cup, namely [5], on our benchmark and compare it with the running time of our implementation. See Figure 11 for the speed-up factors of our implementation over the GIS Cup winner implementation. The speed-ups obtained depend on the data set. While for every data set a significant amount of benchmarks for different  $k$  and  $l$  are more than one order of magnitude faster, for GEOLIFE even speed-ups by 2 orders of magnitude are reached. Speed-ups tend to be higher for larger distance factors. The results on GEOLIFE suggest that for longer curves, our implementation becomes significantly faster relative to the current state of the art. Note that there also are situations where our decider shows similar performance to the one of [5]; however, those are cases where both deciders can easily recognize that the curves are far (due to, e.g., their start or end points being far). We additionally show the percentage of instances that are already decided by the filters in Figure 12.

## 5.2 Query setting

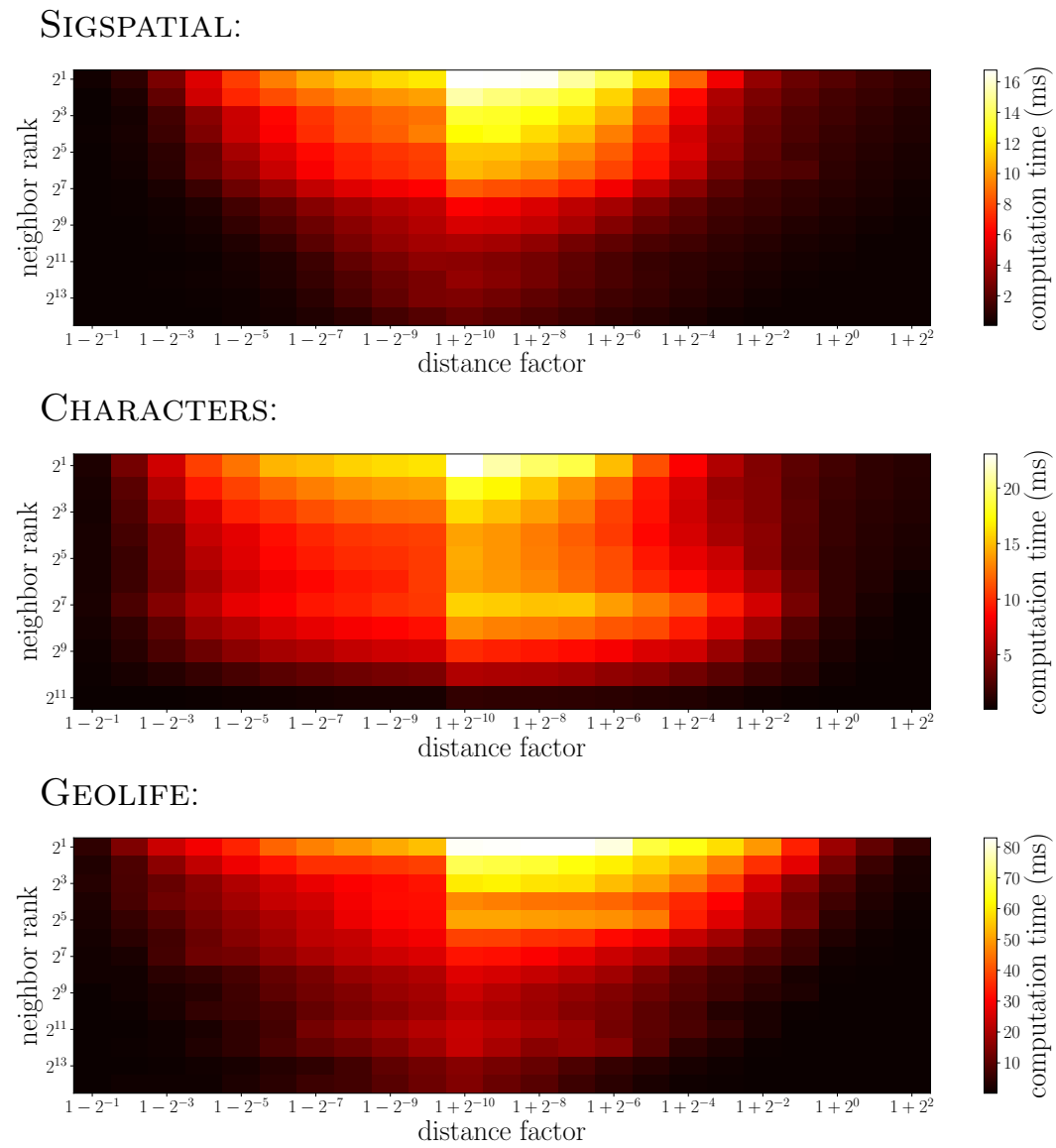
Our query data structure is influenced by [5]. We first use spatial hashing (via a kd-tree) on endpoints and extrema of the curves to select a good set of candidate curves, and then use our decider from Section 4 to decide those candidates. See Section 5 of the full version.

**Benchmark.** We build a query benchmark similar to the one used in [5]. For each  $k \in \{0, 1, 10, 100, 1000\}$ , we select a random curve  $\pi \in \mathcal{D}$  and then pick a threshold distance  $\delta$  such that a query of the form  $(\pi, \delta)$  returns exactly  $k + 1$  curves (note that the curve  $\pi$  itself is also always returned). We repeat this 1000 times for each value of  $k$  and also create such a benchmark for each of the three data sets.

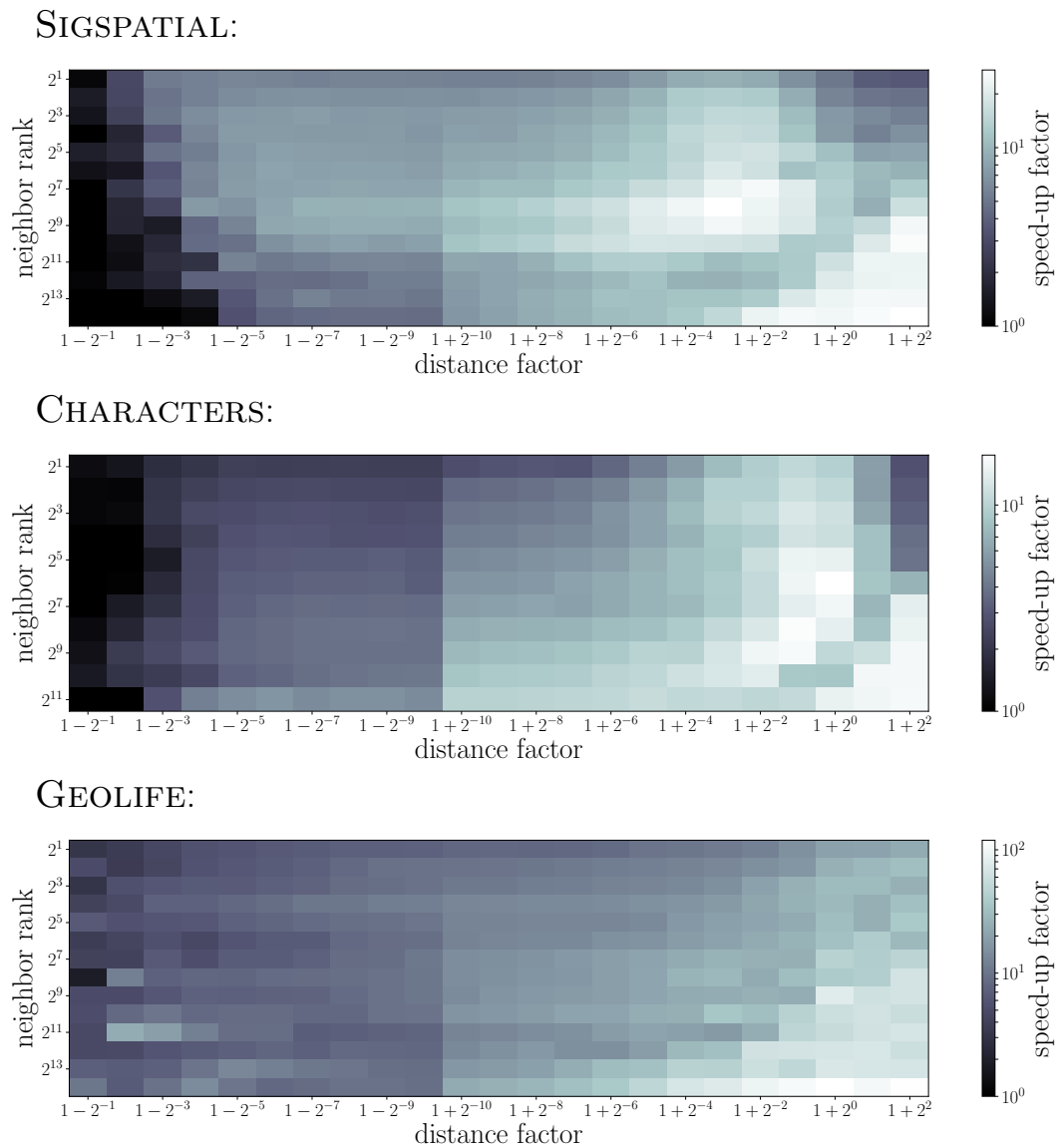
---

<sup>3</sup> Note that for  $k = \lfloor \log n \rfloor$  some curves might be undefined as possibly  $2^{k+1} - 1 > n$ . In this case we select a curve uniformly at random from  $\{\sigma_{2^k}, \dots, \sigma_n\}$ .

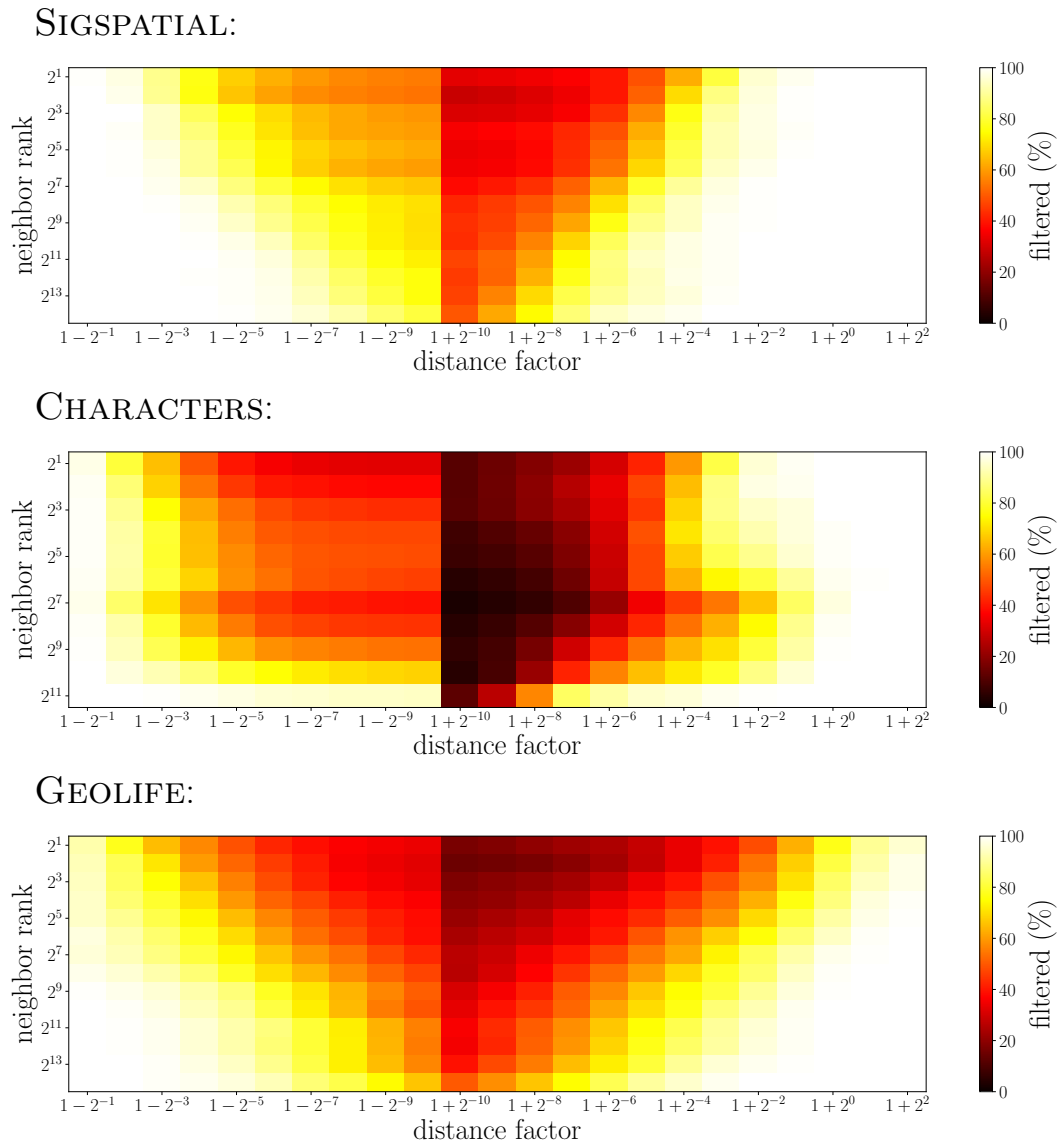




■ **Figure 10** Running times of the decider benchmark when we run our implementation on it.



■ **Figure 11** The speed-up factors obtained over the GIS Cup winner on the decider benchmark.



■ **Figure 12** The percentage of queries that are decided by the filters on the decider benchmark.



**Running times.** We compare our implementation with the top three implementations of the GIS Cup on this benchmark. The results are shown in Table 1. Again the running time improvement of our implementation depends on the data set. For CHARACTERS the maximal improvement factor over the second best implementation is 14.6, for SIGSPATIAL 17.3, and for GEOLIFE 29.1. For SIGSPATIAL and CHARACTERS it is attained at  $k = 1000$ , while for GEOLIFE it is reached at  $k = 100$  but  $k = 1000$  shows a very similar but slightly smaller factor.

To give deeper insights about the single parts of our decider, a detailed analysis of the running times of the single parts of the algorithm is shown in Table 2. Again we witness different behavior depending on the data set. It is remarkable that for SIGSPATIAL the running time for  $k = 1000$  is dominated by the greedy filter. This suggests that improving the filters might still lead to a significant speed-up in this case. However, for most of the remaining cases the running time is clearly dominated by the complete decider, suggesting that our efforts of improving the state of the art focused on the right part of the algorithm.

---

## References

- 1 Pankaj K. Agarwal, Rinat Ben Avraham, Haim Kaplan, and Micha Sharir. Computing the Discrete Fréchet Distance in Subquadratic Time. *SIAM J. Comput.*, 43(2):429–449, 2014. doi:10.1137/130920526.
- 2 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.
- 3 Microsoft Research Asia. GeoLife GPS Trajectories. <https://www.microsoft.com/en-us/download/details.aspx?id=52367>. Accessed: 2018-12-03.
- 4 Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proc. 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS)*, 2018.
- 5 Julian Baldus and Karl Bringmann. A Fast Implementation of Near Neighbors Queries for Fréchet Distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'17, pages 99:1–99:4, New York, NY, USA, 2017. ACM. doi:10.1145/3139958.3140062.
- 6 Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 661–670. IEEE, 2014.
- 7 Karl Bringmann and Marvin Künnemann. Improved Approximation for Fréchet Distance on c-Packed Curves Matching Conditional Lower Bounds. *Int. J. Comput. Geometry Appl.*, 27(1-2):85–120, 2017. doi:10.1142/S0218195917600056.
- 8 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *Journal of Computational Geometry*, 7(2):46–76, December 2015. doi:10.20382/jocg.v7i2a4.
- 9 Kevin Buchin, Maïke Buchin, Wouter Meulemans, and Wolfgang Mulzer. Four Soviets Walk the Dog: Improved Bounds for Computing the Fréchet Distance. *Discrete & Computational Geometry*, 58(1):180–216, 2017. doi:10.1007/s00454-017-9878-7.
- 10 Kevin Buchin, Maïke Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 645–654. Society for Industrial and Applied Mathematics, 2009.
- 11 Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. Efficient Trajectory Queries Under the Fréchet Distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'17, pages 101:1–101:4, New York, NY, USA, 2017. ACM. doi:10.1145/3139958.3140064.

- 12 Kevin Buchin, Anne Driemel, Joachim Gudmundsson, Michael Horton, Irina Kostitsyna, and Maarten Löffler. Approximating  $(k, \ell)$ -center clustering for curves. *CoRR*, abs/1805.01547, 2018. [arXiv:1805.01547](https://arxiv.org/abs/1805.01547).
- 13 Jonathan Campbell, Jonathan Tremblay, and Clark Verbrugge. Clustering Player Paths. In *FDG*, 2015.
- 14 Matteo Ceccarello, Anne Driemel, and Francesco Silvestri. FRESH: Fréchet similarity with hashing. *CoRR*, abs/1809.02350, 2018. [arXiv:1809.02350](https://arxiv.org/abs/1809.02350).
- 15 Erin Chambers, Brittany Terese Fasy, Yusu Wang, and Carola Wenk. Map-matching using shortest paths. In *Proceedings of the 3rd International Workshop on Interactive and Spatial Computing*, pages 44–51. ACM, 2018.
- 16 Daniel Chen, Anne Driemel, Leonidas J Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the Fréchet distance. In *2011 Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 75–83. SIAM, 2011.
- 17 Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL: <http://archive.ics.uci.edu/ml>.
- 18 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet Distance for Realistic Curves in Near Linear Time. *Discrete & Computational Geometry*, 48(1):94–127, July 2012. doi:10.1007/s00454-012-9402-z.
- 19 Fabian Dütsch and Jan Vahrenhold. A Filter-and-Refinement-Algorithm for Range Queries Based on the Fréchet Distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL'17*, pages 100:1–100:4, New York, NY, USA, 2017. ACM. doi:10.1145/3139958.3140063.
- 20 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.
- 21 M Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22(1):1–72, 1906.
- 22 Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.
- 23 Hong Wei, Riccardo Fellegara, Yin Wang, Leila De Florian, and Hanan Samet. Multi-level Filtering to Retrieve Similar Trajectories Under the Fréchet Distance. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '18*, pages 600–603, New York, NY, USA, 2018. ACM. doi:10.1145/3274895.3274978.
- 24 Martin Werner. ACM SIGSPATIAL GIS Cup 2017 Data Set. <https://www.martinwerner.de/datasets/san-francisco-shortest-path.html>. Accessed: 2018-12-03.
- 25 Martin Werner and Dev Oliver. ACM SIGSPATIAL GIS Cup 2017 - Range Queries Under Fréchet Distance. *ACM SIGSPATIAL Newsletter, To Appear.*, 2018.
- 26 Ben H Williams. Character Trajectories Data Set. <https://archive.ics.uci.edu/ml/datasets/Character+Trajectories>. Accessed: 2018-12-03.
- 27 Tim Wylie and Binhai Zhu. Intermittent Map Matching with the Discrete Fréchet Distance. *arXiv preprint*, 2014. [arXiv:1409.2456](https://arxiv.org/abs/1409.2456).
- 28 Jianbin Zheng, Xiaolei Gao, Enqi Zhan, and Zhangcan Huang. Algorithm of On-Line Handwriting Signature Verification Based on Discrete Fréchet Distance. In Lishan Kang, Zhihua Cai, Xuesong Yan, and Yong Liu, editors, *Advances in Computation and Intelligence*, pages 461–469, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
- 29 Yu Zheng, Xing Xie, and Wei-Ying Ma. Understanding Mobility Based on GPS Data. In *Proceedings of the 10th ACM conference on Ubiquitous Computing (UbiComp 2008)*, September 2008. URL: <https://www.microsoft.com/en-us/research/publication/understanding-mobility-based-on-gps-data/>.

- 30 Yu Zheng, Xing Xie, and Wei-Ying Ma. Mining Interesting Locations and Travel Sequences From GPS Trajectories. In *Proceedings of International conference on World Wide Web 2009*, April 2009. WWW 2009. URL: <https://www.microsoft.com/en-us/research/publication/mining-interesting-locations-and-travel-sequences-from-gps-trajectories/>.
- 31 Yu Zheng, Xing Xie, and Wei-Ying Ma. GeoLife: A Collaborative Social Networking Service among User, location and trajectory. *IEEE Data(base) Engineering Bulletin*, June 2010.





# Polyline Simplification has Cubic Complexity

**Karl Bringmann**

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany  
kbringma@mpi-inf.mpg.de

**Bhaskar Ray Chaudhury**

Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany  
Graduate School of Computer Science Saarbrücken, Saarland Informatics Campus, Germany  
braycha@mpi-inf.mpg.de

---

## Abstract

In the classic polyline simplification problem we want to replace a given polygonal curve  $P$ , consisting of  $n$  vertices, by a subsequence  $P'$  of  $k$  vertices from  $P$  such that the polygonal curves  $P$  and  $P'$  are “close”. Closeness is usually measured using the Hausdorff or Fréchet distance. These distance measures can be applied *globally*, i.e., to the whole curves  $P$  and  $P'$ , or *locally*, i.e., to each simplified subcurve and the line segment that it was replaced with separately (and then taking the maximum). We provide an  $\mathcal{O}(n^3)$  time algorithm for simplification under Global-Fréchet distance, improving the previous best algorithm by a factor of  $\Omega(kn^2)$ . We also provide evidence that in high dimensions cubic time is essentially optimal for all three problems (Local-Hausdorff, Local-Fréchet, and Global-Fréchet). Specifically, improving the cubic time to  $\mathcal{O}(n^{3-\epsilon}\text{poly}(d))$  for polyline simplification over  $(\mathbb{R}^d, L_p)$  for  $p = 1$  would violate plausible conjectures. We obtain similar results for all  $p \in [1, \infty), p \neq 2$ . In total, in high dimensions and over general  $L_p$ -norms we resolve the complexity of polyline simplification with respect to Local-Hausdorff, Local-Fréchet, and Global-Fréchet, by providing new algorithms and conditional lower bounds.

**2012 ACM Subject Classification** Theory of computation; Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Mathematical optimization

**Keywords and phrases** Polyline simplification, Fréchet distance, Hausdorff distance, Conditional lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.18

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1810.00621>.

## 1 Introduction

We revisit the classic problem of polygonal line simplification, which is fundamental to computational geometry. The most frequently implemented algorithms for curve simplification go back to the 70s (Douglas and Peucker [11]) and 80s (Imai and Iri [18]). A *polyline* is given by a sequence  $P = \langle v_0, v_1, \dots, v_n \rangle$  of points  $v_i \in \mathbb{R}^d$ , and represents the continuous curve walking along the line segments  $\overline{v_i v_{i+1}}$  in order. Given such a polyline  $P$  and a number  $\delta > 0$ , we want to compute  $P' = \langle v_{i_0}, \dots, v_{i_{k-1}} \rangle$ , with  $0 = i_0 < \dots < i_{k-1} = n$ , of minimal length  $k$  such that  $P$  and  $P'$  have “distance” at most  $\delta$ .

Several distance measures have been used for the curve simplification problem. The most generic distance measure on *point sets*  $A, B$  is the *Hausdorff distance*. However, the most popular distance measure for curves in computational geometry is the *Fréchet distance*  $\delta_F$ . In comparison to Hausdorff distance, it takes the ordering of the vertices along the curves into account, and thus better captures an intuitive notion of distance among curves.

For both of these distance measures  $\delta_* \in \{\delta_H, \delta_F\}$ , we can apply them *locally* or *globally* to measure the distance between the original curve  $P$  and its simplification  $P'$ . In the global variant, we consider the distance  $\delta_*(P, P')$ , i.e., we use the Hausdorff or Fréchet distance of  $P$  and  $P'$ . In the local variant, we consider the distance  $\max_{1 \leq \ell < k} \delta_*(P[i_{\ell-1} \dots i_\ell], \overline{v_{i_{\ell-1}} v_{i_\ell}})$ ,



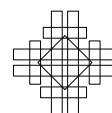
© Karl Bringmann and Bhaskar Ray Chaudhury;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 18; pp. 18:1–18:16

Leibniz International Proceedings in Informatics



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



i.e., for each simplified subcurve  $P[i_{\ell-1} \dots i_{\ell}]$  of  $P$  we compute the distance to the line segment  $\overline{v_{i_{\ell-1}} v_{i_{\ell}}}$  that we simplified the subcurve to, and we take the maximum over these distances. This gives rise to four problem variants, depending on the distance measure: Local-Hausdorff, Local-Fréchet, Global-Hausdorff, and Global-Fréchet.

Among the variants, Global-Hausdorff is unreasonable as it does not take the ordering of vertices of the curve into account and it was recently shown that curve simplification under Global-Hausdorff is NP-hard [20]. Hence, we do not consider this measure in this paper.

The classic  $\hat{O}(n^3)$  time algorithm by Imai and Iri [18] was designed for Local-Hausdorff simplification. By changing the distance computation in this algorithm for the Fréchet distance, one can obtain an  $\hat{O}(n^3)$ -time algorithm for Local-Fréchet [15]. There are improvements for Local-Hausdorff simplification in small dimension  $d$  [19, 8, 6]; the fastest running times are  $2^{O(d)}n^2$  for  $L_1$ -norm,  $\hat{O}(n^2)$  for  $L_\infty$ -norm, and  $\hat{O}(n^{3-\Omega(1/d)})$  for  $L_2$ -norm [6].

The remaining variant, Global-Fréchet, has only been studied very recently [20], although it is a reasonable measure: The Local constraints (i.e., matching each  $v_{i_{\ell}}$  to itself) are not necessary to enforce ordering along the curve, since Fréchet distance already takes the ordering of the vertices into account – in contrast to Hausdorff distance. Van Kreveld et al. [20] presented an  $\hat{O}(k^* \cdot n^5)$  time algorithm for Global-Fréchet simplification where  $k^*$  is the size of the optimal simplification.

## 1.1 Contribution 1: Algorithm for Global-Fréchet simplification

One could get the impression that Global-Fréchet simplification is a well-motivated, but computationally expensive curve simplification problem, in comparison to the local variants. We show that the latter intuition is wrong, by designing an  $\hat{O}(n^3)$ -time algorithm for Global-Fréchet simplification improving the previously best  $\hat{O}(k^* \cdot n^5)$ -time algorithm [20].

► **Theorem 1** (Section 3). *Global-Fréchet simplification can be solved in time  $\hat{O}(n^3)$ .*

This shows that all three problem variants (Local-Hausdorff, Local-Fréchet, and Global-Fréchet) can be solved in time  $\hat{O}(n^3)$ , and thus the choice of which problem variant to apply should not be made for computational reasons, at least in high dimensions.

## 1.2 Contribution 2: Conditional lower bound

Since all three variants can be solved in time  $\hat{O}(n^3)$ , the question arises whether any of them can be solved in time  $\hat{O}(n^{3-\varepsilon})$ . Tools to (conditionally) rule out such algorithms have been developed in recent years in the area of *fine-grained complexity*, see, e.g., the survey [21]. One of the most widely used fine-grained hypotheses is the following.

**$k$ -OV Hypothesis.** *Problem:* Given sets  $A_1, \dots, A_k \subseteq \{0, 1\}^d$  of size  $n$ , determine whether there exist vectors  $a_1 \in A_1, \dots, a_k \in A_k$  that are orthogonal, i.e., for each dimension  $j \in [d]$  there is an  $i \in [k]$  with  $a_i[j] = 0$ .

*Hypothesis:* For any  $k \geq 2$  and  $\varepsilon > 0$  the problem is not in time  $\hat{O}(n^{k-\varepsilon})$ .

Naively,  $k$ -OV can be solved in time  $\hat{O}(n^k)$ , and the hypothesis asserts that no polynomial improvement is possible, at least not with polynomial dependence on  $d$ . Buchin et al. [7] used the 2-OV hypothesis to rule out  $\hat{O}(n^{2-\varepsilon})$ -time algorithms for Local-Hausdorff<sup>2</sup> in the

<sup>1</sup> In  $\hat{O}$ -notation we *hide any polynomial factors in  $d$* , but we make exponential factors in  $d$  explicit.

<sup>2</sup> Their proof can be adapted to also work for Local-Fréchet and Global-Fréchet.

$L_1$ ,  $L_2$ , and  $L_\infty$  norm. This yields a tight bound (in  $\hat{O}$  sense) for  $L_\infty$ , since an  $\hat{O}(n^2)$ -time algorithm is known [6]. However, for all other  $L_p$ -norms ( $p \in [1, \infty)$ ), the question remained open whether  $\hat{O}(n^{3-\varepsilon})$ -time algorithms exist. To answer this question, one could try to generalize the conditional lower bound by Buchin et al. [7] to start from 3-OV. However, curve simplification problems seem to have the wrong “quantifier structure” for such a reduction (we will make this clearer when we give a brief overview of the reduction.) For similar reasons, Abboud et al. [2] introduced the Hitting Set Hypothesis, in which they essentially consider a variant of 2-OV where we have a universal quantifier over the first set of vectors and an existential quantifier over the second one ( $\forall\exists$ -OV). From their hypothesis, however, it is not known how to prove higher lower bounds than quadratic. We therefore consider the following natural extension of their hypothesis. This problem was studied in a more general context by Gao et al. [14].

**$\forall\forall\exists$ -OV Hypothesis.** *Problem:* Given sets  $A, B, C \subseteq \{0, 1\}^d$  of size  $n$ , determine whether for all  $a \in A, b \in B$  there is  $c \in C$  such that  $a, b, c$ , are orthogonal, i.e.,  $\sum_{\ell \in [d]} a[\ell]b[\ell]c[\ell] = 0$ .

*Hypothesis:* For any  $\varepsilon > 0$  the problem is not in time  $\hat{O}(n^{3-\varepsilon})$ .

No algorithm violating this hypothesis is known, and even for much stronger hypotheses on variants of  $k$ -OV and Satisfiability no such algorithms are known, see Section 5 in the full version for details. This shows that the hypothesis is plausible, in addition to being a natural generalization of the hypothesis of Abboud et al. [2].

We now give an brief overview of an  $\forall\forall\exists$ -OV-based lower bound for curve simplification. Given a  $\forall\forall\exists$ -OV instance on vectors  $A, B, C \subseteq \{0, 1\}^d$ , we construct corresponding point sets  $\tilde{A}, \tilde{B} \subset \mathbb{R}^{d'}$  (for some  $d' = O(d)$ ), forming two clusters that are very far apart from each other. We add a start- and an endpoint, which can be chosen far away from these clusters (in a new direction). Near the midpoint between  $\tilde{A}$  and  $\tilde{B}$ , another set of points  $\tilde{C}$  is constructed. The final curve then starts in the startpoint, walks through all points in  $\tilde{A}$ , then through all points in  $\tilde{C}$ , then through all points in  $\tilde{B}$ , and ends in the endpoint. Choosing a size-4 simplification implements an existential quantifier over  $a \in A, b \in B$ . The constraints that all  $\tilde{c} \in \tilde{C}$  are close to the line segment from  $\tilde{a}$  to  $\tilde{b}$  implements a universal quantifier over  $c \in C$ . Naturally, we want the distance from  $\tilde{c}$  to the line segment  $\tilde{a}\tilde{b}$  to be large if  $a, b, c$  are orthogonal, and to be small otherwise. This simulates the negation of  $\forall\forall\exists$ -OV, so any curve simplification algorithm can be turned into an algorithm for  $\forall\forall\exists$ -OV.

► **Theorem 2** (Section 4). *Over  $(\mathbb{R}^d, L_p)$  for any  $p \in [1, \infty)$  with  $p \neq 2$ , Local-Hausdorff, Local-Fréchet, and Global-Fréchet simplification have no  $\hat{O}(n^{3-\varepsilon})$ -time algorithm for any  $\varepsilon > 0$ , unless the  $\forall\forall\exists$ -OV Hypothesis fails. This holds even for the problem of deciding whether the optimal simplification has size  $\leq 4$  or  $\geq 5$ .*

In particular, this rules out improving the  $2^{O(d)}n^2$ -time algorithm for Local-Hausdorff over  $L_1$  [6] to a polynomial dependence on  $d$ . Note that the theorem statement excludes two interesting values for  $p$ , namely  $\infty$  and 2. For  $p = \infty$ , an  $\hat{O}(n^2)$ -time algorithm is known for Local-Hausdorff [6], so proving the above theorem also for  $p = \infty$  would immediately yield an algorithm breaking the  $\forall\forall\exists$ -OV Hypothesis.

For  $p = 2$ , we do not have such a strong reason why it is excluded, however, we now argue that at least a significantly different proof would be necessary in this case. We want the points  $\tilde{C}$  to lie in the middle between  $\tilde{A}$  and  $\tilde{B}$ , which essentially means that we want to consider the distance from  $(\tilde{a} + \tilde{b})/2$  to  $\tilde{c}$ . Now consider just a single dimension of  $\forall\forall\exists$ -OV. Then our task boils down to constructing points  $a_0, a_1$  and  $b_0, b_1$  and  $c_0, c_1$ , corresponding

## 18:4 Polyline Simplification has Cubic Complexity

to the bits in this dimension, such that  $\|(a_i + b_j)/2 - c_k\|_p = \beta_1$  if  $i = j = k = 1$  and  $\beta_0$  otherwise, with  $\beta_1 < \beta_0$ . Writing  $a'_i = a_i/2$  and  $b'_j = b_j/2$  for simplicity, in the case  $p = 2$  we can simplify

$$\begin{aligned} \|a'_i + b'_j - c_k\|_2^2 &= \sum_{\ell=1}^{d'} (a'_i[\ell] + b'_j[\ell] - c_k[\ell])^2 \\ &= \sum_{\ell=1}^{d'} \left( (a'_i[\ell] + b'_j[\ell])^2 + (a'_i[\ell] - c_k[\ell])^2 + (b'_j[\ell] - c_k[\ell])^2 - a'_i[\ell]^2 - b'_j[\ell]^2 - c_k[\ell]^2 \right) \\ &= \|a'_i + b'_j\|_2^2 + \|a'_i - c_k\|_2^2 + \|b'_j - c_k\|_2^2 - \|a'_i\|_2^2 - \|b'_j\|_2^2 - \|c_k\|_2^2 \\ &= f_1(i, j) + f_2(j, k) + f_3(i, k), \end{aligned}$$

for some functions  $f_1, f_2, f_3: \{0, 1\} \times \{0, 1\} \rightarrow \mathbb{R}$ . Note that by assumption this is equal to  $\beta_1^2$  if  $i = j = k = 1$  and  $\beta_0^2$  otherwise, with  $\beta_1 < \beta_0$ . However, it can be checked that such functions do not exist<sup>3</sup>. Therefore, for  $p = 2$  *our outlined reduction cannot work - provably!* We nevertheless make this reduction work for  $p \in [1, \infty)$ ,  $p \neq 2$ . The above argument shows that the construction is necessarily subtle. Indeed, constructing the right points requires some technical effort, see Section 4. This leaves open the possibility of a faster curve simplification algorithm for  $L_2$ , but such a result would need to exploit the Euclidean norm very heavily.

### 1.3 Further related work

Curve simplification has been studied in a variety of different formulations and settings. To list some examples, it was shown that the algorithm by Douglas and Peucker [11] can be implemented in time  $O(n \log n)$  [17], and that the classic  $O(n^3)$ -time algorithm for Local-Hausdorff simplification by Imai and Iri [18] can be implemented in time  $O(n^2)$  in two dimensions [8, 19]. More topics include curve simplification without self-intersections [10], Local-Hausdorff simplification with angular constraints between consecutive line segments [9], approximation algorithms [3], streaming algorithms [1], and the use of curve simplification in subdivision algorithms [16, 12, 13].

### 1.4 Organization

In Section 2 we formally define the problems studied in this paper. In Section 3 we present our new algorithm for Global-Fréchet, and in Section 4 we show our conditional lower bounds.

## 2 Preliminaries

Our ambient space is the metric space  $(\mathbb{R}^d, L_p)$ , where the distance between points  $x, y \in \mathbb{R}^d$  is the  $L_p$ -norm of their difference, i.e.,  $\|x - y\|_p = \left( \sum_{i=1}^d (x[i] - y[i])^p \right)^{1/p}$ . A *polyline*  $P$  of size  $n + 1$ , given by a sequence of points  $\langle v_0, v_1, \dots, v_n \rangle$  is the continuous curve that starts in  $v_0$ , walks along the line segments  $\overline{v_i v_{i+1}}$  for  $i = 0, \dots, n - 1$  in order, and ends in  $v_n$ . We also interpret  $P$  as a function  $P: [0, n] \rightarrow \mathbb{R}^d$  where  $P[i + \lambda] = (1 - \lambda)v_i + \lambda v_{i+1}$  for any  $\lambda \in [0, 1]$  and  $i \in \{0, \dots, n - 1\}$ . We use the notation  $P[t_1 \dots t_2]$  to represent the sub-polyline of  $P$  between  $P[t_1]$  and  $P[t_2]$ . Formally for integers  $0 \leq i \leq j \leq n$  and reals  $\lambda_1, \lambda_2 \in [0, 1)$ ,

$$P[i + \lambda_1 \dots j + \lambda_2] = \langle (1 - \lambda_1)v_i + \lambda_1 v_{i+1}, v_{i+1}, \dots, v_j, (1 - \lambda_2)v_j + \lambda_2 v_{j+1} \rangle$$

<sup>3</sup> We can express this situation by a linear system of equations in 12 variables (4 image values for each function  $f_i$ ) and 8 equations (for the values of  $f$  on  $i, j, k \in \{0, 1\}$ ) and verify that it has no solution.

A *simplification* of  $P$  is a curve  $Q = \langle v_{i_0}, v_{i_1}, \dots, v_{i_m} \rangle$  with  $0 = i_0 < i_1 < \dots < i_m = n$ . The size of the simplification  $Q$  is  $m + 1$ . Our goal is to determine a simplification of minimal size that “very closely” represents  $P$ . To this end we define two popular measures of similarity between the curves, namely the Fréchet and Hausdorff distances.

► **Definition 3** (Fréchet distance). *The (continuous) Fréchet distance  $\delta_F(P_1, P_2)$  between two curves  $P_1$  and  $P_2$  of size  $n$  and  $m$  respectively is*

$$\delta_F(P_1, P_2) = \inf_f \max_{t \in [0, n]} \|P_1[t] - P_2[f(t)]\|_p$$

where  $f: [0, n] \rightarrow [0, m]$  is monotone with  $f(0) = 0$  and  $f(n) = m$ .

Alt and Godau [5] characterize the Fréchet distance in terms of the “free-space diagram”.

► **Definition 4** (Free-Space). *Given two curves  $P_1, P_2$  and  $\delta \geq 0$ , the free-space  $FS_\delta(P_1, P_2) \subseteq \mathbb{R}^2$  is the set  $\{(x, y) \in ([0, n] \times [0, m]) \mid \|P_1[x] - P_2[y]\|_p \leq \delta\}$ .*

Consider the following decision problem. Given two curves  $P_1, P_2$  of size  $n$  and  $m$ , respectively, and given  $\delta \geq 0$ , decide whether  $\delta_F(P_1, P_2) \leq \delta$ . The answer to this question is yes if and only if  $(n, m)$  is reachable from  $(0, 0)$  by a monotone path through  $FS_\delta(P_1, P_2)$ . This “reachability” problem is known to be solvable by a dynamic programming algorithm in time  $\mathcal{O}(nm)$ . In particular, if either  $P_1$  or  $P_2$  is a line segment, then the decision problem can be solved in linear time.

The Hausdorff distance between curves ignores the ordering of the points along the curve. Intuitively, if we remove the monotonicity condition from function  $f$  in Definition 3 we obtain the directed Hausdorff distance between the curves. Formally, it is defined as follows.

► **Definition 5** (Hausdorff distance). *The (directed) Hausdorff distance  $\delta_H(P_1, P_2)$  between curves  $P_1$  and  $P_2$  of size  $n$  and  $m$ , respectively, is*

$$\delta_H(P_1, P_2) = \max_{t_1 \in [0, n]} \min_{t_2 \in [0, m]} \|P_1[t_1] - P_2[t_2]\|_p$$

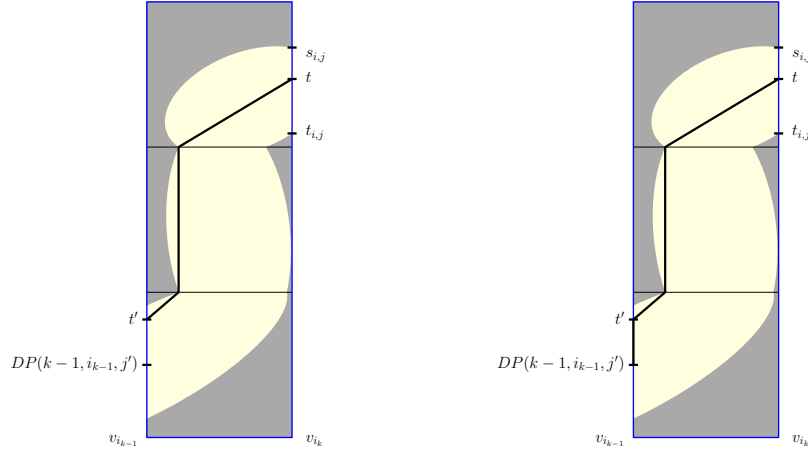
In order to measure the “closeness” between a curve and its simplification, these above similarity measures can be applied either *globally* to the whole curve and its simplification, or *locally* to each simplified subcurve  $P[i_\ell \dots i_{\ell+1}]$  and the segment  $\overline{v_{i_\ell}, v_{i_{\ell+1}}}$  to which it was simplified (taking the maximum over all  $\ell$ ). This gives rise to the following measures for curve simplification.

► **Definition 6** (Similarity for Curve Simplification). *Given a curve  $P = \langle v_0, v_1, \dots, v_n \rangle$  and a simplification  $Q = \langle v_{i_0}, v_{i_1}, \dots, v_{i_m} \rangle$  of  $P$ , we define their Global-Hausdorff distance as  $\delta_H(P, Q)$  and their Local-Hausdorff distance<sup>4</sup> as  $\max_{0 \leq \ell \leq m-1} \delta_H(P[i_\ell \dots i_{\ell+1}], \overline{v_{i_\ell}, v_{i_{\ell+1}}})$ . Their Global-Fréchet and Local-Fréchet distance are defined similarly, with  $\delta_H$  replaced by  $\delta_F$ .*

### 3 Algorithms for Global-Fréchet simplification

In this section we present an  $\mathcal{O}(n^3)$  time algorithm for curve simplification under Global-Fréchet distance, i.e., we prove Theorem 1.

<sup>4</sup> It can be checked that in this expression directed and undirected Hausdorff distance have the same value, and so for Local-Hausdorff we can without loss of generality use the directed Hausdorff distance.



■ **Figure 1** There is a monotone path from  $(0, t')$  to  $(1, t)$  in  $FS_\delta(P, \overline{v_{i_{k-1}}v_{i_k}})$  (left). Since  $DP(k-1, i_{k-1}, j') \leq t' \leq t$ , there is a monotone path in from  $(0, DP(k-1, i_{k-1}, j'))$  (right) to  $(1, t)$  by moving from  $(0, DP(k-1, i_{k-1}, j'))$  to  $(0, t')$  and then following the previous monotone path.

### 3.1 An $O(kn^5)$ algorithm for Global Fréchet simplification

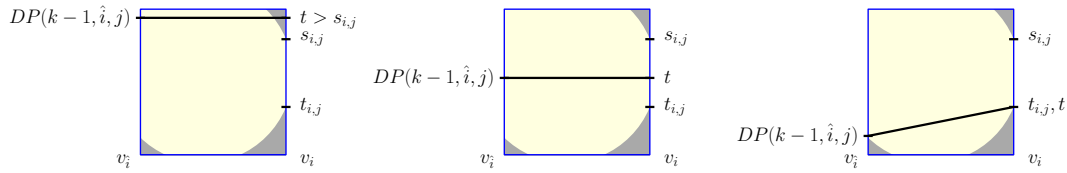
We start by describing the previously best algorithm by [20]. Let  $P$  be the polyline  $\langle v_0, v_1, \dots, v_n \rangle$ . Let  $DP(k, i, j)$  represent the earliest reachable point on  $\overline{v_j v_{j+1}}$  with a length  $k$  simplification of the polyline  $P[0 \dots i]$ , i.e.,  $DP(k, i, j)$  represents the smallest  $t$  such that  $P[t]$  lies on the line-segment  $\overline{v_j v_{j+1}}$  (i.e.  $j \leq t \leq j+1$ ) and there is a simplification  $\tilde{Q}$  of the polyline  $P[0 \dots i]$  of size at most  $k$  such that  $\delta_F(\tilde{Q}, P[0 \dots t]) \leq \delta$ . If such a point does not exist then we set  $DP(k, i, j) = \infty$ . To solve Global-Fréchet simplification, we need to return the minimum  $k$  such that  $DP(k, n, n-1) \neq \infty$ . Let  $P[t_{i,j}]$  and  $P[s_{i,j}]$  be the first point and the last point respectively on the line segment  $\overline{v_j v_{j+1}}$  such that  $\|v_i - P[t_{i,j}]\|_p \leq \delta$  and  $\|v_i - P[s_{i,j}]\|_p \leq \delta$ . Observe that if  $DP(k, i, j) \neq \infty$  then  $t_{i,j} \leq DP(k, i, j) \leq s_{i,j}$  for all  $k$ . We will crucially make use of the following characterization of the DP table entries.

► **Lemma 7.**  $DP(k, i, j)$  is the minimal  $t \in [t_{i,j}, s_{i,j}]$ , such that for some  $i' < i$  and  $j' \leq j$ , we have  $DP(k-1, i', j') \neq \infty$  and  $\delta_F(P[DP(k-1, i', j') \dots t], \overline{v_{i'} v_i}) \leq \delta$ . If no such  $t$  exists then  $DP(k, i, j) = \infty$ .

**Proof Sketch.**  $DP(k, i, j)$  is the minimal  $t \in [t_{i,j}, s_{i,j}]$  such that for some  $i' < i$  and  $t' \leq t$  we have a simplification  $\hat{Q}$  of the polyline  $P[0 \dots i']$  and  $\delta_F(\hat{Q}, P[0 \dots t']) \leq \delta$  and  $\delta_F(P[t' \dots t], \overline{v_{i'} v_i}) \leq \delta$ , and  $DP(k, i, j) = \infty$  if no such  $t$  exists. Let  $j' \leq t' \leq j+1$  and  $i' < i$ , then  $DP(k-1, i', j') \neq \infty$  and by inspecting  $FS_\delta(P, \overline{v_{i'} v_i})$ , it is clear that there also exists a monotone path from  $(0, DP(k-1, i', j'))$  to  $(1, t)$  (see Figure 1). Thus it suffices to consider only  $DP(k-1, i', j')$  as candidates for  $t'$ . ◀

A dynamic programming algorithm follows more or less directly from Lemma 7. Note that for fixed  $i' < i$  and  $j' \leq j$  such that  $DP(k-1, i', j') \neq \infty$  we can determine the minimal  $t$  such that  $(1, t)$  is reachable from  $(0, DP(k-1, i', j'))$  by a monotone path in  $FS_\delta(P, \overline{v_{i'} v_i})$  in  $\mathcal{O}(n)$  time. This follows from the standard algorithm for the decision version of the Fréchet distance between two polygonal curves of length at most  $n$  (in particular here one of the curves is of length 1). To determine  $DP(k, i, j)$  we enumerate over all  $i' < i$  and  $j' \leq j$  such that  $DP(k-1, i', j') \neq \infty$  and determine the minimum  $t$  that is reachable. The running time to determine  $DP(k, i, j)$  is thus  $\mathcal{O}(n^3)$ . As there are  $\mathcal{O}(k^*n^2)$  DP-cells to fill, the algorithm runs in time  $\mathcal{O}(k^*n^5)$  where  $k^*$  is the size of the optimal simplification.





■ **Figure 2** For  $\hat{i} < i$ ,  $t \in [t_{i,j}, s_{i,j}]$  is minimal such that  $(1, t)$  is reachable from  $(0, DP(k - 1, \hat{i}, j))$  by a monotone path in  $fbx_j$ . If  $DP(k - 1, \hat{i}, j) > s_{i,j}$  (left) then no such  $t$  exists. If  $t_{i,j} \leq DP(k - 1, \hat{i}, j) \leq s_{i,j}$  (middle) then  $t = DP(k - 1, \hat{i}, j)$ . If  $DP(k - 1, \hat{i}, j) < t_{i,j}$  (right) then  $t = t_{i,j}$ .

### 3.2 An $\mathcal{O}(n^3)$ algorithm for Global-Fréchet simplification

Now we improve the running time by a more careful understanding of the monotone paths through  $FS_\delta(P, \overline{v_i v_i})$  to  $(1, DP(k, i, j))$  for fixed  $i, j$  and  $i'$ . Let  $fbx_j$  denote the intersection of the free-space  $FS_\delta(P, \overline{v_i v_i})$  with the square with corner vertices  $(0, j)$  and  $(1, j + 1)$ . The following fact will be useful later.

► **Fact 8.**  $fbx_j$  is convex for all  $j \in [n - 1]$ .

Let  $ver_j$  be the free space on the vertical line segment with endpoints  $(0, j)$  and  $(0, j + 1)$  and let  $hor_j$  be the free space on the horizontal line segment  $(0, j)$  to  $(1, j)$  in the free space  $FS_\delta(P, \overline{v_i v_i})$ . We consider the point  $(0, j)$  to belong to  $ver_j$ , but not  $hor_j$ , to avoid certain corner cases. We split the monotone paths from  $(0, DP(k - 1, i', j'))$  for  $i' < i$  and  $j' \leq j$  to  $(1, DP(k, i, j))$  in  $FS_\delta(P, \overline{v_i v_i})$  into two categories: the ones that intersect  $ver_j$  and the ones that intersect  $hor_j$ . We first look at the monotone paths that intersect  $ver_j$ . If the monotone path intersects  $ver_j$  then  $j' = j$ . Let  $\overline{DP}_1(k, i, j) = \min_{i' < i} DP(k - 1, i', j)$ . We define

$$DP_1(k, i, j) = \begin{cases} \max(\overline{DP}_1(k, i, j), t_{i,j}) & \text{if } \overline{DP}_1(k, i, j) \leq s_{i,j} \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

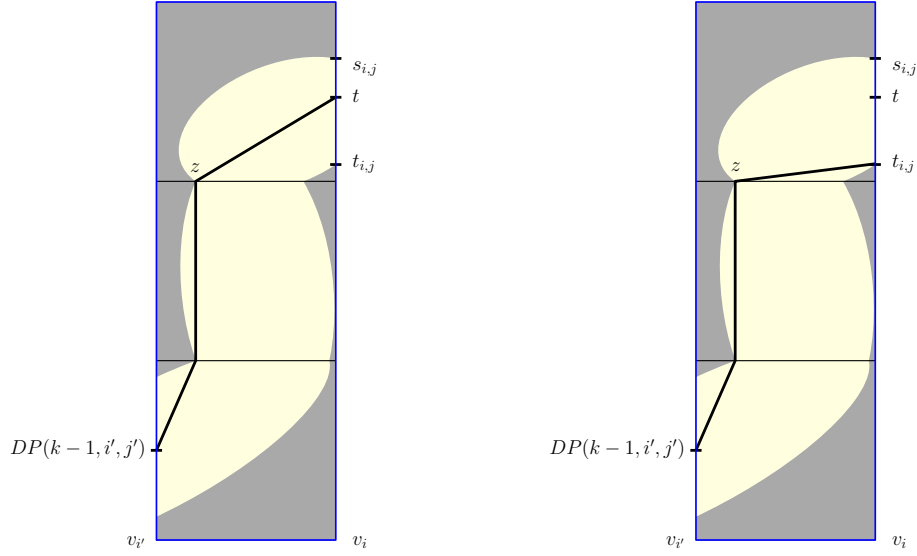
We show a characterization of  $DP_1$  similar to the characterization of  $DP$  in Lemma 7 and thus establish that  $DP_1$  correctly handles all paths intersecting  $ver_j$ .

► **Observation 9.**  $DP_1(k, i, j)$  is the minimal  $t \in [t_{i,j}, s_{i,j}]$  such that  $DP(k - 1, i', j) \neq \infty$  and  $\delta_F(P[DP(k - 1, i', j) \dots t], \overline{v_i v_i}) \leq \delta$  for some  $i' < i$ . If no such  $t$  exists then  $DP_1(k, i, j) = \infty$ .

**Proof Sketch.** Observe that for any  $\hat{i} < i$ , the minimal  $t \in [t_{i,j}, s_{i,j}]$  such that there is a monotone path from  $(0, DP(k - 1, \hat{i}, j))$  to  $(1, t)$  in  $FS_\delta(P, \overline{v_i v_i})$  or equivalently in  $fbx_j$  is  $\max(DP(k - 1, \hat{i}, j), t_{i,j})$  if  $DP(k - 1, \hat{i}, j) \leq s_{i,j}$  (see Figure 2 middle and right) and  $t$  does not exist when  $DP(k - 1, \hat{i}, j) > s_{i,j}$  (see Figure 2 left). Therefore, if  $DP_1(k, i, j) = \min_{i' < i} DP(k - 1, i', j) \leq s_{i,j}$  then the minimal  $t \in [t_{i,j}, s_{i,j}]$  that is reachable from  $(0, DP(k - 1, i', j))$  for  $i' < i$  is  $\max(\min_{i' < i} (DP(k - 1, i', j), t_{i,j}) = \max(DP_1(k, i, j), t_{i,j})$ . Otherwise (if  $DP_1(k, i, j) = \min_{i' < i} DP(k - 1, i', j) > s_{i,j}$ ) then no such  $t$  exists. ◀

We now look at the monotone paths that intersect  $hor_j$ . Observe that if the monotone path intersects  $hor_j$  then  $j' < j$ . Along this line, we define  $\overline{DP}_2(k, i, j) = 1$  if there exists some  $i' < i$  and  $j' < j$ , such that  $DP(k - 1, i', j') \neq \infty$  and there exists a monotone path from  $(0, DP(k - 1, i', j'))$  to  $(1, t_{i,j})$  in the free-space  $FS_\delta(P, \overline{v_i v_i})$ , and otherwise we set  $\overline{DP}_2(k, i, j) = 0$ . Using this, we define

$$DP_2(k, i, j) = \begin{cases} t_{i,j} & \text{if } \overline{DP}_2(k, i, j) = 1 \\ \infty & \text{otherwise} \end{cases}$$



■ **Figure 3** For  $t \geq t_{i,j}$ , there is a monotone path from  $(0, DP(k-1, i', j'))$  to  $(1, t)$  in  $FS_\delta(P, \overline{v_{i'}v_i})$  (left) for  $i' < i$  and  $j' < j$  that intersect  $hor_j$  at  $z$ . Then there is also a monotone path from  $(0, DP(k-1, i', j'))$  to  $(1, t_{i,j})$  (right) following the previous monotone path upto  $z$  and then from  $z$  to  $(1, t_{i,j})$ .

We show a characterization of  $DP_2$  similar our characterization of DP in Lemma 7, thus establishing that  $DP_2$  correctly handles all paths intersecting  $hor_j$ .

► **Observation 10.**  $DP_2(k, i, j)$  is the minimal  $t \in [t_{i,j}, s_{i,j}]$  such that  $DP(k-1, i', j') \neq \infty$  and  $\delta_F(P[DP(k-1, i', j') \dots t], \overline{v_{i'}v_i}) \leq \delta$  for some  $i' < i$  and  $j' < j$ . If no such  $t$  exists then  $DP_2(k, i, j) = \infty$ .

► **Proof Sketch.** The key idea is that if any  $t \in [t_{i,j}, s_{i,j}]$ ,  $(1, t)$  is reachable by a monotone path in  $FS_\delta(P, \overline{v_{i'}v_i})$  from  $(0, DP(k-1, i', j'))$  for  $i' < i$ , then so is  $(1, t_{i,j})$  (see Figure 3). ◀

► **Lemma 11.**  $DP(k, i, j) = \min(DP_1(k, i, j), DP_2(k, i, j))$ .

► **Proof.** Follows directly from Observations 7, 9, and 10. ◀

In particular this yields a dynamic programming formulation for  $DP(k, i, j)$ , since both  $DP_1(k, i, j)$  and  $DP_2(k, i, j)$  depend on values of  $DP(k', i', j')$  with  $k' < k$ ,  $i' < i$  and  $j' \leq j$ .

We define  $\kappa(i, j)$  as the minimal  $k$  such that  $DP(k, i, j) \neq \infty$ . Similarly we define  $\kappa_1(i, j)$  and  $\kappa_2(i, j)$  as the minimal  $k$  such that  $DP_1(k, i, j) \neq \infty$  and  $DP_2(k, i, j) \neq \infty$ , respectively. Note that  $\kappa(i, j) = \min(\kappa_1(i, j), \kappa_2(i, j))$  (by Lemma 11). Also note that both  $\kappa_1(i, j)$  and  $\kappa_2(i, j)$  depend only on the values of  $DP(k', i', j')$  with  $k' < k$ ,  $i' < i$  and  $j' \leq j$ .

With these preparations can now present our dynamic programming algorithm, except for one subroutine  $\kappa_2$ -subroutine( $i$ ) that we describe in Section 3.3. In particular, for any  $i$ ,  $\kappa_2$ -subroutine( $i$ ) determines  $\kappa_2(i, j)$  for all  $j \in [n]$  only using the values of  $\kappa(i', j)$  for all  $i' < i$  and all  $0 \leq j \leq n-1$ . Now we show how to compute  $DP_1(k, i, j)$ . Observe that for any  $i, j$  and  $k$  we can compute  $\overline{DP}_1(k, i, j)$  from  $\overline{DP}_1(k, i-1, j)$  and  $DP(k-1, i-1, j)$  as  $\overline{DP}_1(k, i, j) = \min(\overline{DP}_1(k, i-1, j), DP(k-1, i-1, j))$ . Then we can compute  $DP_1(k, i, j)$  by using equation (1) and set  $\kappa_1(i, j)$  to the minimal  $k$  such that  $DP_1(k, i, j) \neq \infty$ . We compute  $DP_2(k, i, j)$  by setting  $DP_2(k, i, j) = t_{i,j}$  if  $k \geq \kappa_2(i, j)$  and  $DP_2(k, i, j) = \infty$  otherwise. Also, we set  $\kappa(i, j)$  as  $\min(\kappa_1(i, j), \kappa_2(i, j))$ . After that we can update  $DP(k, i, j)$  by the formulation in Lemma 11.

---

**Algorithm 1** Solving curve simplification under Global-Fréchet distance.
 

---

```

1: Precompute all  $t_{i,j}$  and  $s_{i,j}$  and initialize all  $\overline{DP}_1(k, 0, j)$ ,  $DP(k, 0, j)$ ,  $\kappa(0, j)$  and  $DP(0, i, j)$ .
2: for all  $i = 1$  to  $n$  do
3:   Determine  $\kappa_2(i, j)$  for all  $0 \leq j \leq n - 1$  using  $\kappa_2$ -subroutine( $i$ )
4:   for  $j = 0$  to  $n - 1$  do
5:     for every  $k \in [n + 1]$  set  $\overline{DP}_1(k, i, j)$  to  $\min(\overline{DP}_1(k, i - 1, j), DP(k - 1, i - 1, j))$ 
6:     for every  $k \in [n + 1]$  set  $DP_1(k, i, j)$  to  $\max(\overline{DP}_1(k, i, j), t_{i,j})$  if  $\overline{DP}_1(k, i, j) \leq s_{i,j}$ 
       and to  $\infty$  otherwise
7:     set  $\kappa_1(i, j)$  to the smallest  $k$  such that  $DP_1(k, i, j) \neq \infty$ 
8:     set  $\kappa(i, j) = \min(\kappa_1(i, j), \kappa_2(i, j))$ 
9:     for every  $k \in [n + 1]$  set  $DP_2(k, i, j)$  to  $t_{i,j}$  if  $k \geq \kappa_2(i, j)$  and to  $\infty$  otherwise
10:    for every  $k \in [n + 1]$  set  $DP(k, i, j)$  to  $\min(DP_1(k, i, j), DP_2(k, i, j))$ 

```

---

Denote the running time of  $\kappa_2$ -subroutine( $i$ ) by  $T(n)$ . Since we fill each of  $O(n^3)$  DP cells in time  $O(1)$ , the total running time of Algorithm 1 is  $O(n^3 + n \cdot T(n))$ .

### 3.3 Implementing the $\kappa_2$ -subroutine( $i$ )

In this subsection we show how to implement the  $\kappa_2$ -subroutine( $i$ ) in time  $T(n) = \mathcal{O}(n^2)$ . Then in total we have  $\mathcal{O}(n^3)$  for solving Global-Fréchet simplification.

#### 3.3.1 Cell Reachability

We introduce an auxiliary problem *Cell Reachability*. We shall see later that an  $\mathcal{O}(n)$  time solution to this problem ensures that the  $\kappa_2$ -subroutine( $i$ ) can be implemented in time  $\mathcal{O}(n^2)$ .

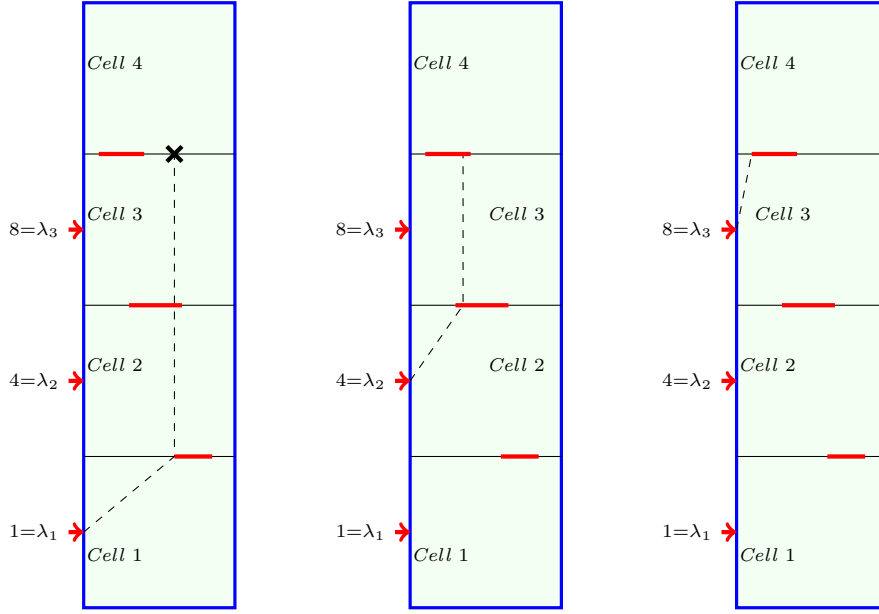
► **Definition 12.** *In an instance of the Cell Reachability problem, we are given:*

- A set of  $n$  cells. Each cell  $j$  with  $1 \leq j \leq n$  is a unit square with corner points  $(0, j)$  and  $(1, j + 1)$ . We say that cells  $j$  and  $j + 1$  are consecutive.
- An integral entry-cost  $\lambda_j > 0$  for every cell  $j$ .
- A set of  $n - 1$  passages between consecutive cells. The passage  $p_j$  is the horizontal line segment with endpoints  $(j, a_j)$  and  $(j, b_j)$  where  $b_j > a_j$ .

A cell  $j$  is reachable from a cell  $j'$  with  $j' < j$  if and only if there exists  $x_{j'+1} \leq x_{j'+2} \dots \leq x_j$  such that  $x_k \in [a_k, b_k]$  for every  $j' < k \leq j$ . Intuitively, cell  $j$  is reachable from cell  $j'$  if and only if there is a monotone path through the passages from cell  $j'$  to cell  $j$ . Let the exit-cost  $\mu_j$  of a cell  $j$  as the minimal  $\lambda_{j'}$  such that  $j$  is reachable from cell  $j'$ ,  $j' < j$ . The goal is to determine the sequence  $\langle \mu_1, \mu_2, \dots, \mu_n \rangle$ . See Figure 4 for an illustration.

We make a more refined notion of reachability. For any cells  $j$  and  $j' < j$  we define the *first reachable point*  $frp(j, j')$  on cell  $j$  from cell  $j'$  as the minimal  $t$  such that there exist  $x_{j'+1} \leq x_{j'+2} \leq \dots \leq x_j$  such that  $x_k \in [a_k, b_k]$  for every  $j' < k \leq j$  and  $x_j = t$ , and we set  $frp(j, j') = \infty$  if there exists no such  $t$ . Let  $t_j(k)$  be the first reachable point on cell  $j$  from any cell  $j'$  with entry-cost at most  $k$  i.e.  $t_j(k) = \min \{ frp(j, j') \mid j' < j, \lambda_{j'} \leq k \}$ . Note that  $\mu_j$  is the minimal  $k$  such that  $t_j(k) \neq \infty$ . Thus, it suffices to show how to determine  $t_j(\cdot)$  and  $\mu_j$  from  $t_j(\cdot)$  for all  $j \in [n]$  in  $\mathcal{O}(n)$  time. To this end we generalize an algorithm by Alt et al. [4, Lemma 2.3]; the details can be found in the full version of this paper.

► **Theorem 13.** *Cell Reachability can be solved in  $\mathcal{O}(n)$  time.*



■ **Figure 4** The red horizontal line segments between the cells indicate the passages. Cell 4 is only reachable from cells 2 and 3. Thus  $\mu_4 = \min(\lambda_2, \lambda_3) = \min(4, 8) = 4$ .

### 3.3.2 Implementing $\kappa_2$ -subroutine( $i$ ) using Cell Reachability

Recall the definition of  $\kappa_2(\cdot, \cdot)$  and what our goal is now: For a fixed  $i' < i$ , let  $\kappa(i, j, i')$  be the minimal  $k$  such that for some  $j' < j$ , we have  $\text{DP}(k-1, i', j') \neq \infty$  and  $\delta_F(P[\text{DP}(k-1, i', j') \dots t_{i,j}], \overline{v_{i'}v_i}) \leq \delta$ . Note that  $\kappa_2(i, j) = \min_{i' < i} \kappa(i, j, i')$ . In order to show that the  $\kappa_2$ -subroutine( $i$ ) can be implemented in  $\mathcal{O}(n^2)$ , it suffices to show that for fixed  $i' < i$  we can determine  $\kappa(i, j, i')$  for all  $j \in [n-1]$  in  $\mathcal{O}(n)$  time.

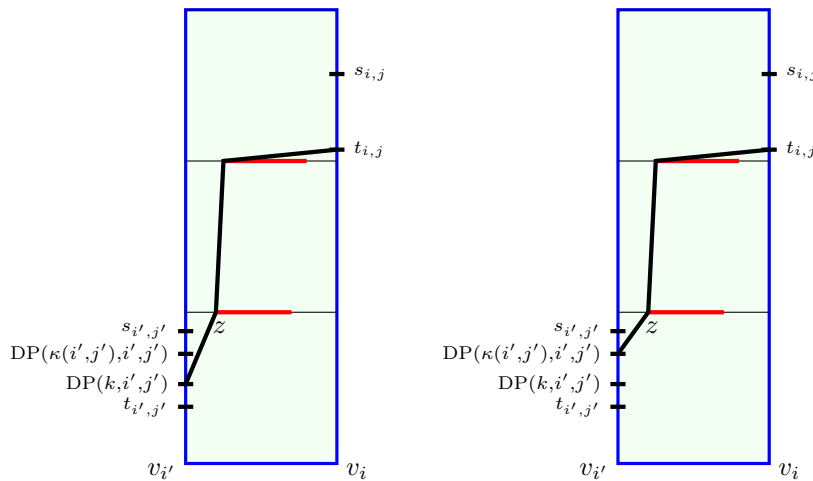
Let the line segment between  $(a_j, j)$  and  $(b_j, j)$  denote the free-space on  $\text{hor}_j$ . Due to the convexity of every cell in the free space  $FS_\delta(P, \overline{v_{i'}v_i})$ , it suffices to look at reachability through the free space at the boundary of the cells ( $\text{hor}_j$ ). Thus for any  $j' < j$  there is a monotone path from  $(0, \text{DP}(\kappa(i', j'), i', j'))$  to  $(1, t_{i,j})$  in the free-space  $FS_\delta(P, \overline{v_{i'}v_i})$  if and only if there exist  $x_{j'+1} \leq x_{j'+2} \leq \dots \leq x_j$  with each  $x_k \in [a_k, b_k]$  for all  $j' < k \leq j$ . Similarly, it suffices to look at the reachability only from the points  $(0, \text{DP}(\kappa(i', j'), i', j'))$ . This yields the following observation, whose proof is illustrated in Figure 5.

► **Observation 14.** *For any  $i' < i$  if there is a monotone path from  $(0, \text{DP}(k, i', j'))$  to  $(1, t_{i,j})$  in the free-space  $FS_\delta(P, \overline{v_{i'}v_i})$  intersecting  $\text{hor}_j$ , then there is also a monotone path from  $(0, \text{DP}(\kappa(i', j'), i', j'))$  to  $(1, t_{i,j})$  in the free-space  $FS_\delta(P, \overline{v_{i'}v_i})$  intersecting  $\text{hor}_j$ .*

Observation 14 implies that  $\kappa(i, j, i')$  is the minimal value of  $1 + \kappa(i', j')$  over all  $j' < j$  such that there exist  $x_{j'+1} \leq x_{j'+2} \leq \dots \leq x_j$  with  $x_k \in [a_k, b_k]$  for  $j' < k \leq j$ . Note that now we are in an instance of Cell Reachability. Thus for any fixed  $i'$  we can determine  $\kappa(i, j, i')$  in  $\mathcal{O}(n)$  time and therefore we can implement  $\kappa_2$ -subroutine( $i$ ) in  $\mathcal{O}(n^2)$  time.

## 4 Conditional lower bound for curve simplification

In this section we show that an  $\mathcal{O}(n^{3-\varepsilon} \text{poly}(d))$  time algorithm for Global-Fréchet, Local-Fréchet or Local-Hausdorff simplification over  $(\mathbb{R}^d, \|\cdot\|_p)$  for any  $p \in [1, \infty)$ ,  $p \neq 2$ , would yield an  $\mathcal{O}(n^{3-\varepsilon} \text{poly}(d))$  algorithm for  $\forall \exists \exists$ -OV.



■ **Figure 5** Illustration of the proof of Observation 14. For any  $i' < i, j' < j$  and any  $k$ , there is a monotone path from  $(0, DP(k, i', j'))$  to  $(1, t_{i,j})$  in  $FS_\delta(P, \overline{v_{i'}v_i})$  (left) that intersects  $hor_j$  at  $z$ . Then there is a monotone path from  $(0, DP(\kappa(i', j'), i', j'))$  to  $(1, t_{i,j})$  in  $FS_\delta(P, \overline{v_{i'}v_i})$  (right) by walking from  $(0, DP(\kappa(i', j'), i', j'))$  to  $z$  and then following the previous path from  $z$  to  $(1, t_{i,j})$ .

### 4.1 Overview of the reduction

Consider any instance  $(A, B, C)$  of  $\forall\forall\exists$ -OV where  $A, B, C \subseteq \{0, 1\}^d$  have size  $n$ . We write  $A = \{a_1, a_2, \dots, a_n\}, B = \{b_1, b_2, \dots, b_n\}$  and  $C = \{c_1, c_2, \dots, c_n\}$ . We will efficiently construct  $3n + 1$  points in  $\mathbb{R}^D$  with  $D \in \mathcal{O}(d)$  namely the sets of points  $\tilde{A} = \{\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_n\}, \tilde{B} = \{\tilde{b}_1, \tilde{b}_2, \dots, \tilde{b}_n\}$  and  $\tilde{C} = \{\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n\}$  and one more point  $s$ . We also determine  $\delta \geq 0$  such that the following properties are satisfied.

- (P<sub>1</sub>) For any  $\tilde{a} \in \tilde{A}, \tilde{b} \in \tilde{B}, \tilde{c} \in \tilde{C}$ , there is a point  $x$  on the line segment  $\overline{\tilde{a}\tilde{b}}$  with  $\|x - \tilde{c}\|_p \leq \delta$  if and only if  $\|\frac{\tilde{a} + \tilde{b}}{2} - \tilde{c}\|_p \leq \delta$ .
- (P<sub>2</sub>) For any  $\tilde{a} \in \tilde{A}, \tilde{b} \in \tilde{B}, \tilde{c} \in \tilde{C}$ , we have  $\|\frac{\tilde{a} + \tilde{b}}{2} - \tilde{c}\|_p \leq \delta$  if and only if  $\sum_{\ell \in [d]} a[\ell] \cdot b[\ell] \cdot c[\ell] \neq 0$ .
- (P<sub>3</sub>)  $\|x - y\|_p \leq \delta$  holds for all  $x, y \in \tilde{A}$ , and for all  $x, y \in \tilde{B}$  and for all  $x, y \in \tilde{C}$ .
- (P<sub>4</sub>) For any  $y_1, y_2 \in \{s\} \cup \tilde{B} \cup \tilde{C}$  and any point  $x$  on the line segment  $\overline{y_1y_2}$  we have  $\|x - \tilde{a}\|_p > \delta$  for all  $\tilde{a} \in \tilde{A}$ .
- (P<sub>5</sub>) For any  $y_1, y_2 \in \{s\} \cup \tilde{A} \cup \tilde{C}$  and any point  $x$  on the line segment  $\overline{y_1y_2}$  we have  $\|x - \tilde{b}\|_p > \delta$  for all  $\tilde{b} \in \tilde{B}$ .
- (P<sub>6</sub>) For any  $y \in \tilde{B} \cup \tilde{A}$  and any point  $x$  on the line segment  $\overline{sy}$  we have  $\|x - \tilde{c}\|_p > \delta$  for all  $\tilde{c} \in \tilde{C}$ .

We postpone the exact construction of these points. Our hard instance for curve simplification will be  $Q = \langle s, \tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_n, \tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n, \tilde{b}_1, \tilde{b}_2, \dots, \tilde{b}_n, s \rangle$ .

► **Fact 15.** Let  $r_1, r_2, s_1, s_2 \in \mathbb{R}^D$  and  $\|r_k - s_k\|_p \leq \delta$  for  $k \in [2]$ . Then  $\delta_F(\overline{r_1r_2}, \overline{s_1s_2}) \leq \delta$ .

► **Lemma 16.** Let  $\hat{Q} = \langle s, \tilde{a}_i, \tilde{b}_j, s \rangle$  for some  $\tilde{a}_i \in \tilde{A}$  and  $\tilde{b}_j \in \tilde{B}$ . If  $\|\frac{\tilde{a}_i + \tilde{b}_j}{2} - \tilde{c}\|_p \leq \delta$  for all  $\tilde{c} \in \tilde{C}$  then the Local-Frechet distance between  $Q$  and  $\hat{Q}$  is at most  $\delta$ .

**Proof.** Both  $Q$  and  $\hat{Q}$  have the same starting point  $s$ . By property P<sub>3</sub> we have  $\|\tilde{a} - \tilde{a}_i\|_p \leq \delta$  for all  $\tilde{a} \in \tilde{A}$ , and  $\|\tilde{b} - \tilde{b}_j\|_p \leq \delta$  for all  $\tilde{b} \in \tilde{B}$ . Thus it follows that  $\delta_F(\langle s, \tilde{a}_1, \dots, \tilde{a}_i \rangle, \langle s, \tilde{a}_i \rangle) \leq \delta$  and  $\delta_F(\langle \tilde{b}_j, \dots, \tilde{b}_n, s \rangle, \langle \tilde{b}_j, s \rangle) \leq \delta$ . It remains to show that  $\delta_F(Q_{ij}, \tilde{a}_i\tilde{b}_j) \leq \delta$  where  $Q_{ij} =$

## 18:12 Polyline Simplification has Cubic Complexity

$\langle \tilde{a}_i, \dots, \tilde{a}_n, \tilde{c}_1, \dots, \tilde{c}_n, \tilde{b}_1, \dots, \tilde{b}_j \rangle$ . To this end first note that both  $Q_{ij}$  and  $\overline{\tilde{a}_i \tilde{b}_j}$  have the same endpoints. We outline monotone walks within distance  $\delta$  on both  $Q_{ij}$  and  $\overline{\tilde{a}_i \tilde{b}_j}$ .

- (1) Walk on  $Q_{ij}$  from  $\tilde{a}_i$  to  $\tilde{a}_n$  and remain at  $\tilde{a}_i$  on  $\overline{\tilde{a}_i \tilde{b}_j}$ . (by Property **P<sub>3</sub>**)
- (2) Walk uniformly on both polylines, up to  $\frac{\tilde{a}_i + \tilde{b}_j}{2}$  on  $\overline{\tilde{a}_i \tilde{b}_j}$  and up to  $\tilde{c}_1$  on  $Q_{ij}$ . (by Fact 15)
- (3) Walk on  $Q_{ij}$  from  $\tilde{c}_1$  to  $\tilde{c}_n$  and remain at  $\frac{\tilde{a}_i + \tilde{b}_j}{2}$  on  $\overline{\tilde{a}_i \tilde{b}_j}$ . (by assumption)
- (4) Walk uniformly on both curves up to  $\tilde{b}_j$  on  $\overline{\tilde{a}_i \tilde{b}_j}$  and up to  $\tilde{b}_1$  on  $Q_{ij}$ . (by Fact 15)
- (5) Walk on  $Q_{ij}$  until  $\tilde{b}_j$  and remain at  $\tilde{b}_j$  on  $\overline{\tilde{a}_i \tilde{b}_j}$ . (by Property **P<sub>3</sub>**) ◀

Observe that property **P<sub>3</sub>** implies that there is a simplification of size five namely  $\hat{Q} = \langle s, \tilde{a}, \tilde{c}, \tilde{b}, s \rangle$  for any  $\tilde{a} \in \tilde{A}$ ,  $\tilde{b} \in \tilde{B}$ , and  $\tilde{c} \in \tilde{C}$ , such that the distance between  $\hat{Q}$  and  $Q$  is at most  $\delta$  under all three distance measures. We now show that a smaller simplification is only possible if there exist  $a \in A$ ,  $b \in B$  such that for all  $c \in C$  we have  $\sum_{\ell \in [d]} a[\ell] \cdot b[\ell] \cdot c[\ell] \neq 0$ .

► **Lemma 17.** *Let  $\hat{Q}$  be a simplification of the polyline  $Q$  of size 4. Then the following statements are equivalent:*

- (1) *The Global-Fréchet distance between  $Q$  and  $\hat{Q}$  is at most  $\delta$ .*
- (2) *The Local-Fréchet distance between  $Q$  and  $\hat{Q}$  is at most  $\delta$ .*
- (3) *The Local-Hausdorff distance between  $Q$  and  $\hat{Q}$  is at most  $\delta$ .*
- (4) *There exists  $\tilde{a} \in \tilde{A}$ ,  $\tilde{b} \in \tilde{B}$ , such that  $\hat{Q} = \langle s, \tilde{a}, \tilde{b}, s \rangle$  and  $\|\frac{\tilde{a} + \tilde{b}}{2} - \tilde{c}\|_p \leq \delta$  for every  $\tilde{c} \in \tilde{C}$ .*
- (5) *There exist  $a \in A$ ,  $b \in B$  such that for all  $c \in C$  we have  $\sum_{\ell \in [d]} a[\ell] \cdot b[\ell] \cdot c[\ell] \neq 0$ .*

A detailed proof can be found in the full version of this paper. Assuming that we can determine  $Q$  and  $\delta$  in  $\mathcal{O}(nd)$  time, the above lemma directly yields the following theorem.

► **Theorem 18.** *For any  $\varepsilon > 0$ , there is no  $\mathcal{O}(n^{3-\varepsilon} \text{poly}(d))$  algorithm for Global-Fréchet, Local-Fréchet and Local-Hausdorff simplification over  $(\mathbb{R}^d, \|\cdot\|_p)$  for any  $p \in [1, \infty)$ ,  $p \neq 2$ , unless  $\forall \exists$ -OV Hypothesis fails. This holds even for the problem of deciding whether the optimal simplification has size  $\leq 4$  or  $\geq 5$ .*

**Proof.** given an instance  $A, B, C$  of  $\forall \exists$ -OV, we can construct the curve  $Q$  and  $\delta$  in time  $\mathcal{O}(nd)$ . By Lemma 17, the simplification problem on  $(Q, \delta)$  is equivalent to  $\forall \exists$ -OV on  $A, B, C$ . Thus, any  $\mathcal{O}(n^{3-\varepsilon} \text{poly}(d))$  time algorithm for the curve simplification problem would yield an  $\mathcal{O}(n^{3-\varepsilon} \text{poly}(d))$  time algorithm for  $\forall \exists$ -OV. ◀

It remains to construct the point  $s$ , the sets  $\tilde{A}$ ,  $\tilde{B}$  and  $\tilde{C}$  and  $\delta$ . We first introduce some notation. For vectors  $x$  and  $y$  and  $\alpha \in [-\frac{1}{2}, \frac{1}{2}]$ , we define  $P_{xy}(\alpha)$  as  $(\frac{1}{2} - \alpha)x + (\frac{1}{2} + \alpha)y$ . Moreover let  $u_i \in \mathbb{R}^d$ . We write  $v = [u_1 u_2 \dots u_m]$  for the vector  $v \in \mathbb{R}^{md}$  with  $v[(j-1)d+k] = u_j[k]$  for any  $j \in [m]$  and  $k \in [d]$ .

► **Fact 19.** *Let  $u_1, u_2, \dots, u_m \in \mathbb{R}^d$  and  $v = [u_1 u_2 \dots u_m]$ . Then we have  $\|v\|_p^p = \sum_{i \in [m]} \|u_i\|_p^p$ .*

## 4.2 Cordinate gadgets

In this section, our aim is to construct points  $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$  for  $i \in \{0, 1\}$  such that the distance  $\|\mathbf{C}_i - P_{\mathbf{A}_j \mathbf{B}_k}(0)\|_p$  only depends on whether the bits  $i, j, k \in \{0, 1\}$  seen as coordinates of vectors are orthogonal. In other words, the points  $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$  form a *cordinate gadget*. Formally we will prove the following lemma,

► **Lemma 20.** For any  $p \neq 2$

$$\|C_i - P_{A_j B_k}(0)\|_p^p = \begin{cases} \beta_1 & \text{if } i = 1, j = 1, k = 1 \\ \beta_2 & \text{otherwise} \end{cases}$$

where  $\beta_1 < \beta_2$ .

In Section 4.3 we will use this lemma to construct the final point sets. Let  $\theta_1, \theta_2, \theta_3, \theta_4$  and  $\theta_5$  be positive constants. We construct the points  $A_1, B_1, C_1$  and  $A_0, B_0, C_0$  in  $\mathbb{R}^9$ .

$$\begin{aligned} A_0 &= [ -\theta_1, & 0, & -\theta_2, & 0, & \theta_3, & 2\theta_3, & \theta_4, & -2\theta_4, & 0 ] \\ A_1 &= [ \theta_1, & 2\theta_1, & \theta_2, & -2\theta_2, & -\theta_3, & 0, & -\theta_4, & 0, & 0 ] \\ B_0 &= [ -\theta_1, & 0, & \theta_2, & 2\theta_2, & \theta_3, & -2\theta_3, & -\theta_4, & 0, & 0 ] \\ B_1 &= [ \theta_1, & -2\theta_1, & -\theta_2, & 0, & -\theta_3, & 0, & \theta_4, & 2\theta_4, & 0 ] \\ C_0 &= [ 0, & 0, & 0, & 0, & 0, & 0, & 0, & 0, & \theta_5 ] \\ C_1 &= [ -\theta_1, & 0, & -\theta_2, & 0, & -\theta_3, & 0, & -\theta_4, & 0, & 0 ] \end{aligned}$$

From these points we can compute the points  $P_{A_i B_j}(0)$  for all  $i, j \in \{0, 1\}$ .

$$\begin{aligned} P_{A_0 B_0}(0) &= [ -\theta_1, & 0, & 0, & \theta_2, & \theta_3, & 0, & 0, & -\theta_4, & 0 ] \\ P_{A_1 B_0}(0) &= [ 0, & \theta_1, & \theta_2, & 0, & 0, & -\theta_3, & -\theta_4, & 0, & 0 ] \\ P_{A_1 B_1}(0) &= [ \theta_1, & 0, & 0, & -\theta_2, & -\theta_3, & 0, & 0, & \theta_4, & 0 ] \\ P_{A_0 B_1}(0) &= [ 0, & -\theta_1, & -\theta_2, & 0, & 0, & \theta_3, & \theta_4, & 0, & 0 ] \end{aligned}$$

Observe that  $\|C_0 - P_{A_i B_j}(0)\|_p^p = \sum_{r \in [5]} \theta_r^p$  for all  $i, j \in \{0, 1\}$ . Thus all the points  $P_{A_i B_j}(0)$  are equidistant from  $C_0$  irrespective of the exact values of  $\theta_r$  for  $r \in [5]$ . Note that when  $\theta_r = \theta$  for all  $r \in [5]$ , then  $\|C_1 - P_{A_i B_j}(0)\|_p^p = 4\theta^p + 2^p \theta^{2p}$  for all  $i, j \in \{0, 1\}$ . Thus all the points  $P_{A_i B_j}(0)$  are equidistant from  $C_1$  when all the  $\theta_r$  are the same. We now determine  $\theta_r$  for  $r \in [5]$  such that all but one point in  $\{P_{A_i B_j}(0) | i, j \in \{0, 1\}\}$  are equidistant and far from  $C_1$ . More precisely,

$$\|C_1 - P_{A_i B_j}(0)\|_p^p = \begin{cases} \beta_1 & \text{if } i = 1, j = 1 \\ \beta_2 & \text{otherwise} \end{cases}$$

and  $\beta_1 < \beta_2$ . We first quantify the distances from  $\{C_0, C_1\}$  to each of the points in  $\{P_{A_j B_k}(0) | j, k \in \{0, 1\}\}$ .

► **Lemma 21.** We have

$$\|C_i - P_{A_j B_k}(0)\|_p^p = \begin{cases} \sum_{r \in [5]} \theta_r^p & \text{if } i = 0 \\ 2\theta_2^p + 2^p \theta_3^p + 2\theta_4^p & \text{if } i = 1, j = 0, k = 0 \\ 2\theta_1^p + 2^p \theta_2^p + 2\theta_3^p & \text{if } i = 1, j = 1, k = 0 \\ 2\theta_1^p + 2\theta_3^p + 2^p \theta_4^p & \text{if } i = 1, j = 0, k = 1 \\ 2^p \theta_1^p + 2\theta_2^p + 2\theta_4^p & \text{if } i = 1, j = 1, k = 1 \end{cases}$$

We set the values of  $\theta_r$  for  $r \in [5]$  depending on  $p$ . When  $1 \leq p < 2$  we set

$$\theta_1 = (2^{p-1} - 1)^{\frac{1}{p}}, \theta_2 = 0, \theta_3 = 1, \theta_4 = 0, \theta_5 = 2^{\frac{p-1}{p}}$$

Substituting these values of  $\theta_r$  for all  $r \in [5]$  in Lemma 21 we make the following observation.



## 18:14 Polyline Simplification has Cubic Complexity

► **Observation 22.** *When  $1 \leq p < 2$ , then*

$$\|\mathbf{C}_i - P_{\mathbf{A}_j \mathbf{B}_k}(0)\|_p^p = \begin{cases} 2^p(2^{p-1} - 1) & \text{if } i = 1, j = 1, k = 1 \\ 2^p & \text{otherwise} \end{cases}$$

In case  $p > 2$ . Then we set

$$\theta_1 = 0, \theta_2 = (2^p - 2)^{\frac{1}{p}}, \theta_3 = (2^p - 4)^{\frac{1}{p}}, \theta_4 = (2^p - 2)^{\frac{1}{p}}, \theta_5 = (2^{2p} - 3 \cdot 2^p)^{\frac{1}{p}}$$

Substituting these values of  $\theta_r$  for all  $r \in [5]$  in Lemma 21 we make the following observation.

► **Observation 23.** *When  $p > 2$ , then*

$$\|\mathbf{C}_i - P_{\mathbf{A}_j \mathbf{B}_k}(0)\|_p^p = \begin{cases} 2^{p+2} - 8 & \text{if } i = 1, j = 1, k = 1 \\ 2^{2p} - 8 & \text{otherwise} \end{cases}$$

Combining Observations 22 and 23 we arrive at Lemma 20.

### 4.3 Vector gadgets

For every  $a \in A$ ,  $b \in B$  and  $c \in C$  we introduce vectors  $a', b', c'$  and  $a''b'', c''$  and then concatenate the respective vectors to form  $\tilde{a}$ ,  $\tilde{b}$  and  $\tilde{c}$  respectively. Intuitively  $a', b', c'$  help us to ensure properties  $\mathbf{P}_1$  and  $\mathbf{P}_2$ , while  $a'', b'', c''$  help us ensure the other properties.

#### 4.3.1 The vectors $a'$ , $b'$ , $c'$ , and $s'$

We construct the vector  $s'$  and the vectors  $a'$ ,  $b'$  and  $c'$  for every  $a \in A$ ,  $b \in B$  and  $c \in C$  respectively, in  $\mathbb{R}^{9d}$  as follows:

$$\begin{aligned} a' &= [\mathbf{A}_{a[1]}, \mathbf{A}_{a[2]}, \dots, \mathbf{A}_{a[d]}] & b' &= [\mathbf{B}_{b[1]}, \mathbf{B}_{b[2]}, \dots, \mathbf{B}_{b[d]}] \\ c' &= [\mathbf{C}_{c[1]}, \mathbf{C}_{c[2]}, \dots, \mathbf{C}_{c[d]}] & s' &= [0, 0, \dots, 0] \end{aligned}$$

We also define the sets  $A' = \{a' \mid a \in A\}$ ,  $B' = \{b' \mid b \in B\}$  and  $C' = \{c' \mid c \in C\}$ . We now make a technical observation about the vectors in  $A'$ ,  $B'$ , and  $C'$ , that will be useful later. We set  $\eta_1 = \max_{i \in [5]} \theta_i$ .

► **Observation 24.** *For any  $x, y \in A' \cup B' \cup C'$ , we have  $\|x - y\|_p \leq \eta_2$  where  $\eta_2 := 36d\eta_1$ .*

**Proof Sketch.** Note that the absolute value of every coordinate of the vectors  $a'$ ,  $b'$ , and  $c'$ , is bounded by  $2\eta_1$  (Since every coordinate is of the form  $\pm\theta_r$  or  $\pm 2\theta_r$  or 0). Combined with the fact that the total number of coordinates is  $9d$ , this immediately implies the observation. ◀

Note that  $a \in A$ ,  $b \in B$  and  $c \in C$  are non orthogonal if and only if  $\#_{111}^{c,a,b} > 0$  where  $\#_{111}^{c,a,b} = |\{i \mid i \in [d], a[i] = b[i] = c[i] = 1\}|$ . The following lemma shows a connection between non-orthogonality and small distance  $\|c' - P_{a'b'}(0)\|_p$ .

► **Lemma 25.** *For any  $a \in A$ ,  $b \in B$  and  $c \in C$ ,  $\|c' - P_{a'b'}(0)\|_p^p = d\beta_2 - (\beta_2 - \beta_1)\#_{111}^{c,a,b}$ .*

**Proof.** By Observation 19 we have

$$\begin{aligned} \|c' - P_{a'b'}(0)\|_p^p &= \sum_{\ell \in [d]} \|\mathbf{C}_{c[\ell]} - P_{\mathbf{A}_{a[\ell]} \mathbf{B}_{b[\ell]}}(0)\|_p^p \\ &= \beta_2(d - \#_{111}^{c,a,b}) + \beta_1 \#_{111}^{c,a,b} \quad (\text{by Lemma 20}) \\ &= d\beta_2 - (\beta_2 - \beta_1)\#_{111}^{c,a,b}. \end{aligned}$$

◀

### 4.3.2 The vectors $a'', b'', c''$ , and $s''$

We construct the vector  $s''$  and the vectors  $a'', b''$ , and  $c''$  for every  $a \in A, b \in B$ , and  $c \in C$ , respectively, in  $\mathbb{R}^3$  as follows:

$$a'' = [\gamma_1, 0, 0] \quad b'' = [\gamma_1, \gamma_2, 0] \quad c'' = [0, \frac{\gamma_2}{2}, 0] \quad s'' = [0, \frac{\gamma_2}{2}, \gamma_2]$$

where  $\gamma_1, \gamma_2$  are positive constants. We are now ready to define the final points of our construction,  $s$  and  $\tilde{a}, \tilde{b}$  and  $\tilde{c}$  for any  $a \in A, b \in B$  and  $c \in C$ , respectively.

$$\tilde{a} = [a', a''] \quad \tilde{b} = [b', b''] \quad \tilde{c} = [c', c''] \quad s = [s', s'']$$

We set

$$\gamma_1 = \eta_1, \delta = (\gamma_1^p + d\beta_2 - (\beta_2 - \beta_1))^{\frac{1}{p}}, \gamma_2 = \max \left( 4\delta, \eta_2 \left( 1 + \frac{(\gamma_1^p + d\beta_2)^{\frac{1}{p}}}{(\gamma_1^p + d\beta_2)^{\frac{1}{p}} - \delta} \right) \right)$$

Note that we have constructed the point sets  $\tilde{A}, \tilde{B}, \tilde{C}$ , and the point  $s$  and determined  $\delta$  in total time  $\mathcal{O}(nd)$ . Therefore now it suffices to show that our point set and  $\delta$  satisfy the properties  $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4, \mathbf{P}_5$ , and  $\mathbf{P}_6$ . To this end, we first show how the distance  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(\alpha)\|_p$  is related with  $\#_{111}^{c,a,b}$  (the non-orthogonality of the vectors  $a, b$ , and  $c$ ).

- **Lemma 26.** *For any  $a \in A, b \in B$  and  $c \in C$  we have,*
- $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p^p = \gamma_1^p + \beta_2 d - (\beta_2 - \beta_1) \#_{111}^{c,a,b}$ .
  - If  $\#_{111}^{c,a,b} = 0$  then  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(\alpha)\|_p^p > \delta$  for all  $\alpha \in [-\frac{1}{2}, \frac{1}{2}]$ .

**Proof Sketch.** Note that  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p = \|c' - P_{a'b'}(0)\|_p + \|c'' - P_{a''b''}(0)\|_p$ . Using Lemma 25 and substituting vectors  $a'', b''$  and  $c''$  we arrive at  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p^p = \gamma_1^p + \beta_2 d - (\beta_2 - \beta_1) \#_{111}^{c,a,b}$ . Since we choose our  $\gamma_2$  to be sufficiently large the closest point on the line-segment  $\tilde{a}\tilde{b}$  to  $\tilde{c}$  is sufficiently near  $P_{\tilde{a}\tilde{b}}(0)$ . Thus when  $\#_{111}^{c,a,b} = 0$ , we have  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p^p = \gamma_1^p + \beta_2 d > \delta$  and no point sufficiently near  $P_{\tilde{a}\tilde{b}}(0)$  has distance smaller than  $\delta$  to  $\tilde{c}$ . ◀

We now show that the properties  $\mathbf{P}_1$  and  $\mathbf{P}_2$  hold. The first result of Lemma 26 implies that for any  $a \in A, b \in B$  and  $c \in C$  we have  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p \leq \delta$  if and only if  $\#_{111}^{c,a,b} \geq 1$ , or equivalently if  $\sum_{\ell \in [d]} a[\ell] \cdot b[\ell] \cdot c[\ell] \neq 0$ . By the second result of Lemma 26, it follows that for any  $\alpha \in [-\frac{1}{2}, \frac{1}{2}]$  if  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(\alpha)\|_p^p \leq \delta$ , then  $\#_{111}^{c,a,b} = 0$  which implies  $\|\tilde{c} - P_{\tilde{a}\tilde{b}}(0)\|_p^p \leq \delta$ . The remaining properties are guaranteed primarily by the component vectors  $a'', b''$  and  $c''$  of  $\tilde{a}, \tilde{b}$  and  $\tilde{c}$ , and the detailed proof is in the full version of the paper.

---

### References

- 1 Mohammad Ali Abam, Mark de Berg, Peter Hachenberger, and Alireza Zarei. Streaming Algorithms for Line Simplification. *Discrete & Computational Geometry*, 43(3):497–515, 2010.
- 2 Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *SODA*, pages 377–391. SIAM, 2016.
- 3 Pankaj K. Agarwal, Sariel Har-Peled, Nabil H. Mustafa, and Yusu Wang. Near-Linear Time Approximation Algorithms for Curve Simplification. *Algorithmica*, 42(3-4):203–219, 2005.
- 4 Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk. Matching planar maps. In *SODA*, pages 589–598. ACM/SIAM, 2003.
- 5 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5(1–2):78–99, 1995.


## 18:16 Polyline Simplification has Cubic Complexity

- 6 Gill Barequet, Danny Z. Chen, Ovidiu Daescu, Michael T. Goodrich, and Jack Snoeyink. Efficiently Approximating Polygonal Paths in Three and Higher Dimensions. *Algorithmica*, 33(2):150–167, 2002.
- 7 Kevin Buchin, Maike Buchin, Maximilian Konzack, Wolfgang Mulzer, and André Schulz. Fine-grained analysis of problems on curves. *EuroCG, Lugano, Switzerland*, 2016.
- 8 W.S. Chan and F. Chin. APPROXIMATION OF POLYGONAL CURVES WITH MINIMUM NUMBER OF LINE SEGMENTS OR MINIMUM ERROR. *International Journal of Computational Geometry & Applications*, 06(01):59–77, 1996.
- 9 Danny Z. Chen, Ovidiu Daescu, John Hershberger, Peter M. Kogge, Ningfang Mi, and Jack Snoeyink. Polygonal path simplification with angle constraints. *Comput. Geom.*, 32(3):173–187, 2005.
- 10 Mark de Berg, Marc van Kreveld, and Stefan Schirra. Topologically Correct Subdivision Simplification Using the Bandwidth Criterion. *Cartography and Geographic Information Systems*, 25(4):243–257, 1998.
- 11 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973.
- 12 Regina Estkowski and Joseph S. B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. In *SoCG*, pages 40–49. ACM, 2001.
- 13 Stefan Funke, Thomas Mendel, Alexander Miller, Sabine Storandt, and Maria Wiebe. Map Simplification with Topology Constraints: Exactly and in Practice. In *ALLENEX*, pages 185–196. SIAM, 2017.
- 14 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and R. Ryan Williams. Completeness for First-Order Properties on Sparse Structures with Algorithmic Applications. In *SODA*, pages 2162–2181. SIAM, 2017.
- 15 Michael Godau. A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. In *STACS*, pages 127–136, 1991.
- 16 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating Polygons and Subdivisions with Minimum Link Paths. *Int. J. Comput. Geometry Appl.*, 3(4):383–415, 1993.
- 17 John Hershberger and Jack Snoeyink. An  $O(n \log n)$  Implementation of the Douglas-Peucker Algorithm for Line Simplification. In *SoCG*, pages 383–384. ACM, 1994.
- 18 Hiroshi Imai and Masao Iri. Polygonal Approximations of a Curve — Formulations and Algorithms. In *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 71–86. North-Holland, 1988.
- 19 Avraham Melkman and Joseph O’Rourke. On Polygonal Chain Approximation. In *Computational Morphology*, volume 6 of *Machine Intelligence and Pattern Recognition*, pages 87–95. North-Holland, 1988.
- 20 Marc J. van Kreveld, Maarten Löffler, and Lionov Wiratma. On Optimal Polyline Simplification Using the Hausdorff and Fréchet Distance. In *SoCG*, volume 99 of *LIPICs*, pages 56:1–56:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
- 21 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM*, 2018.

# A Spanner for the Day After

Kevin Buchin 

Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands  
k.a.buchin@tue.nl

Sariel Har-Peled 

Department of Computer Science, University of Illinois at Urbana-Champaign, USA  
sariel@illinois.edu

Dániel Oláh

Department of Mathematics and Computing Science, TU Eindhoven, The Netherlands  
d.olah@tue.nl

---

## Abstract

---

We show how to construct  $(1 + \varepsilon)$ -spanner over a set  $P$  of  $n$  points in  $\mathbb{R}^d$  that is resilient to a catastrophic failure of nodes. Specifically, for prescribed parameters  $\vartheta, \varepsilon \in (0, 1)$ , the computed spanner  $G$  has  $\mathcal{O}(\varepsilon^{-7d} \log^7 \varepsilon^{-1} \cdot \vartheta^{-6} n \log n (\log \log n)^6)$  edges. Furthermore, for *any*  $k$ , and *any* deleted set  $B \subseteq P$  of  $k$  points, the residual graph  $G \setminus B$  is  $(1 + \varepsilon)$ -spanner for all the points of  $P$  except for  $(1 + \vartheta)k$  of them. No previous constructions, beyond the trivial clique with  $\mathcal{O}(n^2)$  edges, were known such that only a tiny additional fraction (i.e.,  $\vartheta$ ) lose their distance preserving connectivity.

Our construction works by first solving the exact problem in one dimension, and then showing a surprisingly simple and elegant construction in higher dimensions, that uses the one dimensional construction in a black box fashion.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Geometric spanners, vertex failures, robustness

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.19

**Related Version** A full version of the paper is available at <http://arxiv.org/abs/1811.06898>.

**Funding** *Sariel Har-Peled*: Work on this paper was partially supported by a NSF AF awards CCF-1421231.

*Dániel Oláh*: Supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.

## 1 Introduction

**Spanners.** A Euclidean graph is a graph whose vertices are points in  $\mathbb{R}^d$  and the edges are weighted by the Euclidean distance between their endpoints. Let  $G = (P, E)$  be a Euclidean graph and  $p, q \in P$  be two vertices of  $G$ . For a parameter  $t \geq 1$ , a path between  $p$  and  $q$  in  $G$  is a  **$t$ -path** if the length of the path is at most  $t \|p - q\|$ , where  $\|p - q\|$  is the Euclidean distance between  $p$  and  $q$ . The graph  $G$  is a  **$t$ -spanner** of  $P$  if there is a  $t$ -path between any pair of points  $p, q \in P$ . Throughout the paper,  $n$  denotes the cardinality of the point set  $P$ , unless stated otherwise. We denote the length of the shortest path between  $p, q \in P$  in the graph  $G$  by  $d(p, q)$ .

Spanners have been studied extensively. The main goal in spanner constructions is to have small *size*, that is, to use as few edges as possible. Other desirable properties are low degrees [2, 10, 18], low weight [6, 12], low diameter [3, 4] or to be resistant against failures. The book by Narasimhan and Smid [17] gives a comprehensive overview of spanners.



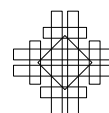
© Kevin Buchin, Sariel Har-Peled, and Dániel Oláh;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 19; pp. 19:1–19:15

Leibniz International Proceedings in Informatics



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



**Robustness.** In this paper, our goal is to construct spanners that are robust according to the notion introduced by Bose *et al.* [7]. Intuitively, a spanner is robust if the deletion of  $k$  vertices only harms a few other vertices. Formally, a graph  $G$  is an  $f(k)$ -robust  $t$ -spanner, for some positive monotone function  $f$ , if for any set  $B$  of  $k$  vertices deleted in the graph, the remaining graph  $G \setminus B$  is still a  $t$ -spanner for at least  $n - f(k)$  of the vertices. Note, that the graph  $G \setminus B$  has  $n - k$  vertices – namely, there are at most  $\mathcal{L}(k) = f(k) - k$  additional vertices that no longer have good connectivity to the remaining graph. The quantity  $\mathcal{L}(k)$  is the *loss*. We are interested in minimizing the loss.

The natural question is how many edges are needed to achieve a certain robustness (since the clique has the desired property). That is, for a given parameter  $t$  and function  $f$ , what is the minimal size that is needed to obtain an  $f(k)$ -robust  $t$ -spanner on any set of  $n$  points.

A priori it is not clear that such a sparse graph should exist (for  $t$  a constant) for a point set in  $\mathbb{R}^d$ , since the robustness property looks quite strong. Surprisingly, Bose *et al.* [7] showed that one can construct a  $\mathcal{O}(k^2)$ -robust  $\mathcal{O}(1)$ -spanner with  $\mathcal{O}(n \log n)$  edges. Bose *et al.* [7] proved various other bounds in the same vein on the size for one-dimensional and higher-dimensional point sets. Their most closely related result is that for the one-dimensional point set  $P = \{1, 2, \dots, n\}$  and for any  $t \geq 1$  at least  $\Omega(n \log n)$  edges are needed to construct an  $\mathcal{O}(k)$ -robust  $t$ -spanner.

An open problem left by Bose *et al.* [7] is the construction of  $\mathcal{O}(k)$ -robust spanners – they only provide the easy upper bound of  $\mathcal{O}(n^2)$  for this case.

**$\vartheta$ -reliable spanners.** We are interested in building spanners where the loss is only fractional. Specifically, given a parameter  $\vartheta$ , we consider the function  $f(k) = (1 + \vartheta)k$ . The loss in this case is  $\mathcal{L}(k) = f(k) - k = \vartheta k$ . A  $(1 + \vartheta)k$ -robust  $t$ -spanner is  **$\vartheta$ -reliable  $t$ -spanner**.

**Exact reliable spanners.** If the input point set is in one dimension, then one can easily construct a 1-spanner for the points, which means that the exact distances between points on the line are preserved by the spanner. This of course can be done easily by connecting the points from left to right. It becomes significantly more challenging to construct such an exact spanner that is reliable.

**Fault tolerant spanners.** Robustness is not the only definition that captures the resistance of a spanner network against vertex failures. A closely related notion is fault tolerance [13, 14, 15]. A graph  $G = (P, E)$  is an  $r$ -fault tolerant  $t$ -spanner if for any set  $B$  of failed vertices with  $|B| \leq r$ , the graph  $G \setminus B$  is still a  $t$ -spanner. The disadvantage of  $r$ -fault tolerance is that each vertex must have degree at least  $r + 1$ , otherwise the vertex can be isolated by deleting its neighbors. Therefore, the graph has size at least  $\Omega(rn)$ . There are constructions that show  $\mathcal{O}(rn)$  edges are enough to build  $r$ -fault tolerant spanners. However, depending on the chosen value  $r$  the size can be too large.

In particular, fault tolerant spanners cannot have a near-linear number of edges, and still withstand a widespread failure of nodes. Specifically, if a fault tolerant spanner has  $m$  edges, then it can withstand a failure of at most  $2m/n$  vertices. In sharp contrast,  $\vartheta$ -reliable spanners can withstand a widespread failure. Indeed, a  $\vartheta$ -reliable spanner can withstand a failure of close to  $n/(1 + \vartheta)$  of its vertices, and still have some vertices that are connected by short paths in the remaining graph.

## 1.1 Our results

In this paper, we investigate how to construct reliable spanners with very small loss – that is  $\vartheta$ -reliable spanners. To the best of our knowledge nothing was known on this case before this work. For omitted proofs we refer the reader to the full version of the paper [8].

- (a) **Expanders are reliable.** Intuitively, a constant degree expander is a robust/reliable graph under a weaker notion of robustness – that is, connectivity. As such, for a parameter  $\vartheta > 0$ , we show that constant degree expanders are indeed  $\vartheta$ -reliable in the sense that all except a small fraction of the points stay connected. Formally, one can build a graph  $G$  with  $\mathcal{O}(\vartheta^{-3}n)$  edges, such that for any failure set  $B$  of  $k$  vertices, the graph  $G \setminus B$  has a connected component of size at least  $n - (1 + \vartheta)k$ . We emphasize, however, that distances are not being preserved in this case. See Lemma 6 for the result.
- (b) **Exact  $\mathcal{O}(1)$ -reliable spanner in one dimension.** Inspired by the reliability of constant degree expanders, we show how to construct an  $\mathcal{O}(1)$ -reliable exact spanner on any one-dimensional set of  $n$  points with  $\mathcal{O}(n \log n)$  edges.<sup>1</sup> The idea of the construction is to build a binary tree over the points, and to build bipartite expanders between certain subsets of nodes in the same layer. One can think of this construction as building different layers of expanders for different resolutions. The construction is described in Section 3.2. See Theorem 12 for the result.
- (c) **Exact  $\vartheta$ -reliable spanner in one dimension.** One can get added redundancy by systematically shifting the layers. Done carefully, this results in a  $\vartheta$ -reliable exact spanner. The construction is described in Section 3.3. See Theorem 13 for the result.
- (d)  **$\vartheta$ -reliable  $(1 + \varepsilon)$ -spanners in higher dimensions.** We next show a *surprisingly simple and elegant* construction of  $\vartheta$ -reliable spanners in two and higher dimensions, using a recent result of Chan *et al.* [11], which show that one needs to maintain only a “few” linear orders. This immediately reduces the  $d$  dimensional problem to maintaining a reliable spanner for each of this orderings, which is the problem we already solved. See Section 4 for details.
- (e)  **$\vartheta$ -reliable  $(1 + \varepsilon)$ -spanner in  $\mathbb{R}^d$  with bounded spread.** Since both general constructions in  $\mathbb{R}^d$  have some additional polylog factors that seems unnecessary, we present a better construction for the bounded spread case. Specifically, for points with spread  $\Phi$  in  $\mathbb{R}^d$ , and for any  $\varepsilon > 0$ , we construct a  $\vartheta$ -reliable  $(1 + \varepsilon)$ -spanner with  $\mathcal{O}(\varepsilon^{-d}\vartheta^{-2}n \log \Phi)$  edges. The basic idea is to construct a well-separated pair decomposition (WSPD) directly on the quadtree of the point set, and convert every pair in the WSPD into a reliable graph using a bipartite expander. The union of these graphs is the required reliable spanner. See Section 5 and Lemma 21 for details.

**Shadow.** Underlying our construction is the notion of identifying the points that loose connectivity when the failure set is removed. Intuitively, a point is in the shadow if it is surrounded by failed points. We believe that this concept is of independent interest – see Section 3.1 for details and relevant results in one dimension and the full version [8] for an additional result in higher dimensions.

Independently, Bose *et al.* [5] also obtained an upper bound on the size of reliable spanners in  $\mathbb{R}^d$ . Their construction has  $\mathcal{O}(n \log^2 n \log \log n)$  edges, which is close to our bound of  $\mathcal{O}(n \log n (\log \log n)^6)$  edges.

---

<sup>1</sup> This also improves an earlier preliminary construction by (some of) the authors [arXiv:1803.08719](https://arxiv.org/abs/1803.08719).

## 2 Preliminaries

### 2.1 Problem definition and notations

Let  $[n]$  denote the set  $\{1, 2, \dots, n\}$  and let  $[i : j] = \{i, i + 1, \dots, j\}$ .

► **Definition 1** (Robust spanner). *Let  $G = (P, E)$  be a  $t$ -spanner for some  $t \geq 1$  and let  $f: \mathbb{N} \rightarrow \mathbb{R}_+$ , and two point sets  $P_1, P_2 \subseteq P$ . The graph  $G$  is an  **$f(k)$ -robust  $t$ -spanner for  $P_1 \oplus P_2$**  if for any set of (failed) vertices  $B \subseteq P$  there exists a set  $B^+ \supseteq B$  with  $|B^+| \leq f(|B|)$  such that the subgraph*

$$G \setminus B = G_{P \setminus B} = \left( P \setminus B, \{uv \in E(G) \mid u, v \in P \setminus B\} \right)$$

*induced by  $P \setminus B$  is a  $t$ -spanner for  $(P_1 \setminus B^+) \oplus (P_2 \setminus B^+)$ . That is,  $G \setminus B$  has a  $t$ -path between all pairs of points  $p \in P_1 \setminus B^+$  and  $q \in P_2 \setminus B^+$ . If  $P_1 = P_2 = P$ , then  $G$  is a  **$f(k)$ -robust  $t$ -spanner**.*

*The vertices of  $B^+ \setminus B$  are the vertices **harmed** by  $B$ , and the quantity  $\mathcal{L}(k) = f(k) - k \geq |B^+| - |B|$  is the **loss**.*

► **Definition 2.** *For a parameter  $\vartheta > 0$ , a graph  $G$  that is  $(1 + \vartheta)k$ -robust  $t$ -spanner is a  **$\vartheta$ -reliable  $t$ -spanner**.*

► **Definition 3.** *For a number  $x > 0$ , let  $\text{pow}_2(x) = 2^{\lceil \log x \rceil}$  be the smallest number that is a power of 2 and is at least as large as  $x$ .*

### 2.2 Expander construction

For a set  $X$  of vertices in a graph  $G = (V, E)$ , let  $\Gamma(X) = \{v \in V \mid uv \in E \text{ for a } u \in X\}$  be the **neighbors** of  $X$  in  $G$ . The following lemma, which is a standard expander construction (see e.g. [16, Section 5.3]), provides the main building block of our one-dimensional construction.

► **Lemma 4.** *Let  $L, R$  be two disjoint sets, with a total of  $n$  elements, and let  $\xi \in (0, 1)$  be a parameter. One can build a bipartite graph  $G = (L \cup R, E)$  with  $\mathcal{O}(n/\xi^2)$  edges, such that*

- (i) *for any subset  $X \subseteq L$ , with  $|X| \geq \xi|L|$ , we have that  $|\Gamma(X)| > (1 - \xi)|R|$ , and*
- (ii) *for any subset  $Y \subseteq R$ , with  $|Y| \geq \xi|R|$ , we have that  $|\Gamma(Y)| > (1 - \xi)|L|$ .*

### 2.3 Expanders are reliable

Let  $P$  be a set with  $n$  elements, and let  $\vartheta \in (0, 1)$  be a parameter. We next build a constant degree expander graph on  $P$  and show that it is  $\vartheta$ -reliable. The following two lemmas are not surprising if one is familiar with expanders and their properties.

► **Lemma 5.** *Let  $n$  be a positive integer number, let  $\alpha > 1$  be an integer constant, and let  $\beta \in (0, 1)$  be some constant. One can build a graph  $G = ([n], E)$ , such that for all sets  $X \subset V$ , we have that  $|\Gamma(X)| \geq \min((1 - \beta)n, \alpha|X|)$ . The graph  $G$  has  $\mathcal{O}((\alpha/\beta)n)$  edges.*

► **Lemma 6.** *Let  $n$  and  $\vartheta \in (0, 1/2)$  be parameters. One can build a graph  $G = ([n], E)$  with  $\mathcal{O}(\vartheta^{-3}n)$  edges, such that for any set  $B \subseteq [n]$ , we have that  $G \setminus B$  has a connected component of size at least  $n - (1 + \vartheta)|B|$ . That is, the graph  $G$  is  $\vartheta$ -reliable.*



### 3 Building reliable spanners in one dimension

#### 3.1 Bounding the size of the shadow

Our purpose is to build a reliable 1-spanner in one dimension. Intuitively, a point in  $[n]$  is in trouble, if many of its close by neighbors belong to the failure set  $B$ . Such an element is in the shadow of  $B$ , defined formally next.

► **Definition 7.** Consider an arbitrary set  $B \subseteq [n]$  and a parameter  $\alpha \in (0, 1)$ . A number  $i$  is in the **left  $\alpha$ -shadow** of  $B$ , if and only if there exists an integer  $j \geq i$ , such that  $|[i : j] \cap B| \geq \alpha |[i : j]|$ . Similarly,  $i$  is in the **right  $\alpha$ -shadow** of  $B$ , if and only if there exists an integer  $h \leq i$  and  $|[h : i] \cap B| \geq \alpha |[h : i]|$ . The left and right  $\alpha$ -shadow of  $B$  is denoted by  $\mathcal{S}_{\rightarrow}(B)$  and  $\mathcal{S}_{\leftarrow}(B)$ , respectively. The combined shadow is denoted by  $\mathcal{S}(\alpha, B) = \mathcal{S}_{\rightarrow}(B) \cup \mathcal{S}_{\leftarrow}(B)$ .

► **Lemma 8.** Fix a set  $B \subseteq [n]$  and let  $\alpha \in (0, 1)$  be a parameter. Then, we have that  $|\mathcal{S}_{\rightarrow}(B)| \leq (1 + \lceil 1/\alpha \rceil) |B|$ . In particular, the size of  $\mathcal{S}(\alpha, B)$  is at most  $2(1 + \lceil 1/\alpha \rceil) |B|$ .

Lemma 8 is somewhat restrictive because the shadow is at least twice larger than the failure set  $B$ . Intuitively, as  $\alpha \rightarrow 1$ , the shadow should converge to  $B$ . The following lemma, which is a variant of Lemma 8 quantify this.

► **Lemma 9.** Fix a set  $B \subseteq [n]$ , let  $\alpha \in (2/3, 1)$  be a parameter, and let  $\mathcal{S}(\alpha, B)$  be the set of elements in the  $\alpha$ -shadow of  $B$ . We have that  $|\mathcal{S}(\alpha, B)| \leq |B| / (2\alpha - 1)$ .

**Proof.** Let  $c = 1 - 1/\alpha < 0$ . For  $i = 1, \dots, n$ , let  $x_i = c$  if  $i \in B$ , and  $x_i = 1$  otherwise. For any interval  $I$  of length  $\Delta$ , with  $\tau\Delta$  elements in  $B$ , such that  $x(I) = \sum_{i \in I} x_i \leq 0$ , we have that

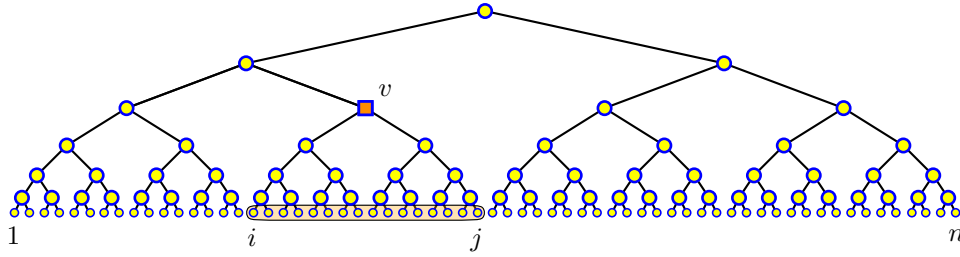
$$\begin{aligned} x(I) \leq 0 &\iff (1 - \tau)\Delta + c\tau\Delta \leq 0 \iff 1 - \tau \leq -\tau c \iff 1/\tau \leq 1 - c \\ &\iff 1/\tau \leq 1 - (1 - 1/\alpha) \iff 1/\tau \leq 1/\alpha \iff \tau \geq \alpha. \end{aligned}$$

An element  $j \in [n]$  is in the left  $\alpha$ -shadow of  $B$  if and only if there exists an integer  $j'$ , such that  $|[j : j'] \cap B| \geq \alpha |[j : j']|$  and, by the above,  $x([j : j']) \leq 0$ . Namely, an integer  $j$  in the left  $\alpha$ -shadow of  $B$  corresponds to some prefix sum of the  $x_i$ s that starts at  $j$  and add up to some non-positive sum. From this point on, we work with the sequence of numbers  $x_1, \dots, x_n$ , using the above summation criterion to detect the elements in the left  $\alpha$ -shadow.

For a location  $j \in [n]$  that is in the left  $\alpha$ -shadow, let  $W_j = [j : j']$  be the *witness interval* for  $j$  – this is the shortest interval that has a non-positive sum that starts at  $j$ . Let  $I = W_k = [k : k']$  be the shortest witness interval, for any number in  $\mathcal{S}(\alpha, B) \setminus B$ . For any  $j \in [k + 1 : k']$ , we have  $x([k : j - 1]) + x([j : k']) = x([k : k']) \leq 0$ . Thus, if  $x_j = 1$ , this implies that either  $j$  or  $k$  have shorter witness intervals than  $I$ , which is a contradiction to the choice of  $k$ . We conclude that  $x_j < 0$  for all  $j \in [k + 1 : k']$ , that is,  $[k + 1 : k'] \subseteq B$ .

Letting  $\ell = |I| = k' - k + 1$ , we have that  $(\ell - 1)/\ell \geq \alpha \iff \ell - 1 \geq \alpha\ell \iff \ell \geq 1/(1 - \alpha) \iff \ell \geq \lceil 1/(1 - \alpha) \rceil \geq 3$ , as  $\alpha \geq 2/3$ . In particular, by the minimality of  $I$ , it follows that  $\ell = \lceil 1/(1 - \alpha) \rceil$ .

Let  $J = [k : k' - 1] \subset I$ . We have that  $x(J) > 0$ . For any  $j \in \mathcal{S}(\alpha, B) \setminus B$ , such that  $j \neq k$ , consider the witness interval  $W_j$ . If  $j > k$ , then  $j > k'$ , as all the elements of  $I$ , except  $k$ , are in  $B$ . If  $j < k$  and  $j' \in J$ , then  $\tau = x([k : j']) > 0$ , which implies that  $x([j : k - 1]) = x(W_j) - \tau < 0$ , but this is a contradiction to the definition of  $W_j$ . Namely, all the witness intervals either avoids  $J$ , or contain it in their interior. Given a witness interval  $W_j$ , such that  $J \subset W_j$ , we have  $x(W_j \setminus J) = x(W_j) - x(J) < x(W_j) \leq 0$ , since  $x(J) > 0$ .



■ **Figure 1** The binary tree built over  $[n]$ . The block of node  $v$  is the interval  $[i : j]$ .

So consider the new sequence of numbers  $x_{[n]\setminus J} = x_1, \dots, x_{k-1}, x_{k'}, \dots, x_n$  resulting from removing the elements that corresponds to  $J$  from the sequence. Reclassify which elements are in the left shadow in the new sequence. By the above, any element that was in the shadow before, is going to be in the new shadow. As such, one can charge the element  $k$ , that is in the left shadow (but not in  $B$ ), to all the other elements of  $J$  (that are all in  $B$ ). Applying this charging scheme inductively, charges all the elements in the left shadow (that are not in  $B$ ) to elements in  $B$ . We conclude that the number of elements in the left shadow of  $B$ , that are not in  $B$  is bounded by

$$\frac{|B|}{|J| - 1} = \frac{|B|}{\ell - 2} = \frac{|B|}{\lceil 1/(1 - \alpha) \rceil - 2} \leq \frac{1 - \alpha}{1 - 2(1 - \alpha)} |B| = \frac{1 - \alpha}{2\alpha - 1} |B|.$$

The above argument can be applied symmetrically to the right shadow. We conclude that

$$|\mathcal{S}(\alpha, B)| \leq |B| + 2 \frac{1 - \alpha}{2\alpha - 1} |B| = \frac{2\alpha - 1 + 2 - 2\alpha}{2\alpha - 1} |B| = \frac{|B|}{2\alpha - 1}. \quad \blacktriangleleft$$

### 3.2 Construction of $\mathcal{O}(1)$ -reliable exact spanners in one dimension

#### 3.2.1 Constructing the graph $H$

Assume  $n$  is a power of two, and consider building the natural full binary tree  $T$  with the numbers of  $[n]$  as the leaves. Every node  $v$  of  $T$  corresponds to an interval of numbers of the form  $[i : j]$  its canonical interval, which we refer to as the block of  $v$ , see Figure 1. Let  $\mathcal{I}$  be the resulting set of all blocks. In each level one can sort the blocks of the tree from left to right. Two adjacent blocks of the same level are neighbors. For a block  $I \in \mathcal{I}$ , let  $\text{next}(I)$  and  $\text{prev}(I)$  be the blocks (in the same level) directly to the right and left of  $I$ , respectively.

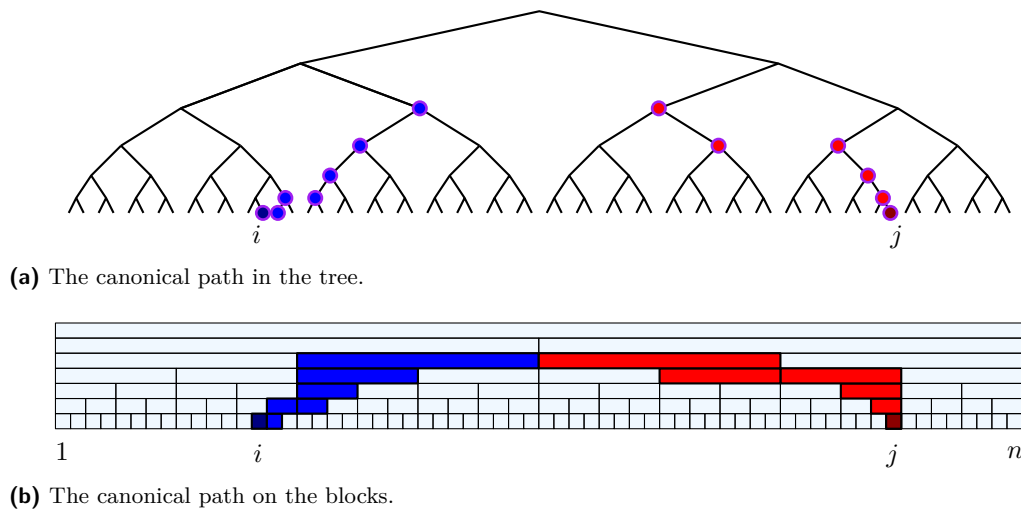
We build the graph of Lemma 4 with  $\xi = 1/16$  for any two neighboring blocks in  $\mathcal{I}$ . Let  $H$  be the resulting graph when taking the union over all the sets of edges generated by the above.

#### 3.2.2 Analysis

In the following we show that the resulting graph  $H$  is an  $\mathcal{O}(k)$ -robust 1-spanner and has  $\mathcal{O}(n \log n)$  edges. We start by verifying the size of the graph.

► **Lemma 10.** *The graph  $H$  has  $\mathcal{O}(n \log n)$  edges.*

**Proof.** Let  $h = \log n$  be the depth of the tree  $T$ . In each level  $i = 1, 2, \dots, h$  of  $T$  there are  $2^{h-i}$  nodes and the blocks of these nodes have size  $2^i$ . The number of pairs of adjacent blocks in level  $i$  is  $2^{h-i} - 1$  and each pair contributes  $\mathcal{O}(2^i)$  edges. Therefore, each level of  $T$  contributes  $\mathcal{O}(n)$  edges. We get  $\mathcal{O}(n \log n)$  for the overall size by summing up for all levels. ◀



**Figure 2** The canonical path between the vertices  $i$  and  $j$  in two different representations. The blue nodes and blocks correspond to the ascent part and the red nodes and blocks correspond to the descent part of the walk.

There is a natural path between two leaves in the tree  $T$ , described above, going through their lowest common ancestor. However, for our purposes we need something somewhat different – intuitively because we only want to move forward in the 1-path.

Given two numbers  $i$  and  $j$ , where  $i < j$ , consider the two blocks  $I, J \in \mathcal{I}$  that correspond to the two numbers at the bottom level. Set  $I_0 = I$ , and  $J_0 = J$ . We now describe a canonical walk from  $I$  to  $J$ , where initially  $\ell = 0$ . During the walk we have two active blocks  $I_\ell$  and  $J_\ell$ , that are both in the same level. For any block  $I \in \mathcal{I}$  we denote its parent by  $p(I)$ . At every iteration we bring the two active blocks closer to each other by moving up in the tree.

Specifically, repeatedly do the following:

- (a) If  $I_\ell$  and  $J_\ell$  are neighbors then the walk is done.
- (b) If  $I_\ell$  is the right child of  $p(I_\ell)$ , then set  $I_{\ell+1} = \text{next}(I_\ell)$  and  $J_{\ell+1} = J_\ell$ , and continue to the next iteration.
- (c) If  $J_\ell$  is the left child of  $p(J_\ell)$ , then set  $I_{\ell+1} = I_\ell$  and  $J_{\ell+1} = \text{prev}(J_\ell)$ , and continue to the next iteration.
- (d) Otherwise – the algorithm ascends. It sets  $I_{\ell+1} = p(I_\ell)$ , and  $J_{\ell+1} = p(J_\ell)$ , and it continues to the next iteration.

It is easy to verify that this walk is well defined, and let

$$\pi(i, j) \equiv \underbrace{I_0 \rightarrow I_1 \rightarrow \dots \rightarrow I_\ell}_{\text{ASCENT}} \rightarrow \underbrace{J_\ell \rightarrow \dots \rightarrow J_0}_{\text{DESCENT}}$$

be the resulting walk on the blocks where we removed repeated blocks. Figure 2 illustrates the path of blocks between two vertices  $i$  and  $j$ .

In the following, consider a fixed set  $B \subseteq [n]$  of faulty nodes. A block  $I \in \mathcal{I}$  is  $\alpha$ -contaminated, for some  $\alpha \in (0, 1)$ , if  $|I \cap B| \geq \alpha |I|$ .

► **Lemma 11.** Consider two nodes  $i, j \in [n]$ , with  $i < j$ , and let  $\pi(i, j)$  be the canonical path between  $i$  and  $j$ . If any block of  $\pi = \pi(i, j)$  is  $\alpha$ -contaminated, then  $i$  or  $j$  are in the  $\alpha/3$ -shadow of  $B$ .

**Proof.** Assume the contamination happens in the left half of the path, i.e., at some block  $I_t$ , during the ascent from  $i$  to the connecting block to the descent path into  $j$ . By construction, there could be only one block before  $I_t$  on the path of the same level, and all previous blocks are smaller, and there are at most two blocks at each level. Furthermore, for two consecutive  $I_j, I_{j+1}$  that are blocks of different levels,  $I_j \subseteq I_{j+1}$ . Thus we have that either  $i \in I_t$ , or  $i \in \text{prev}(I_t)$ , or  $i \in \text{prev}(\text{prev}(I_t))$ , since there are at most  $|I_t| + |I_t|/2 + \dots + 2 + 1 = 2|I_t| - 1$  vertices that are contained in the path before the block  $I_t$ . Notice that if  $i \in I_t$ , then it is the leftmost point of  $I_t$ .

In particular, let  $r$  be the maximum number in  $I_t$ , and observe  $|[i : r] \cap B| \geq \alpha |I_t| \geq (\alpha/3) |[i : r]|$ . Thus, the number  $i$  is the  $\alpha/3$ -shadow, as claimed.

The other case, when the contamination happens in the right part during the descent, is handled symmetrically.  $\blacktriangleleft$

► **Theorem 12.** *The graph  $H$  constructed above on the set  $[n]$  is an  $\mathcal{O}(1)$ -reliable exact spanner and has  $\mathcal{O}(n \log n)$  edges.*

**Proof.** The size is proved in Lemma 10. Let  $\alpha = 1/32$ . Let  $B^+$  be the set of vertices that are in the  $\alpha/3$ -shadow of  $B$ , that is,  $B^+ = \mathcal{S}(\alpha/3, B)$ . By Lemma 8 we have that  $|B^+| \leq 2(1 + \lceil 3/\alpha \rceil) |B| \leq 200 |B|$ .

Consider any two vertices  $i, j \in [n] \setminus B^+$ . Let  $\pi(i, j)$  be the canonical path between  $i$  and  $j$ . None of the blocks in this path are  $\alpha$ -contaminated, by Lemma 11.

Let  $\mathcal{S}$  be the set of all vertices that have a 1-path from  $i$  to them. Consider the ascent part of the path  $\pi(i, j): I_0 \rightarrow I_1 \rightarrow \dots \rightarrow I_\ell$ . The claim is that for every block  $I_t$  in this path, we have that at least  $\frac{3}{4}$  of the vertices have 1-paths from  $i$  (i.e.,  $|I_t \cap \mathcal{S}| \geq \frac{3}{4} |I_t|$ ).

This claim is proven by induction. The claim trivially holds for  $I_0$ . Now, consider two consecutive blocks  $I_t \rightarrow I_{t+1}$ . There are two cases:

- (i)  $I_{t+1} = \text{next}(I_t)$ . Then, the graph  $H$  includes the expander graph on  $I_t, I_{t+1}$  described in Lemma 4. At least  $\frac{3}{4} |I_t|$  vertices of  $I_t$  are in  $\mathcal{S}$ . As such, at least  $\frac{15}{16} |I_{t+1}|$  vertices of  $I_{t+1}$  are reachable from the vertices of  $I_t$ . Since  $I_{t+1}$  is not  $\alpha$ -contaminated, at most an  $\alpha$ -fraction of vertices of  $I_{t+1}$  are in  $B$ , and it follows that  $|I_{t+1} \cap \mathcal{S}| \geq (\frac{15}{16} - \alpha) |I_{t+1}| \geq \frac{3}{4} |I_{t+1}|$ , as claimed.
- (ii)  $I_{t+1}$  is the parent of  $I_t$ . In this case,  $I_t$  is the left child of  $I_{t+1}$ . Let  $I'_t$  be the right child of  $I_{t+1}$ . Since  $I_{t+1}$  is not  $\alpha$ -contaminated, we have that  $|I_{t+1} \cap B| \leq \alpha |I_{t+1}|$ . As such,

$$|I'_t \cap B| \leq |I_{t+1} \cap B| \leq 2\alpha |I'_t|$$

Now, by the expander construction on  $(I_t, I'_t)$ , and arguing as above, we have

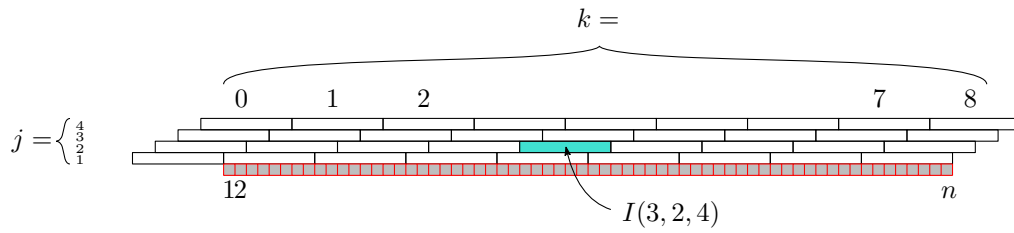
$$|I'_t \cap \mathcal{S}| \geq \left( \frac{15}{16} - 2\alpha \right) |I'_t| \geq \frac{3}{4} |I'_t|,$$

which implies that  $|I_{t+1} \cap \mathcal{S}| \geq \frac{3}{4} |I_{t+1}|$ .

The symmetric claim for the descent part of the path is handled in a similar fashion, therefore, at least  $\frac{3}{4}$  of the points in  $J_\ell$  can reach  $j$  with a 1-path. Using these and the expander construction between  $I_\ell$  and  $J_\ell$ , we conclude that there is a 1-path from  $i$  to  $j$  in  $H \setminus B$ , as claimed.  $\blacktriangleleft$

Note that it is easy to generalize the construction for arbitrary  $n$ . Let  $h$  be an integer such that  $2^{h-1} < n < 2^h$  and build the graph  $H$  on  $\{1, 2, 3, \dots, 2^h\}$ . Since  $H$  is a 1-spanner, the 1-paths between any pair of vertices of  $[n]$  does not use any vertices from  $\{n+1, \dots, 2^h\}$ . Therefore, we can simply delete the part of  $H$  that is beyond  $n$  to obtain an  $\mathcal{O}(1)$ -reliable 1-spanner on  $[n]$ . Since we defined  $B^+$  to be the shadow of  $B$ , the  $\mathcal{O}(1)$ -reliability is inherited automatically.

We also note that no effort was made to optimize the constants in the above construction.



■ **Figure 3** The shifted intervals  $I(i, \cdot, \cdot)$  for  $i = 3$  with  $N = 4$  and  $n = 64$ . Each interval has length  $2^i = 8$ , there are  $N = 4$  different shifts and there are  $\frac{n}{2^i} + 1 = 9$  blocks per each shift.

### 3.3 Construction of $\vartheta$ -reliable exact spanners in one dimension

Here, we show how to extend Theorem 12, to build a one dimensional graph, such that for any fixed  $\vartheta > 0$  and any set  $B$  of  $k$  deleted vertices, at most  $(1 + \vartheta)k$  vertices are no longer connected (by a 1-path) after the removal of  $B$ . The basic idea is to retrace the construction of Theorem 12, and extend it to this more challenging case. The main new ingredient is a shifting scheme.

Let  $[n]$  be the ground set, and assume that  $n$  is a power of two, and let  $h = \log n$ . Let

$$N = \text{pow}_2(c/\vartheta^2) \quad \text{and} \quad \xi = \frac{1}{32N}, \tag{1}$$

where  $c$  is a sufficiently large constant ( $c \geq 512$ ). We first connect any  $i \in [n]$ , to all the vertices that are in distance at most  $3N$  from it, by adding an edge between the two vertices. Let  $G_0$  be the resulting graph.

Let  $i_0 = \log N$ . For  $i = i_0, \dots, h - 1$ , and  $j = 1, \dots, N$ , let

$$\Delta(i, j) = 1 + (j - 1)2^i/N - 2^i$$

be the *shift* corresponding to  $i$  and  $j$ . For a fixed  $i$ , the  $\Delta(i, j)$ s are  $N$  equally spaced numbers in the block  $[1 - 2^i : 1 - 2^i/N]$ , starting at its left endpoint. For  $k = 0, \dots, n/2^i$ , let

$$I(i, j, k) = [\Delta(i, j) + k2^i : \Delta(i, j) + (k + 1)2^i]$$

be the shifted interval of length  $2^i$  that starts at  $\Delta(i, j)$  and is shifted  $k$  blocks to the right, see Figure 3. The set of all intervals of interest is

$$\mathcal{I} = \left\{ I(i, j, k) \mid \begin{array}{l} i = i_0, \dots, \log n \\ j = 1, \dots, N \\ k = 0, \dots, n/2^i \end{array} \right\}.$$

**Constructing the graph  $H_\vartheta$ .** Let  $G_E(i, j, k)$  denote the expander graph of Lemma 4, constructed over  $I(i, j, k)$  and  $I(i, j, k + 1)$ , with the value of the parameter  $\xi$  as specified in Eq. (1). We define  $H_\vartheta$  to be the union of all the graphs  $G_E$  over all choices of  $i, j, k$ , and also including the graph  $G_0$  (described above). In the case that  $n$  is not a power of two, do the construction on  $[\text{pow}_2(n)]$ . In any case, the last step is to delete vertices from  $H_\vartheta$  that are outside the range of interest  $[n]$ .

► **Theorem 13.** *For parameters  $n$  and  $\vartheta > 0$ , the graph  $H_\vartheta$  constructed over  $[n]$ , is a  $\vartheta$ -reliable exact spanner. Furthermore,  $H_\vartheta$  has  $\mathcal{O}(\vartheta^{-6}n \log n)$  edges.*

## 4 Building a reliable spanner in $\mathbb{R}^d$

In the following, we assume that  $P \subseteq [0, 1]^d$  – this can be done by an appropriate scaling and translation of space. We use a novel result of Chan *et al.* [11], called locality-sensitive orderings. These orderings can be thought as an alternative to quadtrees and related structures. For an ordering  $\sigma$  of  $[0, 1]^d$ , and two points  $p, q \in [0, 1]^d$ , such that  $p \prec q$ , let  $(p, q)_\sigma = \{z \in [0, 1]^d \mid p \prec z \prec q\}$  be the set of points between  $p$  and  $q$  in the order  $\sigma$ .

► **Theorem 14** ([11]). *For  $\varsigma \in (0, 1)$  fixed, there is a set  $\Pi^+(\varsigma)$  of  $M(\varsigma) = \mathcal{O}(\varsigma^{-d} \log \varsigma^{-1})$  orderings of  $[0, 1]^d$ , such that for any two (distinct) points  $p, q \in [0, 1]^d$ , with  $\ell = \|p - q\|$ , there is an ordering  $\sigma \in \Pi^+$ , and a point  $z \in [0, 1]^d$ , such that*

- (i)  $p \prec_\sigma q$ ,
- (ii)  $(p, z)_\sigma \subseteq \text{ball}(p, \varsigma\ell)$ ,
- (iii)  $(z, q)_\sigma \subseteq \text{ball}(q, \varsigma\ell)$ , and
- (iv)  $z \in \text{ball}(p, \varsigma\ell)$  or  $z \in \text{ball}(q, \varsigma\ell)$ .

Furthermore, given such an ordering  $\sigma$ , and two points  $p, q$ , one can compute their ordering, according to  $\sigma$ , using  $\mathcal{O}(d \log \varsigma^{-1})$  arithmetic and bitwise-logical operations.

### 4.1 Construction

Given a set  $P$  of  $n$  points in  $[0, 1]^d$ , and parameters  $\varepsilon, \vartheta \in (0, 1)$ , let  $\varsigma = \varepsilon/c$ ,

$$M = M(\varsigma) = \mathcal{O}(\varsigma^{-d} \log \varsigma^{-1}) = \mathcal{O}(\varepsilon^{-d} \log \varepsilon^{-1}),$$

and  $c$  be some sufficiently large constant. Next, let  $\vartheta' = \vartheta/(3N \cdot M)$  where  $N = \lceil \log \log n \rceil + 1$ , and let  $\Pi^+ = \Pi^+(\varsigma)$  be the set of orderings of Theorem 14. For each ordering  $\sigma \in \Pi^+$ , compute the  $\vartheta'$ -reliable exact spanner  $G_\sigma$  of  $P$ , see Theorem 13, according to  $\sigma$ . Let  $G$  be the resulting graph by taking the union of  $G_\sigma$  for all  $\sigma \in \Pi^+$ .

### 4.2 Analysis

► **Theorem 15.** *The graph  $G$ , constructed above, is a  $\vartheta$ -reliable  $(1 + \varepsilon)$ -spanner and has size*

$$\mathcal{O}\left(\varepsilon^{-7d} \log^7 \frac{1}{\varepsilon} \cdot \vartheta^{-6} n \log n (\log \log n)^6\right).$$

**Proof.** First, we show the bound on the size. There are  $M$  different orderings for which we build the graph of Theorem 13. Each of these graphs has  $\mathcal{O}((\vartheta')^{-6} n \log n)$  edges. Therefore, the size of  $G$  is

$$M \cdot \mathcal{O}((\vartheta')^{-6} n \log n) = \mathcal{O}\left(M \left(\frac{3NM}{\vartheta}\right)^6 n \log n\right) = \mathcal{O}\left(\varepsilon^{-7d} \log^7 \frac{1}{\varepsilon} \cdot \vartheta^{-6} n \log n (\log \log n)^6\right).$$

Next, we identify the set of harmed vertices  $B^+$  given a set of failed vertices  $B \subseteq P$ . First, let  $B_1$  be the union of all the  $(1 - \vartheta'/4)$ -shadows resulting from  $B$  in  $G_\sigma$ , for all  $\sigma \in \Pi^+$ . Then, for  $i = 2, \dots, N$ , we define  $B_i$  in a recursive manner to be the union of all the  $(1 - \vartheta'/4)$ -shadows resulting from  $B_{i-1}$  in  $G_\sigma$ , for all  $\sigma \in \Pi^+$ . We set  $B^+ = B_N$ .

By the recursion and Lemma 9 we have that

$$\begin{aligned} |B_i| &\leq \left( \frac{|B_{i-1}|}{2(1 - \frac{\vartheta'}{4}) - 1} - |B_{i-1}| \right) M + |B_{i-1}| = \frac{|B_{i-1}| - (1 - \frac{\vartheta'}{2})|B_{i-1}|}{1 - \frac{\vartheta'}{2}} M + |B_{i-1}| \\ &= \frac{\vartheta' |B_{i-1}|}{2 - \vartheta'} M + |B_{i-1}| \leq (1 + \vartheta' M) |B_{i-1}| = \left(1 + \frac{\vartheta'}{3N}\right) |B_{i-1}|. \end{aligned}$$

Therefore, we obtain

$$|B^+| = |B_N| \leq \left(1 + \frac{\vartheta}{3N}\right)^N |B| \leq \exp\left(N \frac{\vartheta}{3N}\right) |B| \leq (1 + \vartheta) |B|,$$

using  $1 + x \leq e^x \leq 1 + 3x$ , for  $x \in [0, 1]$ .

Now we show that there is a  $(1 + \varepsilon)$ -path  $\hat{\pi}$  between any pair of vertices  $p, q \in P \setminus B^+ \equiv P \setminus B_N$  such that the path  $\hat{\pi}$  does not use any vertices of  $B$ . By Theorem 13, the graph  $G_\sigma \setminus B_{N-1} \subseteq G \setminus B_{N-1}$  contains a monotone path connecting  $p$  to  $q$ , according to  $\sigma$ . Observe that there is a unique edge  $(p', q')$  on this path that “jumps” from the locality of  $p$  to the locality of  $q$ . Formally, we have the following:

- (a)  $\|p' - q'\| \leq \|p - q\| + 2\varsigma \|p - q\| = (1 + 2\varepsilon/c) \|p - q\|$ .
- (b)  $\|p - p'\| \leq 2\varsigma \|p - q\| = 2(\varepsilon/c) \|p - q\|$  and similarly  $\|q - q'\| \leq 2(\varepsilon/c) \|p - q\|$ .
- (c)  $p', q' \in P \setminus B_{N-1}$ .

We fix the edge  $(p', q')$  to be used in the computed path  $\hat{\pi}$  connecting  $p$  to  $q$ . We still need to build the two parts of the path  $\hat{\pi}$  between  $p, p'$  and  $q, q'$ .

This procedure reduced the problem of computing a reliable path between two points  $p, q \in P \setminus B_N$ , into computing two such paths between two points of  $P \setminus B_{N-1}$  (i.e.,  $p, p'$  and  $q, q'$ ). The benefit here is that  $\|p - p'\|, \|q - q'\| \ll \|p - q\|$ . We now repeat this refinement process  $N - 1$  times.

To this end, let  $Q_i$  be the set of active pairs that needs to be connected in the  $i$ th level of the recursion. Thus, we have  $Q_0 = \{(p, q)\}$ ,  $Q_1 = \{(p, p'), (q, q')\}$ , and so on. We repeat this  $N - 1$  times. In the  $i$ th level there are  $|Q_i| = 2^i$  active pairs. Let  $(x, y) \in Q_i$  be such a pair. Then, there is an edge  $(x', y')$  in the graph  $G \setminus B_{N-(i+1)}$ , such that we have the following:

- (a)  $\|x' - y'\| \leq \|x - y\| (1 + 2\varepsilon/c) \leq (2\varepsilon/c)^i (1 + 2\varepsilon/c) \|p - q\|$ .
- (b)  $\|x - x'\| \leq 2(\varepsilon/c) \|x - y\| \leq (2\varepsilon/c)^{i+1} \|p - q\|$  and  $\|y - y'\| \leq (2\varepsilon/c)^{i+1} \|p - q\|$ .
- (c)  $x', y' \in P \setminus B_{N-(i+1)}$ .

The edge  $(x', y')$  is added to the path  $\hat{\pi}$ . After  $N - 1$  iterations the set of active pairs is  $Q_{N-1}$  and for each pair  $(x, y) \in Q_{N-1}$  we have that  $x, y \in P \setminus B_1$ . For each of these pairs  $(x, y) \in Q_{N-1}$  we apply Theorem 14 and Theorem 13 to obtain a path of length at most  $\|x - y\| 2 \log n$  between  $x$  and  $y$  (and this subpath of course does not contain any vertex in  $B$ ). We add all these subpaths to connect the active pairs in the path  $\hat{\pi}$ , which completes  $\hat{\pi}$  into a path.

Now, we bound the length of path  $\hat{\pi}$ . Since, for all  $(x, y) \in Q_{N-1}$ , we have  $\|x - y\| \leq \|p - q\| \cdot (2\varepsilon/c)^{N-1}$  and  $|Q_{N-1}| = 2^{N-1}$ , the total length of the subpaths calculated, in the last step, is

$$\begin{aligned} \sum_{(x,y) \in Q_{N-1}} \text{length}(\hat{\pi}[x, y]) &\leq 2^{N-1} \|p - q\| \cdot \left(\frac{2\varepsilon}{c}\right)^{N-1} 2 \log n \\ &\leq \|p - q\| \cdot \left(\frac{4\varepsilon}{c}\right)^{\log \log n} 2 \log n \leq \|p - q\| \cdot \varepsilon^{\log \log n} \left(\frac{4}{c}\right)^{\log \log n} 2 \log n \\ &\leq \|p - q\| \cdot \frac{\varepsilon}{4} \cdot \frac{1}{\log n} \cdot 2 \log n \leq \frac{\varepsilon}{2} \|p - q\|, \end{aligned}$$

for large enough  $n$  and  $c \geq 8$ . The total length of the long edges added to  $\hat{\pi}$  in the recursion,



## 19:12 A Spanner for the Day After

is bounded by

$$\begin{aligned} \sum_{i=0}^{N-2} 2^i \|p - q\| \left(\frac{2\varepsilon}{c}\right)^i \left(1 + \frac{2\varepsilon}{c}\right) &\leq \|p - q\| \left(1 + \frac{2\varepsilon}{c}\right) \sum_{i=0}^{\infty} \left(\frac{4\varepsilon}{c}\right)^i \\ &= \|p - q\| \left(1 + \frac{2\varepsilon}{c}\right) \frac{1}{1 - 4\varepsilon/c} = \|p - q\| \left(1 + \frac{6\varepsilon}{c - 4\varepsilon}\right) \leq \left(1 + \frac{\varepsilon}{2}\right) \|p - q\|, \end{aligned}$$

which holds for  $c \geq 16$ . Therefore, the computed path  $\hat{\pi}$  between  $p$  and  $q$  is a  $(1 + \varepsilon)$ -path in  $G \setminus B$ , which concludes the proof of the theorem.  $\blacktriangleleft$

### 5 Construction for points with bounded spread in $\mathbb{R}^d$

The input is again a set  $P \subset \mathbb{R}^d$  of  $n$  points, and parameters  $\vartheta \in (0, 1/2)$  and  $\varepsilon \in (0, 1)$ . The goal is to build a  $\vartheta$ -reliable  $(1 + \varepsilon)$ -spanner on  $P$  that has optimal size under some condition on  $P$ . The condition is that the spread  $\Phi(P)$  is bounded by a polynomial of  $n$ . The construction is based on well-separated pair decompositions (WSPD), which was introduced by Callahan and Kosaraju [9]. For preliminaries see the full version [8].

#### 5.1 The construction of $G_{\Phi}$

First, compute a quadtree  $T$  for the point set  $P$ . For any node  $v \in T$ , let  $\square_v$  denote the *cell* (i.e. square or cube, depending on the dimension) represented by  $v$ . Let  $P_v = \square_v \cap P$  be the point set stored in the subtree of  $v$ . Compute a  $(6/\varepsilon)$ -WSPD  $\mathcal{W}$  over  $T$  for  $P$  using the construction of Abam and Har-Peled [1, Lemma 2.8]. The pairs in  $\mathcal{W}$  can be represented by pairs of nodes  $\{u, v\}$  of the quadtree  $T$ . Note that the algorithm uses the diameters and distances of the cells of the quadtree, that is, for a pair  $\{u, v\} \in \mathcal{W}$ , we have

$$(6/\varepsilon) \cdot \max(\text{diam}(\square_u), \text{diam}(\square_v)) \leq d(\square_u, \square_v).$$

For any pair  $\{u, v\} \in \mathcal{W}$ , we build the bipartite expander of Lemma 4 on the sets  $P_u$  and  $P_v$  such that the expander property holds with  $\xi = \vartheta/8$ . Furthermore, for every two node  $u$  and  $v$  that have the same parent in the quadtree  $T$  we add the edges of the bipartite expander of Lemma 4 between  $P_u$  and  $P_v$ . Let  $G_{\Phi}$  be the resulting graph when taking the union over all the sets of edges generated by the above.

#### 5.2 Analysis

► **Lemma 16.** *The graph  $G_{\Phi}$  has  $\mathcal{O}(\xi^{-2}\varepsilon^{-d}n \log \Phi(P))$  edges.*

**Proof.** By Lemma 5.4 of [8], every point participates in  $\mathcal{O}(\varepsilon^{-d} \log \Phi(P))$  WSPD pairs. By Lemma 4 the average degree in all the expanders is at most  $\mathcal{O}(1/\xi^2)$ , resulting in the given bound on the number of edges. There are also the additional pairs between a node in  $T$  and its parent, but since every point participates in only  $\mathcal{O}(\log \Phi(P))$  such pairs, the number of edges is dominated by the expanders on the WSPD pairs. It follows that the number of edges in the resulting graph is  $\mathcal{O}(\xi^{-2}\varepsilon^{-d}n \log \Phi(P))$ .  $\blacktriangleleft$

► **Definition 17.** *For a number  $\gamma \in (0, 1)$ , and failed set of vertices  $B \subseteq P$ , a node  $v$  of the quadtree  $T$  is in the  $\gamma$ -shadow if  $|B \cap P_v| \geq \gamma |P_v|$ . Naturally, if  $v$  is in the  $\gamma$ -shadow, then the points of  $P_v$  are also in the shadow. As such, the  $\gamma$ -shadow of  $B$  is the set of all the points in the shadow – formally,  $\mathcal{S}(\gamma, B) = \bigcup_{v \in T: |B \cap P_v| \geq \gamma |P_v|} P_v$ .*

Let  $\gamma = 1 - \vartheta/2$ . Note that  $B \subseteq \mathcal{S}(\gamma, B)$ , since every point of  $B$  is stored as a singleton in a leaf of  $T$ .

► **Definition 18.** For a node  $x$  in  $T$ , let  $n(x) = |P_x|$ , and  $b(x) = |P_x \cap B|$ .

► **Lemma 19.** Let  $\gamma = 1 - \vartheta/2$  and  $B \subseteq P$  be fixed. Then, the size of the  $\gamma$ -shadow of  $B$  is at most  $(1 + \vartheta)|B|$ .

**Proof.** Let  $H$  be the set of nodes of  $T$  that are in the  $\gamma$ -shadow of  $B$ . A node  $u \in H$  is *maximal* if none of its ancestors is in  $H$ . Let  $H' = \{u_1, \dots, u_m\}$  be the set of all maximal nodes in  $H$ , and observe that  $\cup_{u \in H'} P_u = \cup_{v \in H} P_v = \mathcal{S}(\gamma, B)$ . For any two nodes  $x, y \in H'$ , we have  $P_x \cap P_y = \emptyset$ . Therefore, we have

$$|B| = \sum_{u \in H'} b(u) \geq \sum_{u \in H'} \gamma n(u) = \gamma |\mathcal{S}(\gamma, B)|.$$

Dividing both sides by  $\gamma$  implies the claim, since  $1/\gamma = 1/(1 - \vartheta/2) \leq 1 + \vartheta$ . ◀

► **Lemma 20.** Let  $\gamma = 1 - \vartheta/2$ . Fix a node  $u \in T$  of the quadtree, the failure set  $B \subseteq P$ , its shadow  $B^+ = \mathcal{S}(\gamma, P)$ , and the residual graph  $H = G_\Phi \setminus B$ . For a point  $p \in P_u \setminus B^+$ , let  $R_u(p)$  be the set of all reachable points in  $P_u$  with stretch two, formally,  $R_u(p) = \{q \in P_u \setminus B \mid d_H(p, q) \leq 2 \cdot \text{diam}(\square_u)\}$ . Then, we have  $|R_u(p)| \geq 3\xi |P_u|$ .

**Proof.** Let  $u_1, u_2, \dots, u_j = u$  be the sequence of nodes in the quadtree from the leaf  $u_1$  that contains (only)  $p$ , to the node  $u$ . A *level* of a point  $q \in P_u$ , denoted by  $\ell(q)$ , is the first index  $i$ , such that  $q \in P_{u_i}$ . A *skipping path* in  $G_\Phi$ , is a sequence of edges  $pq_1, q_1q_2, \dots, q_{m-1}q_m$ , such that  $\ell(q_i) < \ell(q_{i+1})$ , for all  $i$ .

Let  $Q_i$  be the set of all points in  $P_{u_i} \setminus B$  that are reachable by a skipping path in  $H$  from  $p$ . We claim, for  $i = 1, \dots, j$ , that

$$|Q_i| \geq (1 - \xi)n(u_i) - b(u_i) \geq (1 - \xi - \gamma)n(u_i) = (\vartheta/2 - \xi)n(u_i) = 3\xi n(u_i),$$

since  $\xi = \vartheta/8$  and  $p$  is not in the  $\gamma$ -shadow. The claim clearly holds for  $u_1$ . So, assume inductively that the claim holds for  $u_1, \dots, u_{j-1}$ . Let  $v_1, \dots, v_m$  be the children of  $u_j$  that have points stored in them (excluding  $u_{j-1}$ ). There is an expander between  $P_{u_{j-1}}$  and  $P_{v_i}$ , for all  $i$ , as a subgraph of  $G_\Phi$ . It follows, by induction, that

$$\begin{aligned} |Q_j| &\geq (1 - \xi)n(u_{j-1}) - b(u_{j-1}) + \sum_i ((1 - \xi)n(v_i) - b(v_i)) \\ &= (1 - \xi)n(u_{j-1}) + \sum_i (1 - \xi)n(v_i) - \left(b(u_{j-1}) + \sum_i b(v_i)\right) = (1 - \xi)n(u_j) - b(u_j). \end{aligned}$$

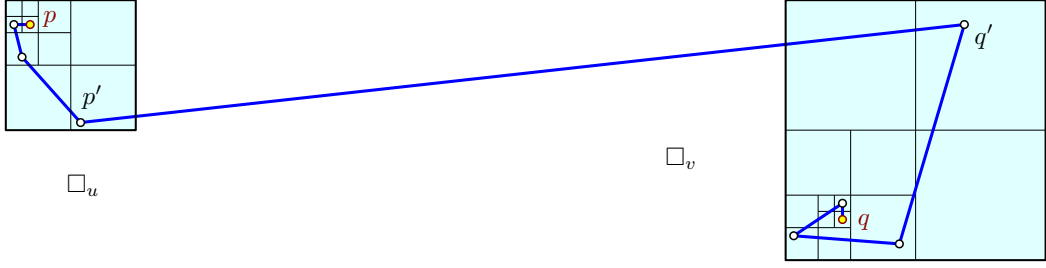
Observe that a skipping path from  $p$  to  $q \in P_{u_j}$  has length at most

$$\sum_{i=1}^j \text{diam}(\square_{u_i}) \leq \text{diam}(\square_{u_j}) \sum_{i=1}^j 2^{1-j} \leq 2 \cdot \text{diam}(\square_{u_j}).$$

Thus,  $Q_j \subseteq R_u(p)$ , and the claim follows. ◀

Now we are ready to prove that  $G_\Phi$  is a reliable spanner.

► **Lemma 21.** For a set  $P \subseteq \mathbb{R}^d$  of  $n$  points and parameters  $\varepsilon \in (0, 1)$  and  $\vartheta \in (0, 1/2)$ , the graph  $G_\Phi$  is a  $\vartheta$ -reliable  $(1 + \varepsilon)$ -spanner with  $\mathcal{O}(\varepsilon^{-d} \vartheta^{-2} n \log \Phi(P))$  edges, where  $\Phi(P)$  is the spread of  $P$ .



■ **Figure 4** The pair  $\{u, v\} \in \mathcal{W}$  that separates  $p$  and  $q$ . The blue path is a  $(1 + \varepsilon)$ -path between  $p$  and  $q$  in the graph  $G_\Phi \setminus B$ .

**Proof.** Let  $\xi = \vartheta/8$  and  $\gamma = 1 - \vartheta/2$ . The bound on the number of edges follows by Lemma 16.

Let  $B$  be a set of faulty vertices of  $G_\Phi$ , and let  $H = G_\Phi \setminus B$  be the residual graph. We define  $B^+$  to contain the vertices that are in the  $\gamma$ -shadow of  $B$ . Then, we have  $B \subseteq B^+$  and  $|B^+| \leq (1 + \vartheta) |B|$  by Lemma 19. Finally, we need to show that there exists a  $(1 + \varepsilon)$ -path between any  $p, q \in P \setminus B^+$ .

Let  $\{u, v\} \in \mathcal{W}$  be the pair that separates  $p$  and  $q$  with  $p \in P_u$  and  $q \in P_v$ , see Figure 4. Let  $R_u(p)$  (resp.  $R_v(q)$ ) be the set of points in  $P_u$  (resp.  $P_v$ ) that are reachable in  $H$  from  $p$  (resp.  $q$ ) with paths that have lengths at most  $2 \cdot \text{diam}(\square_u)$  (resp.  $2 \cdot \text{diam}(\square_v)$ ). By Lemma 20,  $|R_u(p)| \geq 3\xi n(u) \geq \xi n(u)$  and  $|R_v(q)| \geq 3\xi n(v)$ .

Since there is a bipartite expander between  $P_u$  and  $P_v$  with parameter  $\xi$ , by Lemma 4, the neighborhood  $Y$  of  $R_u(p)$  in  $P_v$  has size at least  $(1 - \xi)n(v)$ . Observe that  $|Y \cap R_v(q)| = |R_v(q) \setminus (P_v \setminus Y)| \geq |R_v(q)| - |P_v \setminus Y| \geq 3\xi n(v) - \xi n(v) > 0$ . Therefore, there is a point  $q' \in Y \cap R_v(q)$ , and a point  $p' \in R_u(p)$ , such that  $p'q' \in E(G_\Phi)$ . We have that

$$\begin{aligned} d_H(p, q) &\leq d_H(p, p') + d_H(p', q') + d_H(q', q) \leq 2 \cdot \text{diam}(\square_u) + \|p' - q'\| + 2 \cdot \text{diam}(\square_v) \\ &\leq 3 \cdot \text{diam}(\square_u) + d(\square_u, \square_v) + 3 \cdot \text{diam}(\square_v) \leq \left(1 + 6 \cdot \frac{\varepsilon}{6}\right) \cdot d(\square_u, \square_v) \\ &\leq (1 + \varepsilon) \cdot \|p - q\|. \end{aligned} \quad \blacktriangleleft$$

## 6 Conclusions

In this paper we have shown several constructions for  $\vartheta$ -reliable spanners. Our results for constructing reliable exact spanners in one dimension have size  $\mathcal{O}(n \log n)$ , which is optimal. In higher dimensions we were able to show a simple construction of a  $\vartheta$ -reliable spanner with optimal size for the case of bounded spread. For arbitrary point sets in  $\mathbb{R}^d$  we obtained a construction with an extra  $(\log \log n)^6$  factor in the size.

It seems clear that our construction for the unbounded case is suboptimal in terms of extra factors, and we leave improving it as an open problem for further research. Another natural open question is how to construct reliable spanners that are required to be subgraphs of a given graph.

---

**References**

---

- 1 M. A. Abam and S. Har-Peled. New Constructions of SSPDs and their Applications. *Computational Geometry: Theory and Applications*, 45(5–6):200–214, 2012. doi:10.1016/j.comgeo.2011.12.003.
- 2 B. Aronov, M. de Berg, O. Cheong, J. Gudmundsson, H. J. Haverkort, M. H. M. Smid, and A. Vigneron. Sparse geometric graphs with small dilation. *Computational Geometry: Theory and Applications*, 40(3):207–219, 2008. doi:10.1016/j.comgeo.2007.07.004.
- 3 S. Arya, D. M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS)*, pages 703–712, 1994. doi:10.1109/SFCS.1994.365722.
- 4 S. Arya, D. M. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry: Theory and Applications*, 13(2):91–107, 1999. doi:10.1016/S0925-7721(99)00014-0.
- 5 P. Bose, P. Carmi, V. Dujmovic, and P. Morin. Near-Optimal  $O(k)$ -Robust Geometric Spanners. *CoRR*, abs/1812.09913, 2018. arXiv:1812.09913.
- 6 P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. Smid. Computing the Greedy Spanner in Near-Quadratic Time. *Algorithmica*, 58(3):711–729, November 2010. doi:10.1007/s00453-009-9293-4.
- 7 P. Bose, V. Dujmović, P. Morin, and M. Smid. Robust Geometric Spanners. *SIAM Journal on Computing*, 42(4):1720–1736, 2013. doi:10.1137/120874473.
- 8 K. Buchin, S. Har-Peled, and D. Oláh. A Spanner for the Day After. *CoRR*, abs/1811.06898, 2018. arXiv:1811.06898.
- 9 P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to  $k$ -nearest-neighbors and  $n$ -body potential fields. *Journal of the Association for Computing Machinery*, 42(1):67–90, 1995. doi:10.1145/200836.200853.
- 10 P. Carmi and L. Chaitman. Stable Roommates and Geometric Spanners. In *Proc. 22nd Canad. Conf. Comput. Geom. (CCCG)*, pages 31–34, 2010. URL: <http://cccg.ca/proceedings/2010/paper11.pdf>.
- 11 T. M. Chan, S. Har-Peled, and M. Jones. On Locality-Sensitive Orderings and their Applications. *CoRR*, abs/1809.11147, 2018. arXiv:1809.11147.
- 12 J. Gudmundsson, C. Levkopoulos, and G. Narasimhan. Fast Greedy Algorithms for Constructing Sparse Geometric Spanners. *SIAM J. Comput.*, 31(5):1479–1500, May 2002. doi:10.1137/S0097539700382947.
- 13 C. Levkopoulos, G. Narasimhan, and M. Smid. Efficient Algorithms for Constructing Fault-Tolerant Geometric Spanners. In *Proc. 30th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 186–195. ACM, 1998. doi:10.1145/276698.276734.
- 14 C. Levkopoulos, G. Narasimhan, and M. Smid. Improved Algorithms for Constructing Fault-Tolerant Spanners. *Algorithmica*, 32(1):144–156, 2002. doi:10.1007/s00453-001-0075-x.
- 15 T. Lukovszki. New Results of Fault Tolerant Geometric Spanners. In *Proc. 6th Workshop Algorithms Data Struct. (WADS)*, volume 1663 of *LNCS*, pages 193–204. Springer, 1999. doi:10.1007/3-540-48447-7\_20.
- 16 R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.
- 17 G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, New York, NY, USA, 2007.
- 18 M. Smid. Geometric Spanners with Few Edges and Degree Five. In *Proc. 12th Australasian Theo. Sym. (CATS)*, volume 51 of *CRPIT*, pages 7–9, 2006. URL: <http://crpit.com/abstracts/CRPITV51Smid.html>.



# Computing Shapley Values in the Plane

Sergio Cabello 

Department of Mathematics, FMF, University of Ljubljana, Slovenia

Department of Mathematics, IMFM, Ljubljana, Slovenia

<https://www.fmf.uni-lj.si/~cabello/>

[sergio.cabello@fmf.uni-lj.si](mailto:sergio.cabello@fmf.uni-lj.si)

Timothy M. Chan

Department of Computer Science, University of Illinois at Urbana-Champaign, USA

<http://tmc.web.engr.illinois.edu/>

[tmc@illinois.edu](mailto:tmc@illinois.edu)

---

## Abstract

We consider the problem of computing Shapley values for points in the plane, where each point is interpreted as a player, and the value of a coalition is defined by the area of usual geometric objects, such as the convex hull or the minimum axis-parallel bounding box.

For sets of  $n$  points in the plane, we show how to compute in roughly  $O(n^{3/2})$  time the Shapley values for the area of the minimum axis-parallel bounding box and the area of the union of the rectangles spanned by the origin and the input points. When the points form an increasing or decreasing chain, the running time can be improved to near-linear. In all these cases, we use linearity of the Shapley values and algebraic methods.

We also show that Shapley values for the area of the convex hull or the minimum enclosing disk can be computed in  $O(n^2)$  and  $O(n^3)$  time, respectively. These problems are closely related to the model of stochastic point sets considered in computational geometry, but here we have to consider random insertion orders of the points instead of a probabilistic existence of points.

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

**Keywords and phrases** Shapley values, stochastic computational geometry, convex hull, minimum enclosing disk, bounding box, arrangements, convolutions, airport problem

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.20

**Related Version** Full version available at <http://arxiv.org/abs/1804.03894>.

**Funding** *Sergio Cabello*: Supported by the Slovenian Research Agency (P1-0297, J1-8130, J1-8155). *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

**Acknowledgements** The authors are very grateful to Sariel Har-Peled for fruitful discussions.

## 1 Introduction

One can associate several meaningful values to a set  $P$  of points in the plane, like for example the area of the convex hull or the area of the axis-parallel bounding box. How can we split this value among the points of  $P$ ? Shapley values are a standard tool in cooperative games to “fairly” split common cost between different players. Our objective in this paper is to present algorithms to compute the Shapley values for points in the plane when the cost of each subset is defined by geometric means.

**Coalitional games in the plane.** Formally, a **coalitional game** is a pair  $(P, v)$ , where  $P$  is the set of players and  $v: 2^P \rightarrow \mathbb{R}$  is the **characteristic function**, which must satisfy  $v(\emptyset) = 0$ . Depending on the problem at hand, the characteristic function can be seen as a cost or a payoff associated to each subset of players (usually called a *coalition*). Coalitional games are a very common model for cooperative games with transferable utility.



© Sergio Cabello and Timothy M. Chan;

licensed under Creative Commons License CC-BY

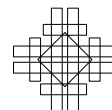
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 20; pp. 20:1–20:19

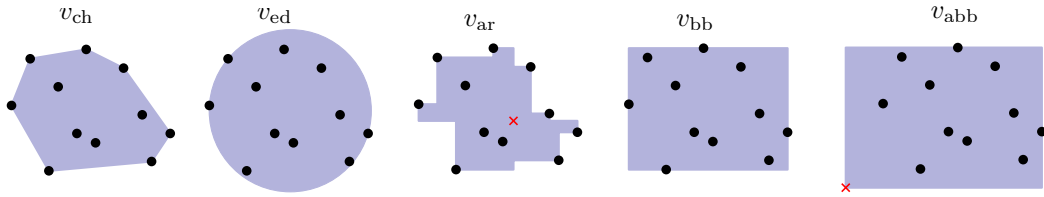
Leibniz International Proceedings in Informatics



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



## 20:2 Computing Shapley Values in the Plane



■ **Figure 1** Different costs associated to a point set that are considered in this paper. The cross represents the origin. In all cases we focus on the area.

In our setting, the players will be points in the plane. Thus  $P \subset \mathbb{R}^2$ . Such scenario arises naturally in the context of game theory through modeling: each point represents an agent, and each coordinate of the point represents an attribute of the agent.

We will consider characteristic functions given by the area of shapes that “enclose” the points. The shapes that we consider are succinctly described in Figure 1. More precisely, we consider the following coalitional games.

**AREA CONVEX HULL game:** The characteristic function is  $v_{\text{ch}}(Q) = \text{area}(CH(Q))$  for each nonempty  $Q \subset P$ , where  $CH(Q)$  denotes the convex hull of  $Q$ .

**AREA ENCLOSED DISK game:** The characteristic function is  $v_{\text{ed}}(Q) = \text{area}(\text{med}(Q))$  for each nonempty  $Q \subset P$ , where  $\text{med}(Q)$  is a disk of smallest radius that contains  $Q$ .

**AREA ANCHORED RECT game:** The characteristic function is  $v_{\text{ar}}(Q) = \text{area}\left(\bigcup_{p \in Q} R_p\right)$  for each nonempty  $Q \subset P$ , where  $R_p$  is the axis-parallel rectangle with one corner at  $p$  and another corner at the origin.

**AREA BOUNDING BOX game:** The characteristic function is  $v_{\text{bb}}(Q) = \text{area}(\text{bb}(Q))$  for each nonempty  $Q \subset P$ , where  $\text{bb}(Q)$  is the smallest axis-parallel bounding box of  $Q$ .

**AREA ANCHORED BOUNDING BOX game:** The characteristic function is  $v_{\text{abb}}(Q)$ , defined by  $\text{area}(\text{bb}(Q \cup \{o\}))$  for each nonempty  $Q \subset P$ , where  $o$  is the origin.

In the full version [5] we also consider variants of these problems where the perimeter of the shapes is used.

**Shapley values.** Shapley values are probably the most popular solution concept for coalitional games. The objective is to split the value  $v(P)$  between the different players of a coalitional game  $(P, v)$  in a meaningful way. It is difficult to overestimate the relevance of Shapley values. See the book edited by Roth [26] or the survey by Winter [29] for a general discussion showing their relevance. There are also different axiomatic characterizations of the concept, meaning that Shapley values can be shown to be the only map satisfying certain natural conditions. Shapley values can be interpreted as a cost allocation, a split of the payoff, or, after normalization, as a power index. The Shapley-Shubik power index arises from considering voting games, a particular type of coalitional game. We refer to some textbooks in Game Theory ([11, Chapter IV.3], [19, Section 9.4], [21, Section 14.4]) for a comprehensive treatment.

In a nutshell, the Shapley value of a player  $p$  in a coalitional game is the expected increase in the value of the characteristic function when inserting the player  $p$ , if the players are inserted in an order chosen uniformly at random. We next make this definition precise.

Consider a coalitional game  $(P, v)$ . We denote by  $n$  the number of players and by  $[n]$  the set of integers  $\{1, \dots, n\}$ . A permutation of  $P$  is a bijective map  $\pi: P \rightarrow [n]$ . Let  $\Pi(P)$  be the set of permutations of  $P$ . Each permutation  $\pi \in \Pi(P)$  defines an ordering in  $P$ , where  $\pi(p)$  is the position of  $p$  in that order. We will heavily use this interpretation of



permutations as defining an order in  $P$ . For each element  $p \in P$  and each permutation  $\pi \in \Pi(P)$ , let  $P(\pi, p)$  be the elements of  $P$  before  $p$  in the order defined by  $\pi$ , including  $p$ . Thus  $P(\pi, p) = \{q \in P \mid \pi(q) \leq \pi(p)\}$ . We can visualize  $P(\pi, p)$  as adding the elements of  $P$  one by one, following the order defined by  $\pi$ , until we insert  $p$ . The increment in  $v(\cdot)$  when adding player  $p$  is

$$\Delta(v, \pi, p) = v(P(\pi, p)) - v(P(\pi, p) \setminus \{p\}).$$

The **Shapley value** of player  $p \in P$  in the game  $(P, v)$  is

$$\phi(p, v) = \frac{1}{n!} \sum_{\pi \in \Pi(P)} \Delta(v, \pi, p) = \mathbb{E}_{\pi} [\Delta(v, \pi, p)],$$

where  $\pi$  is picked uniformly at random from  $\Pi(P)$ . It is not difficult to see that the Shapley values indeed split the value  $v(P)$  among the players, that is,  $\sum_{p \in P} \phi(p, v) = v(P)$ .

Since several permutations  $\pi$  define the same subset  $P(\pi, p)$ , the Shapley value of  $p$  is often rewritten as

$$\phi(p, v) = \sum_{S \subset P \setminus \{p\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{p\}) - v(S)).$$

Computing Shapley values from these formulas is computationally infeasible because we have to consider either all the permutations or all the subsets of the players. In fact, there are several natural instances where computing Shapley values is difficult.

### Overview of our contribution

We show that the Shapley values for the `AREA CONVEX HULL` and `AREA ENCLOSED DISK` games can be computed in  $O(n^2)$  and  $O(n^3)$  time, respectively. These problems resemble the models recently considered in *stochastic computational geometry*; see for example [1, 2, 12, 13, 14, 23, 30]. However, there are some key differences in the models. In the most basic model in stochastic computational geometry, sometimes called *unipoint model*, we have a point set and, for each point, a known probability of being actually present, independently for each point. Then we want to analyze a certain functional, for example, the expected area of the minimum enclosing disk.

In our scenario, we have to consider random insertion orders of the points and analyze the expected increase in the value of the characteristic function after the insertion of a fixed point  $p$ . Thus, we have to consider subsets of points constructed according to a different random process. In particular, whether other points precede  $p$  or not in the random order are not independent events. Conditioning on properties of the shape before adding the new point  $p$ , one can get polynomial-time algorithms. We improve the running time by rewriting the Shapley values in a different way and grouping permutations with a similar behavior. Finally, we use arrangements of lines and planes to speed up the computation by an additional linear factor.

For the `AREA ANCHORED RECT`, `AREA BOUNDING BOX`, and `AREA ANCHORED BOUNDING BOX` games we show that Shapley values can be computed in  $O(n^{3/2} \log n)$  time. In the special case where the points form a *chain* (increasing or decreasing  $y$ -coordinate for increasing  $x$ -coordinate), the Shapley values of those games can be computed in  $O(n \log^2 n)$  time. We refer to these games as *axis-parallel* games.

It is relative easy to compute the Shapley values for these axis-parallel games in quadratic time using arrangements of rectangles and the linearity of Shapley values. We will discuss this as an intermediary step towards our solution. However, it is not obvious how to

get subquadratic time. Besides using the linearity of Shapley values, a key ingredient in our algorithms is using convolutions to evaluate at multiple points some special rational functions that keep track of the ratio of permutations with a certain property. The use of algebraic methods in computational geometry is not very common, and there are few results [3, 4, 15, 18] using such techniques in geometric problems.

Our  $O(n^{3/2} \log n)$  algorithm bears some similarities with other existing algorithms with near  $n^{3/2}$  time complexity in the computational geometry literature, for problems like Klee’s measure problem [22]. As in these previous algorithms, we employ an orthogonal subdivision where each region is empty of input points inside, and is “influenced” on average by  $O(\sqrt{n})$  of the points outside. What is new is our combination of such a geometric partitioning scheme with the aforementioned algebraic techniques.

In summary, our results combine fundamental concepts from several different areas and motivated by classical concepts of game theory, we introduce new problems related to stochastic computational geometry and provide efficient algorithms for them.

### Related work

The book by Chalkiadakis, Elkind, and Wooldridge [6] and the chapter by Deng and Fang [8] give a summary of computational aspects of coalitional games. The book by Nisan et al. [20] provides a general overview of the interactions between game theory and algorithms.

In the classical AIRPORT problem considered by Littlechild and Owen [16], we have a set  $P$  of points with positive coordinate on the real line, and the cost of a subset  $Q$  of the points is given by  $\max(Q)$ . It models the portion of the runway that has to be used by each airplane, and Shapley values provide a way to split the cost of the runway among the airplanes. As pointed out before, the points represent agents, in this case airplanes. Several other airport problems are discussed in the survey by Thomson [27]. Using inclusion-exclusion, the airport problem is equivalent to the problem of allocating the length of the smallest interval that contains a set of points on the line. The problems considered in this paper are natural generalizations of the concept of interval when going from one to two dimensions.

Another very common solution concept for a coalitional game  $(P, v)$  is the *core*, defined as

$$\left\{ (x_p)_{p \in P} \in \mathbb{R}^P \mid \forall S \subseteq P : \sum_{p \in S} x_p \geq v(S) \right\}.$$

Sometimes the condition  $\sum_{p \in P} x_p = v(P)$  is also added to the definition. The size of the core is considered a proxy to the stability of the game and, in particular, it is of interest whether the core is nonempty. There are other solution concepts for coalitional games; we refer the reader to the aforementioned general references.

Puerto, Tamir and Perea [24] study the Minimum Radius Location Game in general metric spaces. When specialized to the Euclidean plane, this is equivalent to using the perimeter of the minimum enclosing disk. The paper also considers the  $L_1$ -metric, which is proportional to the perimeter of the minimum enclosing axis-parallel square (after applying a rotation). However, the focus of their work is on understanding the core of the game, and do not discuss the computation of Shapley values. In particular, they show that the Minimum Radius Location Game in the Euclidean plane has nonempty core. Puerto, Tamir and Perea [25] also discuss the Minimum Diameter Location Game, which can be defined for arbitrary metric spaces, but then focus their discussion on graphs.

Faigle et al. [10] consider the TSP coalitional game in general metric spaces, specialize some results to the Euclidean plane, and provide approximate allocations of the costs.

The computation of Shapley values has been considered for several games on graphs. The aforementioned AIRPORT problem can be considered a shortest spanning-path game in a (graph-theoretic) path. Megiddo [17] extended this to trees, while Deng and Papadimitriou [9] discuss a game on arbitrary graphs defined by induced subgraphs. They show that the Shapley values are easy to compute, while characterizing the core is NP-complete.

There is a very large body of follow up works for graphs, but we could not trace other works considering the computation of Shapley values for games defined through planar objects, despite being very natural.

## Assumptions

We will assume general position in the following sense: no two points have the same  $x$  or  $y$  coordinate, no three points are collinear, and no four points are cocircular. In particular, the points are all different. The actual assumptions depend on the game under consideration. It is simple to consider the general case, but it makes the notation more tedious.

We assume a unit-cost real-RAM model of computation. In a model of computation that accounts for bit complexity, time bounds may increase by polynomial factors (even if the input numbers are integers, the outputs may be rationals with large numerators and denominators).

## Organization

Because of space constraints, we limit our presentation to a selection of our results and an overview of the ideas. In this version we do not include our algorithm for computing the Shapley values for the AREAENCLOSINGDISK game in  $O(n^3)$  time. Also, our results about considering the perimeter are not described in this version; they can be found in the full version [5].

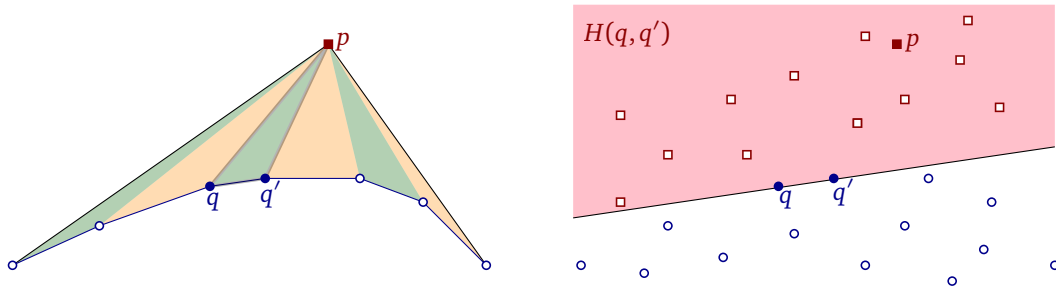
We start with preliminaries in Section 2, where we set the notation, present basic properties of Shapley values, and discuss our needs of algebraic computations. The AREA CONVEX-HULL game is considered in Section 3. In Section 4 we discuss the main ideas for the AREA ANCHORED RECT game and provide some discussion concerning the AREA BOUNDING BOX and AREA ANCHORED BOUNDING BOX games. We conclude in Section 5 with some discussion.

## 2 Preliminaries

All point sets will be in the Euclidean plane  $\mathbb{R}^2$ . The origin is denoted by  $o$ . For a point  $p \in \mathbb{R}^2$ , let  $R_p$  be the axis-parallel rectangle with one corner at the origin  $o$  and the opposite corner at  $p$ . As already mentioned in the introduction, for each  $Q \subset \mathbb{R}^2$  we use  $\text{bb}(Q)$  for the (minimum) axis-parallel bounding box of  $Q$ ,  $\text{med}(Q)$  for a smallest (actually, *the* smallest) disk that contains  $Q$ , and  $\text{CH}(Q)$  for the convex hull of  $Q$ .

We try to use the word *rectangle* when one corner is defined by an input point, while the word *box* is used for more general cases. All rectangles and boxes in this paper are axis-parallel, and we drop the adjective *axis-parallel* when referring to them. An *anchored* box is a box with one corner at the origin.

For a point  $p \in \mathbb{R}^2$  we denote by  $x(p)$  and  $y(p)$  its  $x$ - and  $y$ -coordinate, respectively. We use  $x(Q)$  for  $\{x(q) \mid q \in Q\}$ , and similarly  $y(Q)$ . A set  $P$  of points is a **decreasing chain**, if  $x(p) < x(q)$  implies  $y(p) > y(q)$  for all  $p, q \in P$ . A set  $P$  of points is an **increasing chain** if  $x(p) < x(q)$  implies  $y(p) < y(q)$  for all  $p, q \in P$ .



■ **Figure 2** Left: triangulating the difference between  $CH(P(\pi, p))$  and  $CH(P(\pi, p) \setminus \{p\})$ . Right: in order for  $\Delta pqq'$  to be in  $T(\pi, p)$ ,  $q$  and  $q'$  must appear before  $p$ , which in turn must appear before all other points in the halfplane  $H(q, q')$ .

**Shapley values.** It is easy to see that Shapley values are linear in the characteristic functions. This means that for any two characteristic functions  $v_1, v_2 : 2^P \rightarrow \mathbb{R}$  and for each  $\lambda_1, \lambda_2 \in \mathbb{R}$  we have

$$\phi(p, \lambda_1 v_1 + \lambda_2 v_2) = \lambda_1 \cdot \phi(p, v_1) + \lambda_2 \cdot \phi(p, v_2).$$

The Shapley values when the characteristic function  $v$  is constant over all nonempty subsets of  $P$  is given  $\phi(p, v) = v(P)/n$ .

**Algebraic computations.** For axis-parallel problems we will use the lemma below, which provides multipoint evaluation for a special type of rational functions. The lemma follows from a simple application of convolution, via fast Fourier transform; see the full paper for the proof. (A more general approach of Aronov, Katz and Moroz [4, 18] for arbitrary rational functions gives a slightly worse running time.)

► **Lemma 1.** Let  $b_0, \dots, b_n, \Delta$  be real numbers and consider the rational function

$$R(x) = \sum_{t=0}^n \frac{b_t}{\Delta + t + x}.$$

Given an integer  $\ell > -\Delta$ , possibly negative, and a positive integer  $m$ , we can evaluate  $R(x)$  at all the integer values  $x = \ell, \ell + 1, \dots, \ell + m$  in  $O((n + m) \log(n + m))$  time.

### 3 Convex hull

In this section we consider the area of the convex hull of the points. Consider a fixed set  $P$  of points in the plane. For simplicity we assume that no three points are collinear.

► **Lemma 2.** For each point  $p$  of  $P$  we can compute  $\phi(p, v_{\text{ch}})$  in  $O(n^2)$  time.

**Proof.** For each  $q, q' \in P$  ( $q \neq q'$ ), let  $H(q, q')$  be the open halfplane containing all points to the left of the directed line from  $q$  to  $q'$ . Define  $\text{level}(q, q')$  to be the number of points in  $P \cap H(q, q')$ .

We can decompose the difference between  $CH(P(\pi, p))$  and  $CH(P(\pi, p) \setminus \{p\})$  into a set  $T(\pi, p)$  of triangles (see Figure 2 (left)), where

$$T(\pi, p) = \{\Delta pqq' \mid p \in H(q, q'), q \text{ and } q' \text{ appear before } p \text{ in } \pi, \text{ and no points before } p \text{ in } \pi \text{ lie in } H(q, q')\}.$$

In other words,  $\Delta pqq' \in T(\pi, p)$  if and only if  $p \in H(q, q')$ , and among the  $\text{level}(q, q') + 2$  points in  $H(q, q') \cup \{q, q'\}$ , the two earliest points are  $q$  and  $q'$ , and the third earliest point is  $p$ . See Figure 2 (right). (Note that if  $CH(P(\pi, p)) = CH(P(\pi, p) \setminus \{p\})$ , then  $T(\pi, p)$  is empty.) For fixed  $p, q, q' \in P$  with  $p \in H(q, q')$ , simple counting implies that the probability that  $\Delta pqq' \in T(\pi, p)$  with respect to a random permutation  $\pi$  is exactly

$$\rho(q, q') = \frac{(\text{level}(q, q') - 1)! 2!}{(\text{level}(q, q') + 2)!} = \frac{2}{(\text{level}(q, q') + 2)(\text{level}(q, q') + 1) \text{level}(q, q')}.$$

It follows that the Shapley value of  $p$  is

$$\phi(p, v_{\text{ch}}) = \sum_{\substack{q, q' \in P (q \neq q') \\ \text{with } p \in H(q, q')}} \text{area}(\Delta pqq') \cdot \rho(q, q'). \tag{1}$$

From the formula, we can immediately compute  $\phi(p, v_{\text{ch}})$  for any given  $p \in P$  in  $O(n^2)$  time, if all  $\rho(q, q')$  values have been precomputed.

Each value  $\rho(q, q')$  can be computed from  $\text{level}(q, q')$  using  $O(1)$  arithmetic operations. Thus, precomputing  $\rho(q, q')$  requires precomputing  $\text{level}(q, q')$  for all  $O(n^2)$  pairs  $q, q'$ . In the dual, this corresponds to computing the *levels* of all  $O(n^2)$  vertices in an arrangement of  $n$  lines. The arrangement of  $n$  lines can be constructed in  $O(n^2)$  time [7, Chapter 8], and the levels of all vertices can be subsequently generated by traversing the arrangement in  $O(1)$  time per vertex. ◀

Naively applying Lemma 2 to all points  $p \in P$  gives  $O(n^3)$  total time. We can speed up the algorithm by a factor of  $n$ :

► **Theorem 3.** *The Shapley values of the AREACONVEXHULL game for  $n$  points can be computed in  $O(n^2)$  time.*

**Proof.** Let  $p = (x, y) \in P$ . Observe that for fixed  $q, q' \in P$  ( $q \neq q'$ ), if  $p \in H(q, q')$ , then  $\text{area}(\Delta pqq')$  is a linear function in  $x$  and  $y$  and can thus be written as  $a(q, q')x + b(q, q')y + c(q, q')$ . Let  $A(q, q') = a(q, q') \cdot \rho(q, q')$ ,  $B(q, q') = b(q, q') \cdot \rho(q, q')$ , and  $C(q, q') = c(q, q') \cdot \rho(q, q')$ . (As noted earlier, we can precompute all the  $\rho(q, q')$  values in  $O(n^2)$  time from the dual arrangement of lines.) By (1),

$$\phi(p, v_{\text{ch}}) = \sum_{\substack{q, q' \in P (q \neq q') \\ \text{with } p \in H(q, q')}} (A(q, q')x + B(q, q')y + C(q, q')) = \mathcal{A}(p)x + \mathcal{B}(p)y + \mathcal{C}(p),$$

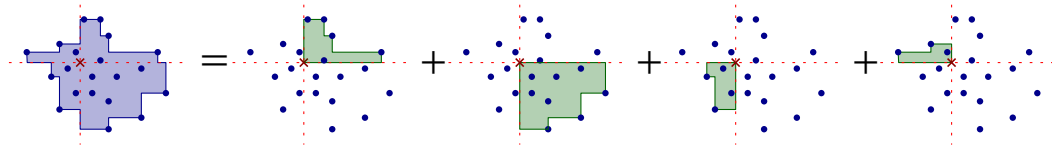
where

$$\mathcal{A}(p) = \sum_{\substack{q, q' \in P (q \neq q') \\ \text{with } p \in H(q, q')}} A(q, q'), \quad \mathcal{B}(p) = \sum_{\substack{q, q' \in P (q \neq q') \\ \text{with } p \in H(q, q')}} B(q, q'), \quad \mathcal{C}(p) = \sum_{\substack{q, q' \in P (q \neq q') \\ \text{with } p \in H(q, q')}} C(q, q').$$

We describe how to compute  $\mathcal{A}(p)$ ,  $\mathcal{B}(p)$ , and  $\mathcal{C}(p)$  for all  $p \in P$  in  $O(n^2)$  total time. Afterwards, we can compute  $\phi(p, v_{\text{ch}})$  for all  $p \in P$  in  $O(n)$  additional time.

The problem can be reduced to 3 instances of the following:

Given a set  $P$  of  $n$  points in the plane, and given  $O(n^2)$  lines each through 2 points of  $P$  and each assigned a weight, compute for all  $p \in P$  the sum of the weights of all lines below  $p$  (or similarly all lines above  $p$ ).



■ **Figure 3** It is enough to consider the cases where all points are in a quadrant.

In the dual, the problem becomes:

Given a set  $L$  of  $n$  lines in the plane, and given  $O(n^2)$  vertices in the arrangement, each assigned a weight, compute for all lines  $\ell \in L$  the sum  $S(\ell)$  of the weights of all vertices below  $\ell$ .

To solve this problem, we could use known data structures for halfplane range searching, but a direct solution is simpler. First construct the arrangement in  $O(n^2)$  time. Given  $\ell, \ell' \in L$ , define  $S(\ell, \ell')$  to be the sum of the weights of all vertices on the line  $\ell'$  that are below  $\ell$ . For a fixed  $\ell' \in L$ , we can precompute  $S(\ell, \ell')$  for all  $\ell \in L \setminus \{\ell'\}$  in  $O(n)$  time, since these values correspond to prefix or suffix sums over the sequence of weights of the  $O(n)$  vertices on the line  $\ell'$ . The total time for all lines  $\ell' \in L$  is  $O(n^2)$ .

Afterwards, for each  $\ell \in L$ , we can compute  $S(\ell)$  in  $O(n)$  time by summing  $S(\ell, \ell')$  over all  $\ell' \in L \setminus \{\ell\}$  and dividing by 2 (since each vertex is counted twice). The total time for all  $\ell \in L$  is  $O(n^2)$ . ◀

#### 4 Axis-parallel problems

Most of this section is dedicated to the AREAANCHOREDRECT game defined by the characteristic function  $v_{ar}$ . Towards the end we discuss the additional challenges to handle the AREABOUNDINGBOX and AREAANCHOREDBOUNDINGBOX games. Unless stated explicitly, the game under consideration is the AREAANCHOREDRECT.

For the AREAANCHOREDRECT game, it is easy to see that one can focus on the special case where all the points are in a quadrant; see Figure 3.

##### 4.1 Notation for axis-parallel problems

Consider a fixed set  $P$  of points in the positive quadrant. In the notation we drop the dependency on  $P$ . For simplicity, we assume general position: no two points have the same  $x$ - or  $y$ -coordinate. We first introduce some notation.

For each point  $q$  of the plane, we use the “cardinal directions” to define subsets of points in quadrants with apex at  $q$ :

$$\begin{aligned} \text{NW}(q) &= \{p \in P \mid x(p) \leq x(q), y(p) \geq y(q)\}, \\ \text{NE}(q) &= \{p \in P \mid x(p) \geq x(q), y(p) \geq y(q)\}, \\ \text{SE}(q) &= \{p \in P \mid x(p) \geq x(q), y(p) \leq y(q)\}. \end{aligned}$$

We use lowercase to denote their cardinality:  $\text{nw}(q) = |\text{NW}(q)|$ ,  $\text{ne}(q) = |\text{NE}(q)|$  and  $\text{se}(q) = |\text{SE}(q)|$ . See Figure 4, left.

Let  $x_1 < \dots < x_n$  denote the  $x$ -coordinates of the points of  $P$ , and let  $y_1 < \dots < y_n$  be their  $y$ -coordinates. We also set  $x_0 = 0$  and  $y_0 = 0$ . For each  $i, j \in [n]$  we use  $w_i = x_i - x_{i-1}$  (for *width*) and  $h_j = y_j - y_{j-1}$  (for *height*).

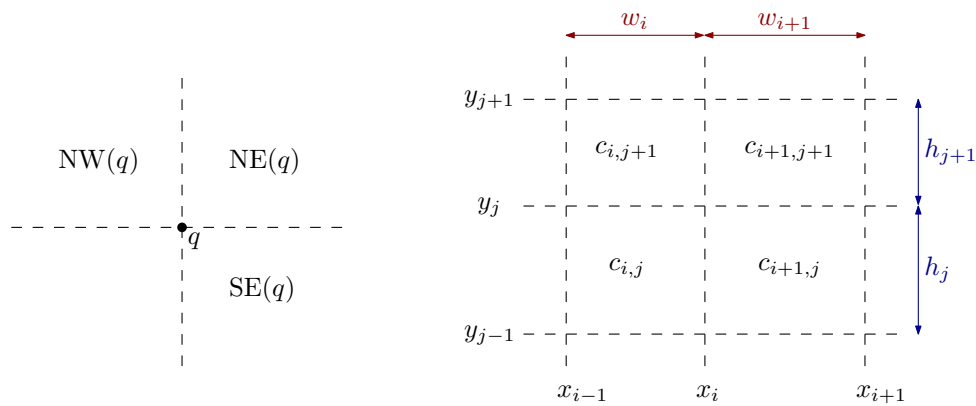


Figure 4 Left: the quadrants to define NW( $q$ ), NE( $q$ ) and SE( $q$ ). Right: cells of  $\mathcal{A}$ .

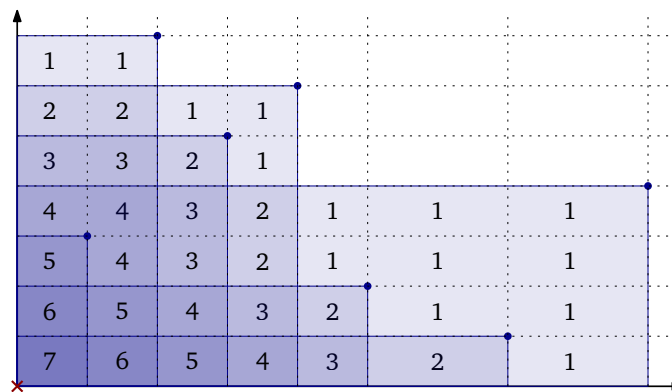


Figure 5 The non-zero counters  $ne(c)$  for the bounded cells  $c$  of  $\mathcal{A}$ . The intensity of the color correlates with the counter  $ne(c)$ .

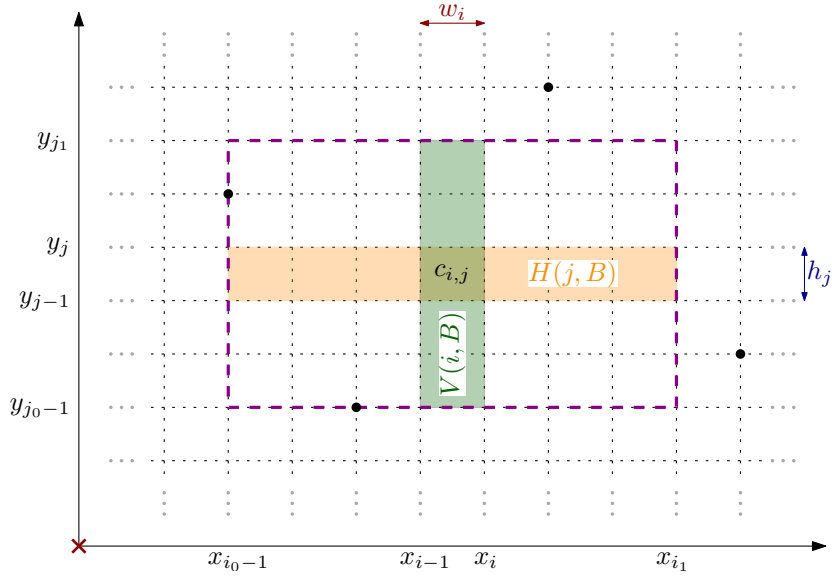
Let  $L$  be the set of horizontal and vertical lines that contain some point of  $P$ . We add to  $L$  both axes. The lines in  $L$  define a partition of the plane into cells, usually called the arrangement and denoted by  $\mathcal{A} = \mathcal{A}(L)$ . The (2-dimensional) cells of  $\mathcal{A}$  are open sets whose closure is a rectangle, possibly unbounded in some direction. We are only interested in the bounded cells, and with a slight abuse of notation, we use  $\mathcal{A}$  for the set of bounded cells. We denote by  $c_{i,j}$  the cell between the vertical lines  $x = x_{i-1}$  and  $x = x_i$  and the horizontal lines  $y = y_{j-1}$  and  $y = y_j$ . Note that  $c_{i,j}$  is the interior of a rectangle with width  $w_i$  and height  $h_j$ . See Figure 4, right.

Since NE( $q$ ) is constant over each 2-dimensional cell  $c$  of  $\mathcal{A}$ , we can define NE( $c$ ), for each cell  $c \in \mathcal{A}$ . The same holds for NW( $c$ ), SE( $c$ ), ne( $c$ ), nw( $c$ ) and se( $q$ ). See Figure 5.

A **block** is a set of cells  $B = B(i_0, i_1, j_0, j_1) = \{c_{i,j} \mid i_0 \leq i \leq i_1, j_0 \leq j \leq j_1\}$  for some indices  $i_0, i_1, j_0, j_1$ , with  $1 \leq i_0 \leq i_1 \leq n$  and  $1 \leq j_0 \leq j_1 \leq n$ . The number of columns and rows in  $B$  is  $i_1 - i_0 + 1 + j_1 - j_0 + 1 = O(i_1 - i_0 + j_1 - j_0)$ . A block  $B$  is **empty** if no point of  $P$  is on the boundary of at least three cells of  $B$ . Equivalently,  $B$  is empty if no point of  $P$  is in the interior of the union of the closure of the cells in  $B$ . See Figure 6 for an example.

We will be using maximal rows and columns within a block  $B$  to compute some partial information. Thus, for each block  $B$  and each index  $i$ , we define the **vertical slab**  $V(i, B) = \{c_{i,j} \mid 1 \leq j \leq n, c_{i,j} \in B\}$ . Similarly, for each block  $B$  and each index  $j$ , we define the **horizontal slab**  $H(j, B) = \{c_{i,j} \mid 1 \leq i \leq n, c_{i,j} \in B\}$ . Such slabs are meaningful only for





■ **Figure 6** An *empty* block  $B = B(i_0, i_1, j_0, j_1)$  with a vertical and a horizontal slab shaded.

indices within the range that defines the slab. We call them the **slabs within**  $B$ .

## 4.2 Interpreting Shapley values geometrically

First, we reduce the problem of computing Shapley values to a neat geometric problem. The following result can be shown by decomposing the game into several games, one per cell of  $\mathcal{A}$ , and using the linearity of Shapley values.

► **Lemma 4.** *If  $P$  is in the positive quadrant, then for each  $p \in P$  we have*

$$\phi(p, v_{\text{ar}}) = \sum_{c \in \mathcal{A}, c \subset R_p} \frac{\text{area}(c)}{\text{ne}(c)}.$$

For each subset  $C$  of cells of  $\mathcal{A}$ , we define

$$\sigma(C) = \sum_{c \in C} \frac{\text{area}(c)}{\text{ne}(c)}$$

(We will only consider sets  $C$  of cells with  $\text{ne}(c) > 0$  for all  $c \in C$ .) Note that we want to compute  $\sigma(\cdot)$  for the sets of cells contained in the rectangles  $R_p$  for all  $p \in P$ .

Using standard tools in computational geometry we can compute the values  $\phi(p, v_{\text{ar}})$  for all  $p \in P$  in near-quadratic time, as follows. An explicit computation of  $\mathcal{A}$  takes quadratic time, and we can use standard data structures for orthogonal range searching ([28], [7, Chapter 5]) to compute  $\text{ne}(c)$  for each cell  $c \in \mathcal{A}$ . Finally, replacing each cell  $c$  by a point  $q_c \in c$  with weight  $w_c = \text{area}(c)/\text{ne}(c)$ , computing  $\phi(p, v_{\text{ar}})$  reduces to computing  $\sum_{q_c \in R_p} w_c$ , which is again an orthogonal range query. An alternative is to use dynamic programming across the cells of  $\mathcal{A}$  to compute  $\text{ne}(c)$  and partial sums of the weights  $w_c$ .

Our objective in the following sections is to improve this result using the correlation between adjacent cells. While at first glance it seems that segment trees [7, Section 10.3] may be useful, the weights are inversely proportional to  $\text{ne}(c)$ , which gives problems

### 4.3 Handling empty blocks

In the following we assume that we have preprocessed  $P$  in  $O(n \log n)$  time such that  $\text{ne}(q)$  can be computed in  $O(\log n)$  time for each point  $q$  given at query time [28]. This is a standard range counting for orthogonal ranges.

When a block  $B$  is empty, then we can use multipoint evaluation to obtain the partial sums  $\sigma(\cdot)$  for each vertical and horizontal slab of the block.

► **Lemma 5.** *Let  $B$  be an empty block with  $k$  columns and rows. We can compute in  $O(k \log n)$  time the values  $\sigma(C)$  for all slabs  $C$  within  $B$ .*

**Proof.** Assume that  $B$  is the block  $B(i_0, i_1, j_0, j_1)$ . We only explain how to compute the values  $\sigma(V(i, B))$  for all  $i_0 \leq i \leq i_1$ . The computation for the horizontal slabs  $\sigma(H(j_0, B)), \dots, \sigma(H(j_1, B))$  is similar.

We look into the first vertical slab  $V(i_0, B)$  and make groups of cells depending on their value  $\text{ne}(\cdot)$ . More precisely, for each  $\ell$  we define  $J(\ell) = \{j \mid j_0 \leq j \leq j_1, \text{ne}(c_{i_0, j}) = \ell\}$ . Let  $\ell_0$  and  $\ell_1$  be the minimum and the maximum  $\ell$  such that  $J(\ell) \neq \emptyset$ , respectively.

Recall that  $h_j$  is the height of the cell  $c_{i_0, j}$  for all  $i_0 \leq i \leq i_1$ . We set up the following rational function with variable  $x$ :

$$R(x) = \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + x}.$$

Setting  $t = \ell - \ell_0$ ,  $b_t = \sum_{j \in J(\ell_0+t)} h_j$  and  $\Delta = \ell_0$ , we have

$$R(x) = \sum_{t=0}^{\ell_1-\ell_0} \frac{b_t}{\Delta + t + x}.$$

Thus, this is a rational function of the shape considered in Lemma 1 with  $\ell_1 - \ell_0 \leq j_1 - j_0 + 1 \leq k$  terms. The coefficients can be computed in  $O(k \log n)$  time because we only need the values  $h_j$  and  $\text{ne}(c_{i_0, j})$  for each  $j$ .

Note that

$$w_{i_0} \cdot R(0) = w_{i_0} \cdot \sum_{\ell=\ell_0}^{\ell_1} \sum_{j \in J(\ell)} \frac{h_j}{\ell} = \sum_{j=j_0}^{j_1} \frac{w_{i_0} h_j}{\text{ne}(c_{i_0, j})} = \sum_{j=j_0}^{j_1} \frac{\text{area}(c_{i_0, j})}{\text{ne}(c_{i_0, j})} = \sigma(V(i_0, B)).$$

A similar statement holds for all the other vertical slabs within  $B$ , as follows.

Consider two consecutive vertical slabs  $V(i, B)$  and  $V(i + 1, B)$  within the block  $B$ . Because the block  $B$  is empty, the difference  $\text{ne}(c_{i+1, j}) - \text{ne}(c_{i, j})$  is independent of  $j$ . See Figure 7. It follows that, for each index  $i$  with  $i_0 \leq i \leq i_1$ , there is an integer  $\delta_i$  such that  $\text{ne}(c_{i, j}) = \text{ne}(c_{i_0, j}) + \delta_i$  for all  $j$  with  $j_0 \leq j \leq j_1$ . Moreover, for each  $i$  with  $i_0 \leq i \leq i_1$  and each  $\ell$  with  $\ell_0 \leq \ell \leq \ell_1$ , the value of  $\text{ne}(c_{i, j})$  is constant over all  $j \in J(\ell)$ . Therefore, for each  $j \in J(\ell)$  we have  $\text{ne}(c_{i, j}) = \ell + \delta_i$ .

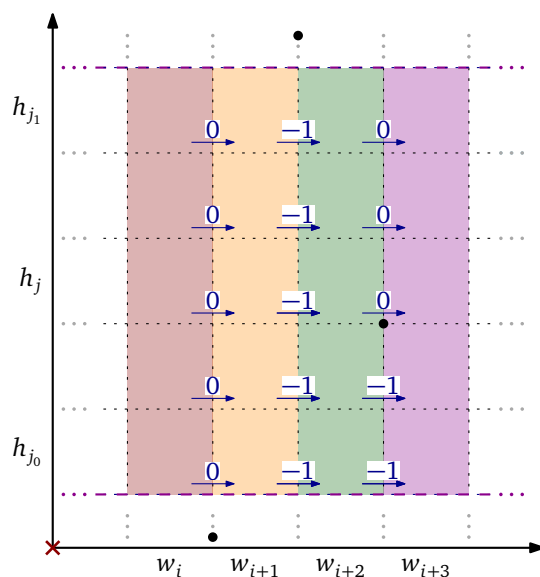
Each value  $\delta_i$  can be obtained using that  $\delta_i = \text{ne}(c_{i, j_0}) - \text{ne}(c_{i_0, j_0})$ . This means that the values  $\delta_{i_0}, \dots, \delta_{i_1}$  can be obtained in  $O(k \log n)$  time.

Now we note that, for each index  $i$  with  $i_0 \leq i \leq i_1$ , we have

$$w_i \cdot R(\delta_i) = w_i \cdot \sum_{\ell=\ell_0}^{\ell_1} \sum_{j \in J(\ell)} \frac{h_j}{\ell + \delta_i} = \sum_{j=j_0}^{j_1} \frac{w_i h_j}{\text{ne}(c_{i, j})} = \sum_{j=j_0}^{j_1} \frac{\text{area}(c_{i, j})}{\text{ne}(c_{i, j})} = \sigma(V(i, B)).$$

We use Lemma 1 to evaluate the  $i_1 - i_0 + 1 \leq k$  values  $R(\delta_i)$ , where  $i_0 \leq i \leq i_1$ , in  $O(k \log k) = O(k \log n)$  time. After this, we get each value  $\sigma(V(i, B)) = w_i \cdot R(\delta_i)$  in constant time. ◀

## 20:12 Computing Shapley Values in the Plane



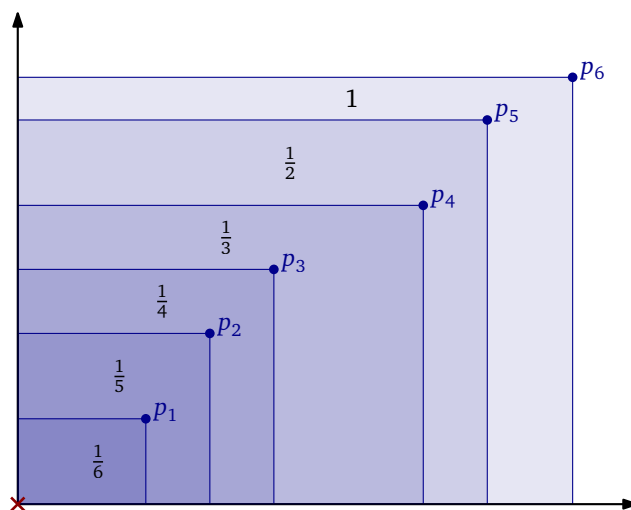
■ **Figure 7** Changes in the values of  $ne(\cdot)$  when passing from a vertical slab to the next one. The rightmost transition shows the need to deal with empty blocks for our argument.

### 4.4 Chains

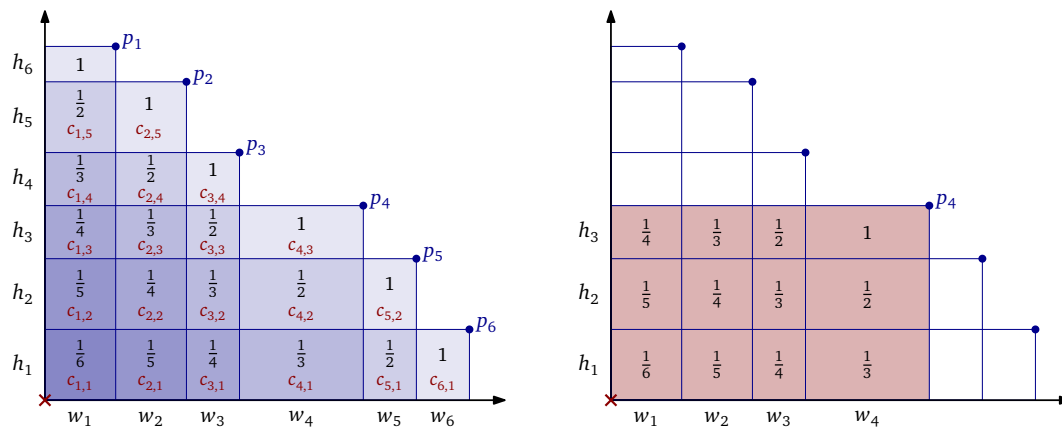
In this section we consider the case where the points are a chain. As discussed before, it is enough to consider that  $P$  is in the positive quadrant. After sorting, we can assume without loss of generality that the points of  $P$  are indexed so that  $0 < x(p_1) < \dots < x(p_n)$ .

For increasing chains, the problem is actually an AIRPORT game in disguise and Shapley values can be computed using prefix sums. See Figure 8.

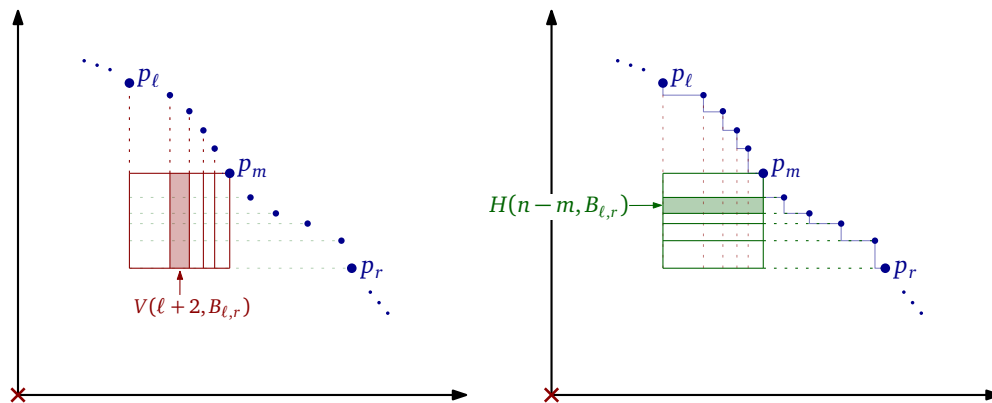
► **Lemma 6.** *If  $P$  is an increasing chain in the positive quadrant, then we can compute the Shapley values of the AREAANCHOREDRECT game in  $O(n \log n)$ .*



■ **Figure 8** Increasing chain. Each region is marked with the multiplicative weight for its area.



■ **Figure 9** Decreasing chain. Left: Each cell  $c_{i,j}$  is marked with the multiplicative weight  $\frac{1}{ne(c)}$  for its area. Right: the cells whose contribution we have to add for the point  $p_4$ .



■ **Figure 10** Vertical and horizontal slabs for the divide-and-conquer.

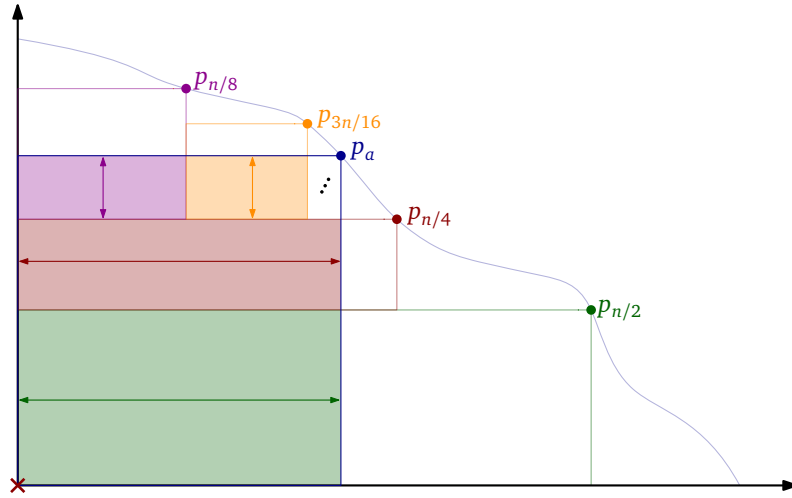
It remains the more interesting case, when the chain is decreasing. See Figure 9 for an example. In this case, the values  $ne(c)$  have a special structure, we do not need data structures to obtain  $ne(c_{i,j})$ , and the proof of Lemma 5 can be simplified.

We use a divide-and-conquer paradigm considering certain empty blocks defined by two indices  $\ell$  and  $r$ , where  $\ell < r$ . Since the indexing of rows is not the most convenient in this case, it is better to introduce the notation  $B_{\ell,r}$  for the block  $B(\ell + 1, m, n - r + 2, n - m + 1)$ , where  $m = m(\ell, r) = \lfloor (\ell + r)/2 \rfloor$ . Initially we will have  $\ell = 0$  and  $r = n + 1$ , which means that we start with the block  $B_{0,n+1} = B(1, m, 1, m)$ . See Figure 10 for a generic case.

The blocks  $B_{\ell,r}$  that we consider are in bijection with the intervals  $(\ell, r)$  that correspond to nodes in a binary search tree with values  $\{0, \dots, n + 1\}$  at the leaves. For each block  $B_{\ell,r}$  considered during the recursion, we compute  $\sigma(C)$  for each slab  $C$  within  $B_{\ell,r}$  using Lemma 5. Furthermore, within each block we compute the sums of  $\sigma(\cdot)$  for prefixes of horizontal and vertical slabs within the block. This means that, for a block  $B_{\ell,r}$ , we compute

$$\sigma(V(\ell + 1, B_{\ell,r})), \sigma(V(\ell + 1, B_{\ell,r}) \cup V(\ell + 2, B_{\ell,r})), \dots, \sigma(B_{\ell,r}),$$

and a similar prefix sums for horizontal slabs.



■ **Figure 11** Expressing  $R_{p_a}$  as the union of  $O(\log n)$  prefixes of slabs within blocks.

For each point  $p_a \in P$ , the rectangle  $R_{p_a}$  is the disjoint union of  $O(\log n)$  prefixes of slabs within blocks considered in the algorithm. Whether we use a column prefix or a row prefix of a block  $B(\ell, r)$  depends on the relative order of the index  $a$  of the point and the median index  $m = m(\ell, r)$ . See Figure 11. Thus,  $\sigma(R_{p_a})$  can be computed as the sum of  $O(\log n)$  values that are already computed. This approach leads to the following result.

► **Lemma 7.** *If  $P$  is a decreasing chain with  $n$  points in the positive quadrant, then we can compute the Shapley values of the AREAANCHOREDRECT game in  $O(n \log^2 n)$  time.*

When the point set is a chain over different quadrants, we can use symmetry to reduce it to a few problems over the positive quadrant, possibly changing the increasing/decreasing character of chain. From Lemmas 6 and 7 we then obtain the following.

► **Theorem 8.** *If  $P$  is a chain with  $n$  points, then we can compute the Shapley values of the AREAANCHOREDRECT game in  $O(n \log^2 n)$  time.*

## 4.5 General point sets

We consider now the general case, where the points do not form a chain, as in Figure 5. Like before, we restrict the discussion to the case where  $P$  is in the positive quadrant.

In this scenario we consider horizontal bands. A horizontal **band**  $B$  is the block between two horizontal lines. Thus,  $B = \{c_{i,j} \mid j_0 \leq j \leq j_1\}$  for some indices  $1 \leq j_0 \leq j_1 \leq n$ . See Figure 12. We keep using the notation introduced for blocks. Thus, for each  $i \in [n]$ , let  $V(i, B)$  be the vertical slab with the cells  $c_{i,j} \in B$ . Let  $P_B$  be the points of  $P$  that are the top-right corner of some cell of  $B$ . We use  $k_B = |P_B|$ . Because of our assumption on general position,  $k_B = j_1 - j_0 + 1$  and thus  $k_B$  is precisely the number of horizontal slabs in the band  $B$ . Furthermore, for each point  $p \in P_B$  we define the rectangle  $R(p, B)$  as the cells of  $B$  to the left and bottom of  $p$ . Formally  $R(p, B) = \{c \in B \mid c \subset R_p\}$ .

Each band can be decomposed further into  $O(k_B)$  empty blocks, and for each such block we can use Lemma 5 to compute  $\sigma(C)$  for each slab  $C$  within each block. We can then compute prefix sums of the values  $\sigma(\cdot)$  for the horizontal and vertical slabs within each block, and express  $\sigma(R(p, B))$ , for each single  $p \in P_B$ , as the sum of  $O(k_B)$  prefixes of rows, at most one per block. This leads to the following.

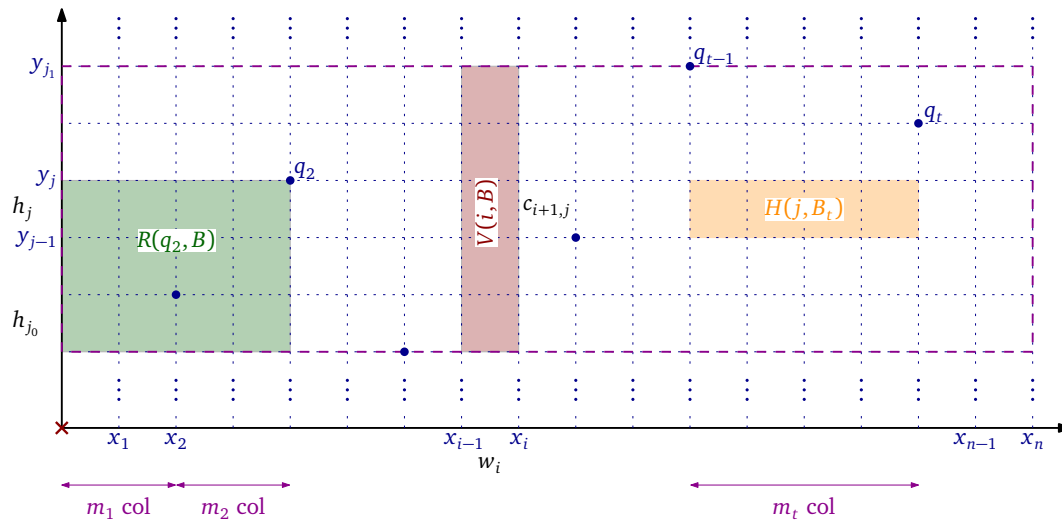


Figure 12 Notation for a band  $B$ .

► **Lemma 9.** For a band  $B$  with  $k_B$  rows we can compute in  $O((k_B)^2 + n) \log n$  time  $\sigma(V(i, B))$ , for all  $i \in [n]$ , and  $\sigma(R(p, B))$ , for all  $p \in P_B$ .

Finally, we can express  $\sigma(R_p)$  as the sum of at most one prefix sum of vertical slabs per band, and  $\sigma(R(p, B))$  for the band  $B$  that contains  $p$ . Using  $O(\sqrt{n})$  bands, each with  $k_B = O(\sqrt{n})$ , this means that we can compute each  $\sigma(R_p)$  as the sum of  $O(\sqrt{n})$  values. Again, the relevant values can be computed with some bookkeeping as prefix sums.

► **Theorem 10.** The Shapley values of the AREAANCHOREDRECT game for  $n$  points can be computed in  $O(n^{3/2} \log n)$  time.

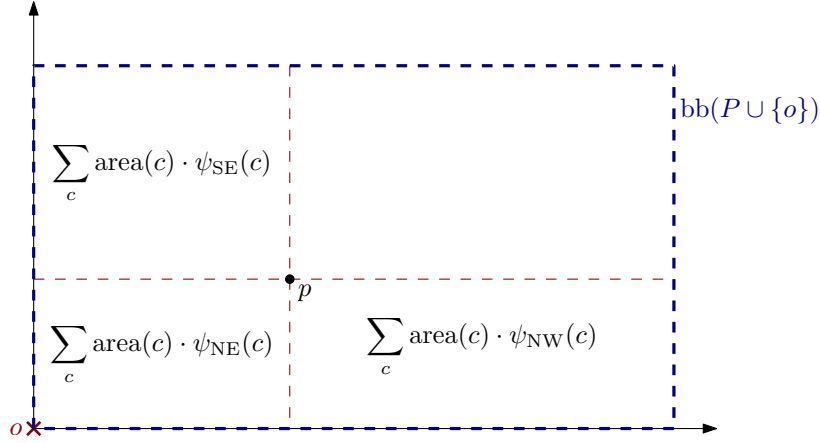
### 4.6 Area of the bounding box

Here we would like to present some of the additional challenges when adapting the algorithm for the AREAANCHOREDRECT game to the AREABOUNDINGBOX game, defined by the characteristic function  $v_{bb}$ . A reduction using inclusion-exclusion shows that it suffices to consider the anchored version, AREAANCHOREDBOUNDINGBOX, when the points are in the positive quadrant.

The geometric interpretation of the Shapley values for the AREAANCHOREDBOUNDINGBOX game becomes slightly more complicated because a cell  $c$  of  $\mathcal{A}$  is inside  $bb(Q \cup \{o\})$  if and only if  $Q$  contains some point in  $NE(c)$  or it contains some point in  $NW(c)$  and in  $SE(c)$ . To obtain a precise description, we define the following values for each cell  $c$  of  $\mathcal{A}$ :

$$\begin{aligned} \psi_{NE}(c) &= \frac{1}{ne(c) + nw(c)} + \frac{1}{ne(c) + se(c)} - \frac{1}{ne(c) + nw(c) + se(c)}, \\ \psi_{NW}(c) &= \frac{1}{ne(c) + nw(c)} - \frac{1}{ne(c) + nw(c) + se(c)}, \\ \psi_{SE}(c) &= \frac{1}{ne(c) + se(c)} - \frac{1}{ne(c) + nw(c) + se(c)}. \end{aligned}$$

The next lemma provides a geometric description of the Shapley values; see Figure 13.



■ **Figure 13** The formula in Lemma 11.

► **Lemma 11.** *If  $P$  is in the positive quadrant, then for each  $p \in P$  the Shapley value  $\phi(p, v_{\text{abb}})$  is*

$$\sum_{c \in \mathcal{A}, p \in \text{NE}(c)} \text{area}(c) \cdot \psi_{\text{NE}}(c) + \sum_{c \in \mathcal{A}, p \in \text{NW}(c)} \text{area}(c) \cdot \psi_{\text{NW}}(c) + \sum_{c \in \mathcal{A}, p \in \text{SE}(c)} \text{area}(c) \cdot \psi_{\text{SE}}(c).$$

Following the paradigm used for the AREAANCHOREDRECT game, we compute

$$\begin{aligned} \sigma_{\text{NE}}(C) &= \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{NE}}(c), \\ \sigma_{\text{NW}}(C) &= \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{NW}}(c), \\ \sigma_{\text{SE}}(C) &= \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{SE}}(c) \end{aligned}$$

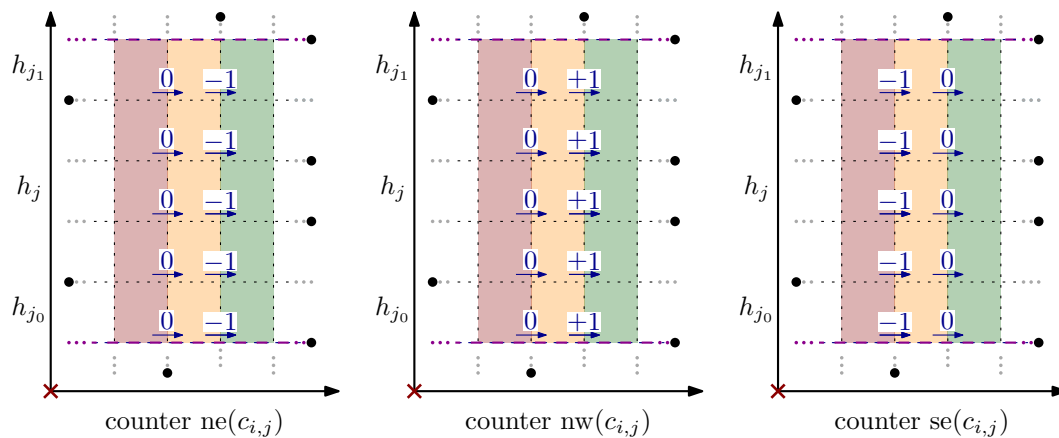
for multiple horizontal and vertical slabs  $C$  inside empty blocks and inside bands. For this we use again the coherence between adjacent slabs of an empty block. More precisely, for two columns within an empty block, the difference between the counters  $\text{ne}()$ ,  $\text{nw}()$  and  $\text{se}()$  is independent of the row. See Figure 14. This is the key idea to obtain an analogue of Lemma 5 for the anchored bounding box.

The rest of the approach used for the AREAANCHOREDRECT game essentially works for the AREAANCHOREDBOUNDINGBOX game, with a few minor modifications. We refer to the full version [5] for additional information. We summarize the final result.

► **Theorem 12.** *The Shapley values of the AREAANCHOREDBOUNDINGBOX and ARE-ABOUNDINGBOX games for  $n$  points can be computed in  $O(n^{3/2} \log n)$  time. If the points form a chain, then we need  $O(n \log^2 n)$  time.*

## 5 Conclusions

In game theory, quite often one considers coalitional games where some point, say the origin, has to be included in the solutions that are considered. This setting is meaningful, for example, when we split the costs to connect to a fixed landmark. For example, we could use the minimum enclosing disk that contains also the origin. We do this for the anchored versions of the games. Our results also hold in this setting through an easy adaptation.



■ **Figure 14** Changes in the counter  $ne(\cdot)$  (left),  $nw(\cdot)$  (center) and  $se(\cdot)$  (right) depending on the position of the points of  $P$ .

The problems we consider here are a new type of stochastic problems in computational geometry. The relation to other problems in stochastic computational geometry is unclear. For example, computing the expected length of the minimum spanning tree (MST) in the plane for a stochastic point set is  $\#P$ -hard [13]. Does this imply the same for the coalitional game based on the length of the Euclidean MST? Is there also a FPRAS for computing the Shapley values of the game defined using the Euclidean MST? Since the length of the Euclidean spanning tree is not monotone, a priori some Shapley values could be potentially 0, which, at least intuitively, makes it harder to get approximations.

In the coalitional games that we consider, the Shapley values that we compute can be interpreted as the relevance of each point within the point set for different geometric concepts. It would be worthwhile to understand whether there is some relation between Shapley values in geometric settings and the concept of depth in point sets, like the Tukey depth. For stochastic points, this relation has been explored and exploited by Agarwal et al. [1].

## References

- 1 Pankaj K. Agarwal, Sarel Har-Peled, Subhash Suri, Hakan Yildiz, and Wuzhou Zhang. Convex Hulls Under Uncertainty. *Algorithmica*, 79(2):340–367, 2017. doi:10.1007/s00453-016-0195-y.
- 2 Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. Range-max queries on uncertain data. *J. Comput. Syst. Sci.*, 94:118–134, 2018. doi:10.1016/j.jcss.2017.09.006.
- 3 Deepak Ajwani, Saurabh Ray, Raimund Seidel, and Hans Raj Tiwary. On Computing the Centroid of the Vertices of an Arrangement and Related Problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *10th International Workshop Algorithms and Data Structures, WADS 2007*, volume 4619 of *Lecture Notes in Computer Science*, pages 519–528. Springer, 2007. doi:10.1007/978-3-540-73951-7\_45.
- 4 Boris Aronov and Matthew J. Katz. Batched Point Location in SINR Diagrams via Algebraic Tools. *ACM Trans. Algorithms*, 14(4):41:1–41:29, 2018. doi:10.1145/3209678.
- 5 Sergio Cabello and Timothy M. Chan. Computing Shapley values in the plane. *CoRR*, abs/1804.03894, 2018. arXiv:1804.03894.
- 6 Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011. doi:10.2200/S00355ED1V01Y201107AIM016.



- 7 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.
- 8 Xiaotie Deng and Qizhi Fang. Algorithmic Cooperative Game Theory. In Altannar Chinchuluun, Panos M. Pardalos, Athanasios Migdalas, and Leonidas Pitsoulis, editors, *Pareto Optimality, Game Theory And Equilibria*, pages 159–185. Springer New York, 2008. doi:10.1007/978-0-387-77247-9\_7.
- 9 Xiaotie Deng and Christos H. Papadimitriou. On the Complexity of Cooperative Solution Concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994. doi:10.1287/moor.19.2.257.
- 10 Ulrich Faigle, Sándor P. Fekete, Winfried Hochstättler, and Walter Kern. On approximately fair cost allocation in Euclidean TSP games. *Operations-Research-Spektrum*, 20(1):29–37, 1998. doi:10.1007/BF01545526.
- 11 Thomas S. Ferguson. Game Theory, 2nd edition, 2014. Electronic text available at [https://www.math.ucla.edu/~tom/Game\\_Theory/Contents.html](https://www.math.ucla.edu/~tom/Game_Theory/Contents.html).
- 12 Martin Fink, John Hershberger, Nirman Kumar, and Subhash Suri. Hyperplane separability and convexity of probabilistic point sets. *JoCG*, 8(2):32–57, 2017. URL: <http://jocg.org/index.php/jocg/article/view/321>.
- 13 Pegah Kamousi, Timothy M. Chan, and Subhash Suri. Stochastic minimum spanning trees in Euclidean spaces. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proceedings of the 27th ACM Symposium on Computational Geometry, SoCG'11*, pages 65–74. ACM, 2011. doi:10.1145/1998196.1998206.
- 14 Pegah Kamousi, Timothy M. Chan, and Subhash Suri. Closest pair and the post office problem for stochastic points. *Comput. Geom.*, 47(2):214–223, 2014. doi:10.1016/j.comgeo.2012.10.010.
- 15 Stefan Langerman. On the Complexity of Halfspace Area Queries. *Discrete & Computational Geometry*, 30(4):639–648, 2003. doi:10.1007/s00454-003-2856-2.
- 16 Stephen C. Littlechild and Guillermo Owen. A Simple Expression for the Shapely Value in A Special Case. *Management Science*, 20(3):370–372, 1973. doi:10.1287/mnsc.20.3.370.
- 17 Nimrod Megiddo. Computational Complexity of the Game Theory Approach to Cost Allocation for a Tree. *Mathematics of Operations Research*, 3(3):189–196, 1978. doi:10.1287/moor.3.3.189.
- 18 Guillaume Moroz and Boris Aronov. Computing the Distance between Piecewise-Linear Bivariate Functions. *ACM Trans. Algorithms*, 12(1):3:1–3:13, 2016. doi:10.1145/2847257.
- 19 Roger B. Myerson. *Game theory - Analysis of Conflict*. Harvard University Press, 1997.
- 20 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- 21 Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.
- 22 Mark H. Overmars and Chee-Keng Yap. New Upper Bounds in Klee’s Measure Problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.
- 23 Pablo Pérez-Lantero. Area and Perimeter of the Convex Hull of Stochastic Points. *Comput. J.*, 59(8):1144–1154, 2016. doi:10.1093/comjnl/bxv124.
- 24 Justo Puerto, Arie Tamir, and Federico Perea. A cooperative location game based on the 1-center location problem. *European Journal of Operational Research*, 214(2):317–330, 2011. doi:10.1016/j.ejor.2011.04.020.
- 25 Justo Puerto, Arie Tamir, and Federico Perea. Cooperative location games based on the minimum diameter spanning Steiner subgraph problem. *Discrete Applied Mathematics*, 160(7-8):970–979, 2012. doi:10.1016/j.dam.2011.07.020.
- 26 Alvin E. Roth, editor. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.
- 27 William Thomson. Cost allocation and airport problems, 2013. Rochester Center for Economic Research Working Paper. Version of 2014 available at [http://www.iser.osaka-u.ac.jp/collabo/20140524/Airport\\_Problems.pdf](http://www.iser.osaka-u.ac.jp/collabo/20140524/Airport_Problems.pdf).

- 28 Dan E. Willard. New Data Structures for Orthogonal Range Queries. *SIAM J. Comput.*, 14:232–253, 1985. doi:10.1137/0214019.
- 29 Eyal Winter. The Shapley value. In R.J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, chapter 53, pages 2025–2054. Elsevier, 1 edition, 2002. doi:10.1016/S1574-0005(02)03016-3.
- 30 Jie Xue, Yuan Li, and Ravi Janardan. On the separability of stochastic geometric objects, with applications. *Comput. Geom.*, 74:1–20, 2018. doi:10.1016/j.comgeo.2018.06.001.



# On the Metric Distortion of Embedding Persistence Diagrams into Separable Hilbert Spaces

Mathieu Carrière

Department of Systems Biology, Columbia University, New York, USA  
mc4660@cumc.columbia.edu

Ulrich Bauer

Department of Mathematics, Technical University of Munich (TUM), Germany  
ulrich.bauer@tum.de

---

## Abstract

---

Persistence diagrams are important descriptors in Topological Data Analysis. Due to the nonlinearity of the space of persistence diagrams equipped with their *diagram distances*, most of the recent attempts at using persistence diagrams in machine learning have been done through kernel methods, i.e., embeddings of persistence diagrams into Reproducing Kernel Hilbert Spaces, in which all computations can be performed easily. Since persistence diagrams enjoy theoretical stability guarantees for the diagram distances, the *metric properties* of the feature map, i.e., the relationship between the Hilbert distance and the diagram distances, are of central interest for understanding if the persistence diagram guarantees carry over to the embedding. In this article, we study the possibility of embedding persistence diagrams into separable Hilbert spaces with bi-Lipschitz maps. In particular, we show that for several stable embeddings into infinite-dimensional Hilbert spaces defined in the literature, any lower bound must depend on the cardinalities of the persistence diagrams, and that when the Hilbert space is finite dimensional, finding a bi-Lipschitz embedding is impossible, even when restricting the persistence diagrams to have bounded cardinalities.

**2012 ACM Subject Classification** Mathematics of computing → Algebraic topology

**Keywords and phrases** Topological Data Analysis, Persistence Diagrams, Hilbert space embedding

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.21

## 1 Introduction

The increase of available data in both academia and industry has been exponential over the past few decades, making data analysis and machine learning ubiquitous in many different fields of science. Topological Data Analysis (TDA) [5] is one specific field of data science, which focuses more on *complex* rather than big data. The general assumption of TDA is that data is actually sampled from geometric or low-dimensional domains, whose topological features are relevant to the analysis. These topological features are usually encoded in a mathematical object called *persistence diagram*, which is roughly a set of points in the plane, each point representing a topological feature whose size is contained in the coordinates of the point. Persistence diagrams have been proved to bring complementary information to other traditional descriptors in many different applications, often leading to large result improvements. This is also due to the *stability* properties of the persistence diagrams, which state that persistence diagrams computed on similar data are also very close in the diagram distances [2, 8, 9].

Unfortunately, the use of persistence diagrams in machine learning methods is not straightforward, since many algorithms expect data to be Euclidean vectors, while persistence diagrams are sets of points with possibly different cardinalities. Moreover, the *diagram distances* used to compare persistence diagrams are computed by means of optimal matchings,

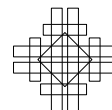


© Mathieu Carrière and Ulrich Bauer;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 21; pp. 21:1–21:15  
Leibniz International Proceedings in Informatics



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



and thus are quite different from Euclidean metrics. The usual way to cope with such difficult data is to use *kernel methods*. A kernel is a symmetric function on the data whose evaluation on a pair of data points equals the scalar product of the images of these points under a *feature map* into a Hilbert space, called the *Reproducing Kernel Hilbert Space* of the kernel. Many algorithms can be *kernelized*, such as PCA and SVM, allowing one to handle non-Euclidean data as soon as either a kernel or a feature map is available.

Hence, the question of defining a feature map into a Hilbert space has been intensively studied in the past few years, and, as of today, various methods have been proposed and implemented, either into finite or infinite dimensional Hilbert spaces [4, 7, 21, 16, 1, 6, 13]. Since persistence diagrams enjoy stability properties, it is also natural to ask the same guarantee for their embeddings. Indeed, various feature maps defined in the literature satisfy a stability property stating that the Hilbert distance between the image of the persistence diagrams is upper bounded by some specific diagram distance, most commonly the 1-Wasserstein diagram distance. In many cases, this upper bound applies only to a restricted set of persistence diagrams with bounded number and bounded range of persistence pairs, and these bounds enter the constant in the stability estimate. However, some unconditional stability results exist as well, e.g., for the Persistence Scale Space feature map [21].

A more difficult question is to prove whether a lower bound also holds or not. As a first step in this direction, a lower bound for the *Sliced Wasserstein distance* was proved in [6], showing that this metric is equivalent to the first diagram distance. Moreover, since the Sliced Wasserstein distance is conditionally negative definite, a Gaussian kernel can be defined with it with Berg's theorem [3]. However, even in this case, the resulting Sliced Wasserstein kernel distance is not equivalent to the Sliced Wasserstein distance, and so the corresponding feature map is not guaranteed to be bi-Lipschitz. Thus, the question remained open in general.

## Contributions

In this article, we consider the general question of the existence of bi-Lipschitz embeddings of persistence diagrams into separable Hilbert spaces. More precisely, we show the following results:

- For several stable feature maps defined in the literature, if such a bi-Lipschitz embedding exists for persistence diagrams with bounded number and range of points, then the ratio between upper and lower bound goes to  $\infty$  as the bounds on the number of points in the persistence diagrams and on their range increase to  $\infty$  (Theorem 3.5 and Proposition 3.9).
- Such a bi-Lipschitz embedding does not exist if the Hilbert space is finite dimensional (Theorem 4.4),

Finally, we also provide experimental evidence of this behavior by computing the metric distortions of various feature maps for persistence diagrams with increasing cardinalities.

## Related work

Feature maps for persistence diagrams can be classified into two different classes, depending on whether the corresponding Hilbert space has finite or infinite dimension.

In the infinite dimensional case, the first attempt was that proposed in [4], in which persistence diagrams are turned into families of  $L^2$  functions, called *landscapes*, by computing the homological rank functions given by the persistence diagram points. Another common way to define a feature map is to see the points of the persistence diagrams as centers of Gaussians with a fixed bandwidth, weighted by the distance of the point to the diagonal.

This is the approach originally advocated in [21], and later generalized in [17], leading to the so-called *Persistence Scale Space* and *Persistence Weighted Gaussian* feature maps. Another possibility is to define a Gaussian-like feature map by using the *Sliced Wasserstein distance* between persistence diagrams, which is conditionally negative definite. This implicit feature map, called the *Sliced Wasserstein* map, was defined in [6].

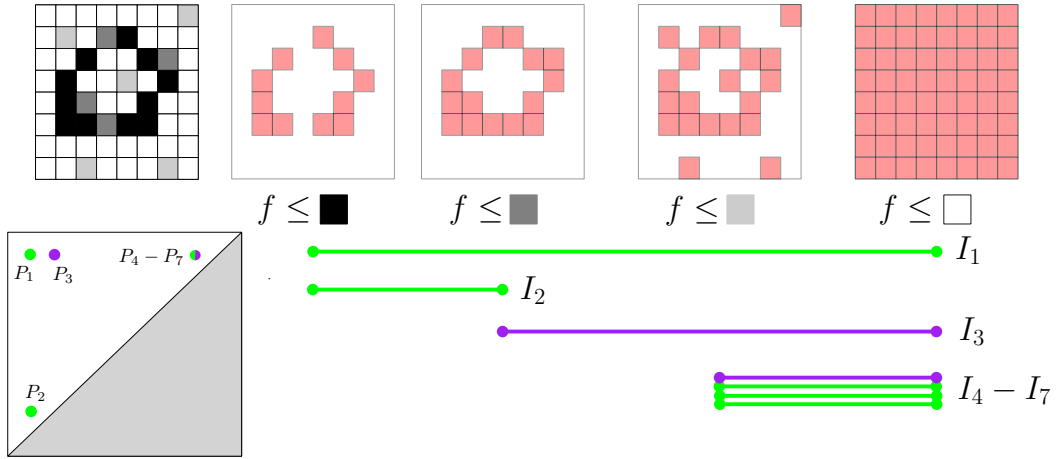
In the finite dimensional case, many different possibilities are available. One may consider evaluating a family of tropical polynomials onto the persistence diagram [14], taking the sorted vector of the pairwise distances between the persistence diagram points [7], or computing the coefficients of a complex polynomial whose roots are given by the persistence diagram points [10]. Another line of work has been proposed in [1] by discretizing the Persistence Scale Space feature map. The idea is to discretize the plane into a fixed grid, and then compute a value for each pixel by integrating Gaussian functions centered on the persistence diagram points. Finally, persistence diagrams have been incorporated in deep learning frameworks in [13], in which Gaussian functions (whose means and variances are optimized by the neural network during training) are integrated against persistence diagrams seen as discrete measures.

## 2 Background

### 2.1 Persistence Diagrams

*Persistent homology* is a technique of TDA, using concepts from algebraic topology, which allows the user to compute and encode topological information of datasets in a compact descriptor called the *persistence diagram*. Given a space  $X$  and a continuous and real-valued function  $f : X \rightarrow \mathbb{R}$ , the persistence diagram of  $f$  can be computed under mild conditions (the function has to be *tame*, see [8] for more details), and consists in a finite set of points with multiplicities in the upper-diagonal half-plane  $\text{Dg}(f) = \{(x_i, y_i)\} \subset \{(x, y) \in \mathbb{R}^2 : y > x\}$ . This set of points is computed from the family of *sublevel sets* of  $f$ , that is the sets of the form  $f^{-1}((-\infty, \alpha])$ , for some  $\alpha \in \mathbb{R}$ . More precisely, persistence diagrams encode the different *topological events* that occur as  $\alpha$  increases from  $-\infty$  to  $\infty$ . Such topological events include creation and merging of connected components and cycles in every dimension. For example, when dealing with a point cloud  $\hat{X} \subset \mathbb{R}^n$ , a common strategy to obtain a persistence diagram from  $\hat{X}$  is to set  $X = \mathbb{R}^n$  and to use the Euclidean distance to  $\hat{X}$  as the function  $f$ . See also Figure 1 for another example.

Intuitively, persistent homology records, for each topological feature that appears in the family of sublevel sets, the value  $\alpha_b$  at which the feature appears, called the *birth value*, and the value  $\alpha_d$  at which it gets merged or filled in, called the *death value*. These values are then used as coordinates for a corresponding point in the persistence diagram. In general, depending on the space  $X$ , there will also be topological features for which there is no finite death value. Such features are however not considered in the context of the present paper. Note that several features may have the same birth and death values, so points in the persistence diagram have multiplicities. Moreover, since  $\alpha_d \geq \alpha_b$ , these points are always located above the diagonal  $\Delta = \{(x, x) : x \in \mathbb{R}\}$ . A general intuition about persistence diagrams is that the distance of a point to  $\Delta$  is a direct measure of its relevance: if a point is close to  $\Delta$ , it means that the corresponding cycle got filled in right after its appearance, thus suggesting that it is likely due to noise in the dataset. On the contrary, points that are far away from  $\Delta$  represent cycles with a significant life span, and are more likely to be relevant for the analysis. We refer the interested reader to [11, 19] for more details about persistent homology.



■ **Figure 1** Example of persistence diagram computation. The space we consider is a blurry image of a zero, and the function  $f$  that we use is the grey level value on each pixel. We show four different sublevel sets of  $f$ . For each sublevel set, the corresponding pixels are displayed in pink color. In the first sublevel set, two connected components are present in the sublevel set, so we start two intervals  $I_1$  and  $I_2$ . In the second one, one connected component got merged to the other, so we stop the corresponding interval  $I_2$ , and a cycle (loop) is created, so we start a third interval  $I_3$ . In the third sublevel set, a new small cycle is created, as well as three more connected components. In the fourth sublevel set, all pixels belong to the set: all cycles are filled in and all connected components are merged together, so all intervals end here. Finally, each interval  $I_k$  is represented as a point  $P_k$  in the plane (using the endpoints as coordinates).

### Notation

Let  $\mathcal{D}$  be the set of persistence diagrams with at most a countable number of points. More formally,  $\mathcal{D}$  can be equivalently defined as a set of multiplicity functions  $\{m : \mathbb{R}^2 \setminus \Delta \rightarrow \mathbb{N} : \text{supp}(m) \text{ is countable}\}$ , where each point  $q \in \text{supp}(m)$  is a point in the corresponding persistence diagram with multiplicity  $m(q)$ . Let  $\mathcal{D}_N$  be the set of persistence diagrams with less than  $N$  points, i.e.,  $\mathcal{D}_N = \{m : \mathbb{R}^2 \setminus \Delta \rightarrow \mathbb{N} : \sum_q m(q) < N\}$ . Let  $\mathcal{D}^L$  be the space of persistence diagrams included in  $[-L, L]^2$ , i.e.,  $\mathcal{D}^L = \{m : \mathbb{R}^2 \setminus \Delta \rightarrow \mathbb{N} : \text{supp}(m) \subset [-L, L]^2\}$ . Finally, let  $\mathcal{D}_N^L$  be the space of persistence diagrams with less than  $N$  points included in  $[-L, L]^2$ , i.e.,  $\mathcal{D}_N^L = \mathcal{D}_N \cap \mathcal{D}^L$ . Obviously, we have the following sequences of (strict) inclusions:  $\mathcal{D}_N^L \subset \mathcal{D}_N \subset \mathcal{D}$ , and  $\mathcal{D}_N^L \subset \mathcal{D}^L \subset \mathcal{D}$ .

### Diagram distances

Persistence diagrams can be efficiently compared using the *diagram distances*, which is a family of distances parametrized by an integer  $p$  that rely on the computation of *partial matchings*. Recall that two persistence diagrams  $\text{Dg}_1$  and  $\text{Dg}_2$  may have different number of points. A *partial matching*  $\Gamma$  between  $\text{Dg}_1$  and  $\text{Dg}_2$  is a subset of  $\text{Dg}_1 \times \text{Dg}_2$ . It comes along with  $\Gamma_1$  (resp.  $\Gamma_2$ ), which is the set of points of  $\text{Dg}_1$  (resp.  $\text{Dg}_2$ ) that are not matched to a point of  $\text{Dg}_2$  (resp.  $\text{Dg}_1$ ) by  $\Gamma$ . The  $p$ -cost of  $\Gamma$  is given as:

$$c_p(\Gamma) = \sum_{(p,q) \in \Gamma} \|p - q\|_\infty^p + \sum_{p \in \Gamma_1} \|p - \Delta\|_\infty^p + \sum_{q \in \Gamma_2} \|q - \Delta\|_\infty^p.$$

The  $p$ -diagram distance is then defined as the cost of the best partial matching:

► **Definition 2.1.** Given two persistence diagrams  $Dg_1$  and  $Dg_2$ , the  $p$ -diagram distance  $d_p$  is defined as:

$$d_p(Dg_1, Dg_2) = \inf_{\Gamma} \sqrt[p]{c_p(\Gamma)}.$$

Note that in the literature, these distances are often called the *Wasserstein distances* between persistence diagrams. Here, we follow the denomination of [6]. In particular, taking a maximum instead of a sum in the definition of the cost,

$$c_{\infty}(\Gamma) = \max_{(p,q) \in \Gamma} \|p - q\|_{\infty} + \max_{p \in \Gamma_1} \|p - \Delta\|_{\infty} + \max_{q \in \Gamma_2} \|q - \Delta\|_{\infty}.$$

allows to add one more distance in the family, the *bottleneck distance*  $d_{\infty}(Dg_1, Dg_2) = \inf_{\Gamma} c_{\infty}(\Gamma)$ .

### Stability

A useful property of persistence diagrams is their *stability* in terms of the data generating the diagrams. Indeed, it is well known in the literature that persistence diagrams computed from close functions are close themselves in the bottleneck distance:

► **Theorem 2.2** ([8, 9]). Given two tame functions  $f, g : X \rightarrow \mathbb{R}$ , one has the following inequality:

$$d_{\infty}(Dg(f), Dg(g)) \leq \|f - g\|_{\infty}. \quad (1)$$

In other words, the map  $Dg$  is 1-Lipschitz. Note that stability results exist as well for the other diagram distances, but these results are weaker than the above Lipschitz condition, and they require more conditions – see [19].

## 2.2 Bi-Lipschitz embeddings.

The main question that we address in this article is the one of preserving the persistence diagram metric properties when using embeddings into Hilbert spaces. For instance, one may ask the images of persistence diagrams under a feature map into a Hilbert space to be stable as well. A natural question is then whether a lower bound also exists, i.e., whether the feature map  $\Phi$  is a *bi-Lipschitz embedding* between  $(\mathcal{D}, d_p)$  and  $\mathcal{H}$ .

► **Definition 2.3.** Let  $(X, d_X)$  and  $(Y, d_Y)$  be two metric spaces. A bi-Lipschitz embedding between  $(X, d_X)$  and  $(Y, d_Y)$  is a map  $\Phi : X \rightarrow Y$  such that there are constants  $0 < A, B < \infty$  such that

$$A d_X(x, x') \leq d_Y(\Phi(x), \Phi(x')) \leq B d_X(x, x')$$

for any  $x, x' \in X$ . The metrics  $d_X$  and  $d_Y$  are called strongly equivalent, and the constants  $A$  and  $B$  are called the lower and upper metric distortion bounds, respectively. If  $A = B = 1$ ,  $\Phi$  is called an isometric embedding.

Note that this definition is equivalent to the commonly used definition, which additionally requires  $A = \frac{1}{B}$ .

► **Remark 2.4.** Finding an isometric embedding of persistence diagrams into a Hilbert space is impossible since geodesics are unique in a Hilbert space, while this is not the case for persistence diagrams, as shown in the proof of Proposition 2.4 in [23].



► **Remark 2.5.** For feature maps that are *bounded*, i.e., those maps  $\Phi$  such that there exists a constant  $C > 0$  for which  $\|\Phi(\text{Dg})\| \leq C$  for all  $\text{Dg}$ , it is obviously impossible to find a bi-Lipschitz embedding. This involves for instance the Sliced Wasserstein (SW) feature map [6], which is defined implicitly from a Gaussian-like function.

### 3 Mapping into separable Hilbert spaces

In our first main result, we use *separability* to determine whether a bi-Lipschitz embedding can exist between the space of persistence diagrams and a Hilbert space.

► **Definition 3.1.** A metric space is called *separable* if it has a dense countable subset.

For instance, the following three Hilbert spaces (equipped with their canonical metrics) are separable:  $\mathbb{R}^n$ ,  $\ell_2$  and  $L_2(\Omega)$ , where  $\Omega$  is separable. The two following results describe well-known properties of separable spaces.

► **Proposition 3.2.** Any subspace of a separable metric space is separable as well.

► **Proposition 3.3.** Let  $(X, d_X)$  and  $(Y, d_Y)$  be two metric spaces, and assume there is a bi-Lipschitz embedding  $\Phi : X \rightarrow Y$ , with Lipschitz constants  $A$  and  $B$ . Then  $X$  is separable if and only if  $\text{im}(\Phi)$  is separable.

The following lemma shows that for a feature map  $\Phi$  which is bi-Lipschitz when restricted to  $\mathcal{D}_N^L$ , the limits of the corresponding constants can actually be used to study the general metric distortion in  $\mathcal{D}$ .

► **Lemma 3.4.** Let  $p \in \mathbb{R}_+^*$  and let  $d$  be a continuous metric on  $(\mathcal{D}, d_p)$ . Let

$$R_N^L = \left\{ \frac{d_p(\text{Dg}, \text{Dg}')}{d(\text{Dg}, \text{Dg}')} : \text{Dg} \neq \text{Dg}' \in \mathcal{D}_N^L \right\},$$

$$A_N^L = \inf R_N^L \quad \text{and} \quad B_N^L = \sup R_N^L.$$

Since  $A_N^L$  is nonincreasing and  $B_N^L$  is nondecreasing with respect to  $N$  and  $L$ , we define

$$A_N = \lim_{L \rightarrow \infty} A_N^L, \quad A^L = \lim_{N \rightarrow \infty} A_N^L, \quad A = \liminf_{N, L \rightarrow \infty} A_N^L,$$

$$B_N = \lim_{L \rightarrow \infty} B_N^L, \quad B^L = \lim_{N \rightarrow \infty} B_N^L, \quad B = \limsup_{N, L \rightarrow \infty} B_N^L,$$

where the limit superior and inferior for  $N, L \rightarrow \infty$  are taken for the nets  $A_N^L$  and  $B_N^L$  over the directed set  $\mathbb{N} \times \mathbb{R}$ . Then the following inequalities hold:

$$A^L d(\text{Dg}, \text{Dg}') \leq d_p(\text{Dg}, \text{Dg}') \leq B^L d(\text{Dg}, \text{Dg}') \quad \text{for all } \text{Dg}, \text{Dg}' \in \mathcal{D}^L,$$

$$A_N d(\text{Dg}, \text{Dg}') \leq d_p(\text{Dg}, \text{Dg}') \leq B_N d(\text{Dg}, \text{Dg}') \quad \text{for all } \text{Dg}, \text{Dg}' \in \mathcal{D}_N,$$

$$A d(\text{Dg}, \text{Dg}') \leq d_p(\text{Dg}, \text{Dg}') \leq B d(\text{Dg}, \text{Dg}') \quad \text{for all } \text{Dg}, \text{Dg}' \in \mathcal{D}.$$

Note that  $A$ ,  $A_N$ ,  $A^L$ ,  $B$ ,  $B_N$  and  $B^L$  may be equal to 0 or  $\infty$ , so it does not necessarily hold that  $d$  and  $d_p$  are strongly equivalent on either  $\mathcal{D}_N$ ,  $\mathcal{D}^L$ , or  $\mathcal{D}$ .

**Proof.** We only prove the last inequality, since the proof extends verbatim to the other two. Pick any two persistence diagrams  $\text{Dg}, \text{Dg}' \in \mathcal{D}$ . Let  $\Gamma = \{(p_i, q_i)\}_{i \in \mathbb{N}}$  be an optimal partial matching achieving  $d_p(\text{Dg}, \text{Dg}')$ , where  $p_i$  (resp.  $q_i$ ) is either in  $\text{Dg}$  (resp.  $\text{Dg}'$ ) or in  $\pi_\Delta(\text{Dg}')$

(resp.  $\pi_\Delta(\text{Dg})$ ), and where  $\pi_\Delta$  is the projection (in the Euclidean norm) onto the diagonal  $\Delta$ . We now define two sequences of persistence diagrams  $\{\text{Dg}_n\}_{n \in \mathbb{N}}$  and  $\{\text{Dg}'_n\}_{n \in \mathbb{N}}$  recursively with  $\text{Dg}_0 = \text{Dg}'_0 = \emptyset$  and

$$\text{Dg}_{n+1} = \begin{cases} \text{Dg}_n & \text{if } p_{n+1} \in \pi_\Delta(\text{Dg}'_n), \\ \text{Dg}_n \cup \{p_{n+1}\} & \text{otherwise,} \end{cases} \quad \text{Dg}'_{n+1} = \begin{cases} \text{Dg}'_n & \text{if } q_{n+1} \in \pi_\Delta(\text{Dg}_n), \\ \text{Dg}'_n \cup \{q_{n+1}\} & \text{otherwise.} \end{cases}$$

Note that  $\Gamma$  might have only a finite number of elements if both  $\text{Dg}$  and  $\text{Dg}'$  have finite cardinalities. In this case, we set  $\{\text{Dg}_n\}_{n \in \mathbb{N}}$  (resp.  $\{\text{Dg}'_n\}_{n \in \mathbb{N}}$ ) to be constant and equal to  $\text{Dg}$  (resp.  $\text{Dg}'$ ) for sufficiently large  $n$ . Moreover, define

$$l_n = \max\{\max\{\|p\|_\infty : p \in \text{Dg}_n\}, \max\{\|q\|_\infty : q \in \text{Dg}'_n\}\},$$

$$s_n = \max\{\text{card}(\text{Dg}_n), \text{card}(\text{Dg}'_n)\},$$

Note that both  $\{l_n\}_{n \in \mathbb{N}}$  and  $\{s_n\}_{n \in \mathbb{N}}$  are nondecreasing. We have  $\text{Dg}_n, \text{Dg}'_n \in \mathcal{D}_{s_n}^{l_n}$  and thus

$$A_{s_n}^{l_n} d(\text{Dg}_n, \text{Dg}'_n) \leq d_p(\text{Dg}_n, \text{Dg}'_n) \leq B_{s_n}^{l_n} d(\text{Dg}_n, \text{Dg}'_n). \tag{2}$$

Now, since  $d_p(\text{Dg}_n, \text{Dg}) \rightarrow 0$  when  $n \rightarrow \infty$ , we have  $d(\text{Dg}_n, \text{Dg}) \rightarrow 0$  by continuity of  $d$ , and similarly  $d(\text{Dg}'_n, \text{Dg}') \rightarrow 0$ . Hence, we have  $d_p(\text{Dg}_n, \text{Dg}'_n) \rightarrow d_p(\text{Dg}, \text{Dg}')$  and  $d(\text{Dg}_n, \text{Dg}'_n) \rightarrow d(\text{Dg}, \text{Dg}')$  with the triangle inequality. We finally obtain the desired inequality by letting  $n \rightarrow \infty$  in (2).  $\blacktriangleleft$

A corollary of the previous results is that even if a feature map taking values in a separable Hilbert space might be bi-Lipschitz when restricted to  $\mathcal{D}_N^L$ , the ratio of upper and lower bound has to go to  $\infty$  as soon as the domain of the feature map is not separable.

► **Theorem 3.5.** *Let  $\Phi : \mathcal{D}_\Phi \rightarrow \mathcal{H}$  be a feature map defined on a non-separable subspace  $\mathcal{D}_\Phi$  of persistence diagrams containing every  $\mathcal{D}_N^L$ , i.e.,  $\mathcal{D}_N^L \subset \mathcal{D}_\Phi$  for each  $N, L$ . Assume  $\Phi$  takes values in a separable Hilbert space  $\mathcal{H}$ , and that  $\Phi$  is bi-Lipschitz on each  $\mathcal{D}_N^L$  with constants  $A_N^L, B_N^L$ . Then  $B_N^L/A_N^L \rightarrow \infty$  as  $N, L \rightarrow \infty$ .*

Note that, by Theorem 12 in [18], if the  $p$ -total persistence, i.e., the  $p$ -diagram distance to the empty diagram, of each element of  $\mathcal{D}_\Phi$  is finite, then  $\mathcal{D}_\Phi$  becomes separable w.r.t.  $d_p$ . In particular, this means that  $\mathcal{D}_N^L$  is separable for each  $N, L$ . Moreover, this shows that Theorem 3.5 applies only to domains  $\mathcal{D}_\Phi$  containing at least one diagram whose  $p$ -total persistence is infinite, in particular, a diagram with an infinite number of points.

However, many feature maps defined in the literature, such as the Persistence Weighted Gaussian feature map [17] or the Landscape feature map [4], are actually defined on such domains, and take values in separable spaces, such as the function space  $L^2(\Omega)$ , where  $\Omega$  is the upper half-plane  $\{(x, y) : x \leq y\}$ . Hence, to illustrate how Theorem 3.5 applies to these feature maps, we now provide two lemmata. In the first one, we define a set  $\mathcal{S}$  which is not separable with respect to  $d_1$ , and in the second one, we show that, nevertheless,  $\mathcal{S}$  is actually included in the domain  $\mathcal{D}_\Phi$  of these feature maps.

► **Lemma 3.6.** *Consider the set of points  $P = \{(k, k + \frac{1}{k}) \in \mathbb{R}^2 : k \in \mathbb{N}\}$ , and define the set  $\mathcal{S}$  of persistence diagrams as the power set  $\mathcal{S} = \mathcal{P}(P)$ . Then  $(\mathcal{S}, d_1)$  is not separable.*

**Proof.** Let  $p_k = (k, k + \frac{1}{k}) \in \mathbb{R}^2$  for all  $k \in \mathbb{N}$ . Then we have  $\mathcal{S} = \{\text{Dg}_u\}_{u \in \mathcal{U}}$ , where  $\mathcal{U} = \{0, 1\}^{\mathbb{N}}$  is the set of sequences with values in  $\{0, 1\}$ , and where  $\text{Dg}_u = \{p_i : i \in \text{supp}(u)\}$ . First note that since the sequences  $u \in \mathcal{U}$  can have infinite support, the spaces  $\mathcal{U}$  and  $\mathcal{S} = \{\text{Dg}_u\}_{u \in \mathcal{U}}$  are not countable.

## 21:8 On the Metric Distortion of Embedding PDs into Separable Hilbert Spaces

Let  $\sim$  be the equivalence relation on  $\mathcal{S}$  defined as

$$\text{Dg}_u \sim \text{Dg}_v \iff |\text{supp}(u) \Delta \text{supp}(v)| < \infty,$$

where  $\Delta$  denotes the symmetric difference of sets. Since the set of sequences with finite support is countable, it follows that each equivalence class  $[\text{Dg}_u]_{\sim}$  is countable as well. In particular, this means that the set of equivalence classes  $\mathcal{S}/\sim$  is uncountable, since otherwise  $\mathcal{S}$  would be countable as a countable union of countable equivalence classes.

We now prove the result by contradiction. Assume that  $\mathcal{S}$  is separable, and let  $\mathcal{S}' \subset \mathcal{S}$  be the corresponding dense countable subset of  $\mathcal{S}$ . Let  $\epsilon > 0$ . Then for each  $u \in \mathcal{U}$ , there is at least one sequence  $u' \in \mathcal{U}$  such that  $\text{Dg}_{u'} \in \mathcal{S}'$  and  $d_1(\text{Dg}_u, \text{Dg}_{u'}) \leq \epsilon$ . We now claim that every such  $u'$  satisfies  $\text{Dg}_{u'} \in [\text{Dg}_u]_{\sim}$ . Indeed, assume  $\text{Dg}_{u'} \notin [\text{Dg}_u]_{\sim}$  and let  $\mathcal{I} = \text{supp}(u') \Delta \text{supp}(u)$ . Then, since  $|\mathcal{I}| = \infty$ , we would have

$$d_1(\text{Dg}_u, \text{Dg}_{u'}) = \sum_{k \in \mathcal{I}} \frac{1}{k} = \infty > \epsilon,$$

which is not possible. Hence, this means that  $|\mathcal{S}'| \geq |\mathcal{S}/\sim|$ . However, we showed that  $\mathcal{S}/\sim$  is uncountable, meaning that  $\mathcal{S}'$  is uncountable as well, which leads to a contradiction, since  $\mathcal{S}'$  is countable by assumption.  $\blacktriangleleft$

We now show that the Persistence Weighted Gaussian and the Landscape feature maps are well-defined on the set  $\mathcal{S}$ . Let us first formally define these feature maps.

► **Definition 3.7.** Given  $p = (u, v) \in \mathbb{R}^2$ ,  $u \leq v$ , let  $\phi_p$  be the piecewise linear hat function defined as

$$\phi_p(t) = \begin{cases} \frac{v-u}{2} \left(1 - \frac{2}{v-u} |t - \frac{u+v}{2}| \right) & \text{if } x \leq t \leq y, \\ 0 & \text{otherwise.} \end{cases}$$

Then, given a persistence diagram  $\text{Dg}$ , let  $\lambda_k : t \mapsto \max_k \{\phi_p(t)\}_{p \in \text{Dg}}$ , where  $\max_k$  denotes the  $k$ -th largest element. The Landscape feature map is defined as:

$$\Phi_{\text{L}} : \text{Dg} \mapsto \bar{\lambda}, \quad \text{where } \bar{\lambda}(x, y) = \begin{cases} \lambda_{\lceil x \rceil}(y) & x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

► **Definition 3.8.** Let  $\omega : \mathbb{R}^2 \rightarrow \mathbb{R}$  be a weight function and  $\sigma > 0$ . The Persistence Weighted Gaussian feature map is defined as:

$$\Phi_{\text{PWG}}^{\omega} : \text{Dg} \mapsto \sum_{p \in \text{Dg}} \omega(p) e^{-\frac{\|\cdot - p\|_2^2}{2\sigma^2}}.$$

► **Proposition 3.9.** Let  $\omega$  be the weight function  $(x, y) \mapsto (y - x)^2$ . Let  $\mathcal{S}$  be the set of persistence diagrams defined in Lemma 3.6. Then:

$$\mathcal{S} \subset \mathcal{D}_{\Phi_{\text{PWG}}^{\omega}} \quad \text{and} \quad \mathcal{S} \subset \mathcal{D}_{\Phi_{\text{L}}}.$$

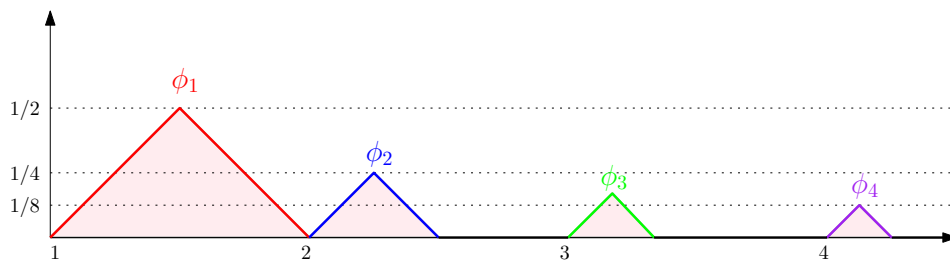
**Proof.** Let  $u_k \in \mathcal{U}$  be the sequence defined with  $u_n = 1$  if  $n \leq k$  and  $u_n = 0$  otherwise. To show the desired result, it suffices to show that  $\{\Phi_{\text{PWG}}^{\omega}(\text{Dg}_{u_k})\}_{k \in \mathbb{N}}$  and  $\{\Phi_{\text{L}}(\text{Dg}_{u_k})\}_{k \in \mathbb{N}}$  are Cauchy sequences in  $L^2(\mathbb{R}^2)$ . Let  $q \geq p \geq 1$ , and let us study  $\|\Phi(\text{Dg}_{u_q}) - \Phi(\text{Dg}_{u_p})\|_{L^2(\mathbb{R}^2)}^2$  for each feature map.

- Case  $\Phi_{\text{PWG}}^\omega$ . We have the following inequalities:

$$\begin{aligned} & \|\Phi_{\text{PWG}}^\omega(\text{Dg}_{u_q}) - \Phi_{\text{PWG}}^\omega(\text{Dg}_{u_p})\|_{L^2(\mathbb{R}^2)}^2 \\ &= \int_{\mathbb{R}^2} \left( \sum_{k=p}^q \frac{1}{k^2} e^{-\frac{\|x-p_k\|_2^2}{2\sigma^2}} \right)^2 dx = \sum_{k=p}^q \sum_{l=p}^q \frac{1}{k^2 l^2} \int_{\mathbb{R}^2} e^{-\frac{\|x-p_k\|_2^2 + \|x-p_l\|_2^2}{2\sigma^2}} dx \\ &= \pi\sigma^2 \sum_{k=p}^q \sum_{l=p}^q \frac{1}{k^2 l^2} e^{-\frac{\|p_k-p_l\|_2^2}{4\sigma^2}} \quad (\text{cf Appendix C in [20]}) \\ &\leq \pi\sigma^2 \left( \sum_{k=p}^q \frac{1}{k^2} \right) \left( \sum_{l=p}^q \frac{1}{l^2} \right) \end{aligned}$$

The result simply follows from the fact that  $\{\sum_{k=1}^n \frac{1}{k^2}\}_{n \in \mathbb{N}}$  is convergent and Cauchy.

- Case  $\Phi_L$ . Since all triangular functions, as defined in Definition 3.7, have disjoint support, it follows that the only non-zero lambda function is  $\lambda_1 = \sum_{n=1}^k \phi_n$ , where  $\phi_n$  is a triangular function defined with  $\phi_n(t) = \frac{1}{2n}(1 - |2n(t - (n + \frac{1}{2n}))|)$  if  $n \leq t \leq n + \frac{1}{n}$  and 0 otherwise. See Figure 2.



■ **Figure 2** Image of  $\text{Dg}_{u_4}$  under  $\Phi_L$ .

Hence, we have the following inequalities:

$$\begin{aligned} & \|\Phi_L(\text{Dg}_{u_q}) - \Phi_L(\text{Dg}_{u_p})\|_{L^2(\mathbb{R}^2)}^2 \\ &= \int_{\mathbb{R}} \left( \sum_{k=p}^q \phi_k(x) \right)^2 dx = \sum_{k=p}^q \sum_{l=p}^q \int_{\mathbb{R}} \phi_k(x)\phi_l(x) dx \\ &= \sum_{k=p}^q \int_{\mathbb{R}} \phi_k(x)^2 dx \leq \sum_{k=p}^q \int_{\mathbb{R}} \phi_k(x) dx = \sum_{k=p}^q \frac{1}{4k^2} \end{aligned}$$

Again, the result follows from the fact that  $\{\sum_{k=1}^n \frac{1}{k^2}\}_{n \in \mathbb{N}}$  is convergent and Cauchy. ◀

Proposition 3.9 shows that Theorem 3.5 applies (with the metric  $d_1$  between persistence diagrams) to the Landscape feature map and to the Persistence Weighted Gaussian feature map with weight function  $(x, y) \mapsto (y - x)^2$  – actually, any weight function that is equivalent to or dominated by  $(y - x)^2$  when  $(y - x)$  goes to 0. Note also that the authors in [16] suggest using weight functions of the form  $(x, y) \mapsto \arctan(C|y - x|^\alpha)$ , which, in this case, means that Theorem 3.5 applies if  $\alpha \geq 2$ . In particular, in this case, any lower bound for the Persistence Weighted Gaussian feature map has to go to 0 when  $N, L \rightarrow \infty$ , since an upper bound exists for this map due to its stability properties – see Corollary 3.5 in [17].

## 4 Mapping into finite-dimensional Hilbert spaces

In our second main result, we show that more can be said about feature maps into  $\mathbb{R}^n$  (equipped with the Euclidean metric), using the so-called *Assouad dimension*. This involves all vectorization methods for persistence diagrams that we described in the related work.

### Assouad dimension

► **Definition 4.1** (Paragraph 10.13 in [12]). *Let  $(X, d_X)$  be a metric space. Given a subset  $E \subset X$  and  $r > 0$ , let  $N_r(E)$  be the least number of open balls of radius less than or equal to  $r$  that can cover  $E$ . The Assouad dimension of  $X$  is:*

$$\dim_A(X, d_X) = \inf\{\alpha > 0 : \exists C > 0 \text{ s.t. } \sup_{x \in X} N_{\beta r}(B(x, r)) \leq C\beta^{-\alpha}, \forall r > 0, \beta \in (0, 1]\}.$$

Intuitively, the Assouad dimension measures the number of open balls of radius  $\beta r$  needed to cover an open ball of radius  $r$ . For example, the Assouad dimension of  $\mathbb{R}^n$  is  $n$ . Moreover, the Assouad dimension is preserved by bi-Lipschitz embeddings.

► **Proposition 4.2** (Lemma 9.6 in [22]). *Let  $(X, d_X)$  and  $(Y, d_Y)$  be metric spaces with a bi-Lipschitz embedding  $\Phi : X \rightarrow Y$ . Then  $\dim_A(X, d_X) = \dim_A(\text{im}(\Phi), d_Y)$ .*

The Assouad dimension is closely related to the familiar notion of *doubling* metric space, where only the number of open balls of radius  $\beta r$  needed to cover an open ball of radius  $r$  is considered: A metric space  $(X, d_X)$  is *doubling* if there is a constant  $M$  such that  $N_{\frac{r}{2}}(B(x, r)) \leq M$  for all  $x \in X$  and  $r > 0$ . In terms of Assouad dimension, this is equivalent to  $\dim_A(X, d_X) < \infty$  [12]. Hence, the property of being doubling is also preserved under bi-Lipschitz maps.

### Non-embeddability

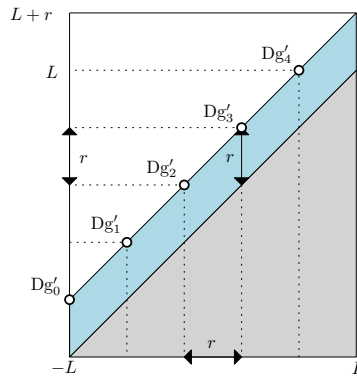
We now show that  $\mathcal{D}_N^L$  cannot be embedded into  $\mathbb{R}^n$  with bi-Lipschitz embeddings. The proof of this fact is a consequence of the following lemma:

► **Lemma 4.3.** *Let  $p \in \mathbb{N} \cup \{\infty\}$ ,  $N \in \mathbb{N}$ , and  $L > 0$ . Then  $\dim_A(\mathcal{D}_N^L, d_p) = \infty$ .*

**Proof.** Let  $B_p$  denote an open ball with  $d_p$ . We want to show that, for any  $\alpha > 0$  and  $C > 0$ , it is possible to find a persistence diagram  $\text{Dg} \in \mathcal{D}_N^L$ , a radius  $r > 0$  and a factor  $\beta \in (0, 1]$  such that the number of open balls of radius at most  $\beta r$  needed to cover  $B_p(\text{Dg}, r)$  is strictly larger than  $C\beta^{-\alpha}$ . To this end, we pick arbitrary  $\alpha > 0$  and  $C > 0$ . The idea of the proof is to define  $\text{Dg}$  as the empty diagram, and to derive a lower bound on the number of balls with radius  $\beta r$  needed to cover  $B_p(\text{Dg}, r)$  by considering a family of persistence diagrams  $\{\text{Dg}'_j\}$  with only one point each, evenly distributed on the line  $\{(x, x+r) : x \in [-L, L]\}$  such that the distance between two consecutive points is  $r$  in the  $\ell_\infty$ -distance. Indeed, the pairwise distance between any two such persistence diagrams is sufficiently large so that they must belong to different balls. Then we can control the number of persistence diagrams, and thus the number of balls, by taking  $r$  sufficiently small.

More formally, choose  $\beta = \frac{1}{4}$ ,  $M = 1 + \lfloor C\beta^{-\alpha} \rfloor > C\beta^{-\alpha}$ , and  $r = 2L/M$ . We want to show that we have at least  $M$  balls in any cover by  $\beta r$ , meaning that  $|\{\text{Dg}_i\}| \geq M$ . To this end, assume we are given an arbitrary cover of  $B_p(\text{Dg}, r)$  with open balls of radius less than  $\beta r$  centered on a family  $\{\text{Dg}_i\}$  as follows:

$$B_p(\text{Dg}, r) \subseteq \bigcup_i B_p(\text{Dg}_i, \beta r). \quad (3)$$



■ **Figure 3** Persistence diagram used in the proof of Lemma 4.3. In this particular example, we have  $M = 5$ .

We now define a particular family of persistence diagrams which all have to lie in different elements of the cover (3). For any  $0 \leq j \leq M - 1$ , we let  $Dg'_j$  denote the persistence diagram containing only the point  $(-L + jr, -L + (j + 1)r)$ , see Figure 3. It is clear that each  $Dg'_j$  is in  $\mathcal{D}_N^L$ .

Moreover, since  $d_p(Dg, Dg'_j) = \frac{r}{2} < r$ , it also follows that  $Dg'_j \in B_p(Dg, r)$ . Hence, according to (3), each  $Dg'_j$  is contained in some ball  $B_p(Dg_i, \beta r)$ . Finally, note that no two  $Dg'_j \neq Dg'_k$  can be contained in the same ball  $B_p(Dg_i, \beta r)$ . Indeed, assuming that  $Dg'_j, Dg'_k \in B_p(Dg_i, \beta r)$ , since the distance between  $Dg'_j$  and  $Dg'_k$  is always obtained by matching their points to the diagonal, we reach a contradiction with the following application of the triangle inequality:

$$d_p(Dg'_j, Dg'_k) = 2^{\frac{1}{p}} \frac{r}{2} \leq d_p(Dg'_j, Dg_i) + d_p(Dg_i, Dg'_k) < 2\beta r = \frac{r}{2}.$$

This observation shows that there are at least  $M$  different open balls in the cover (3), which concludes the proof. ◀

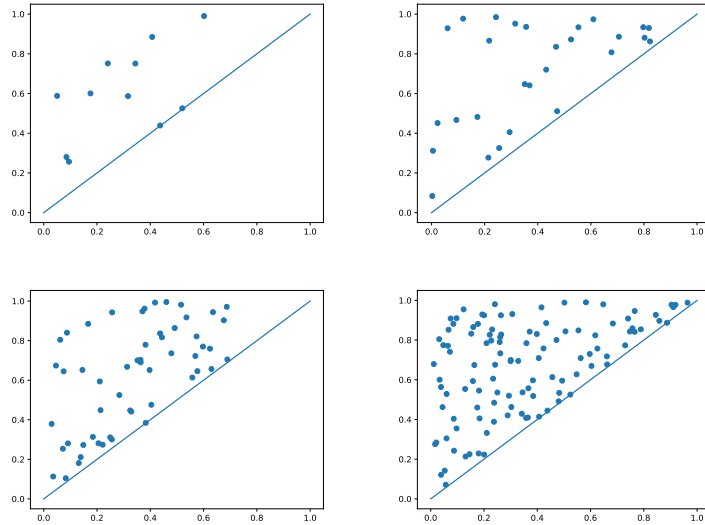
The following theorem is then a simple consequence of Lemma 4.3 and Proposition 4.2:

► **Theorem 4.4.** *Let  $p \in \mathbb{N} \cup \{\infty\}$  and  $n \in \mathbb{N}$ . Then, for any  $N \in \mathbb{N}$  and  $L > 0$ , there is no bi-Lipschitz embedding between  $(\mathcal{D}_N^L, d_p)$  and  $\mathbb{R}^n$ .*

Interestingly, the integers  $N$  and  $n$  are independent in Theorem 4.4: even if one restricts to persistence diagrams with only one point, it is still impossible to find a bi-Lipschitz embedding into  $\mathbb{R}^n$ , whatever  $n$  is.

## 5 Experiments

In this section, we illustrate our main results by computing the lower metric distortion bounds for the main stable feature maps in the literature. We use persistence diagrams with increasing number of points to experimentally observe the convergence of this bound to 0, as described in Theorem 3.5. More precisely, we generate 100 persistence diagrams for each cardinality in a range going from 10 to 1000 by uniformly sampling points in the unit upper half-square  $\{(x, y) : 0 \leq x, y \leq 1, x \leq y\}$ . See Figure 4 for an illustration.



■ **Figure 4** Example of synthetic persistence diagrams with cardinalities 10 (upper left), 30 (upper right), 60 (lower left) and 100 (lower right) generated for the experiment.

Then, we consider the following feature maps:

- the Persistence Weighted Gaussian with unit bandwidth (PWG) [17],
- the Persistence Scale Space with unit bandwidth (PSS) [21],
- the Landscape (LS) [4],
- the Persistence Image with resolution 10 x 10 and unit bandwidth (IM) [1]
- the Topological Vector with 10 dimensions (TV) [7],

Since most of these feature maps enjoy stability properties with respect to the first diagram distance  $d_1$ , we compute the ratios between the metrics in the Hilbert spaces corresponding to these feature maps and  $d_1$ . Moreover, we also look at the ratio induced by the square root of the Sliced Wasserstein distance SW between persistence diagrams [6]. Indeed, if the SW feature map is restricted to a set of persistence diagrams which are close to each other (w.r.t. the SW distance), then the distance in the corresponding Hilbert space is actually equivalent to the square root of the SW distance from the formula:

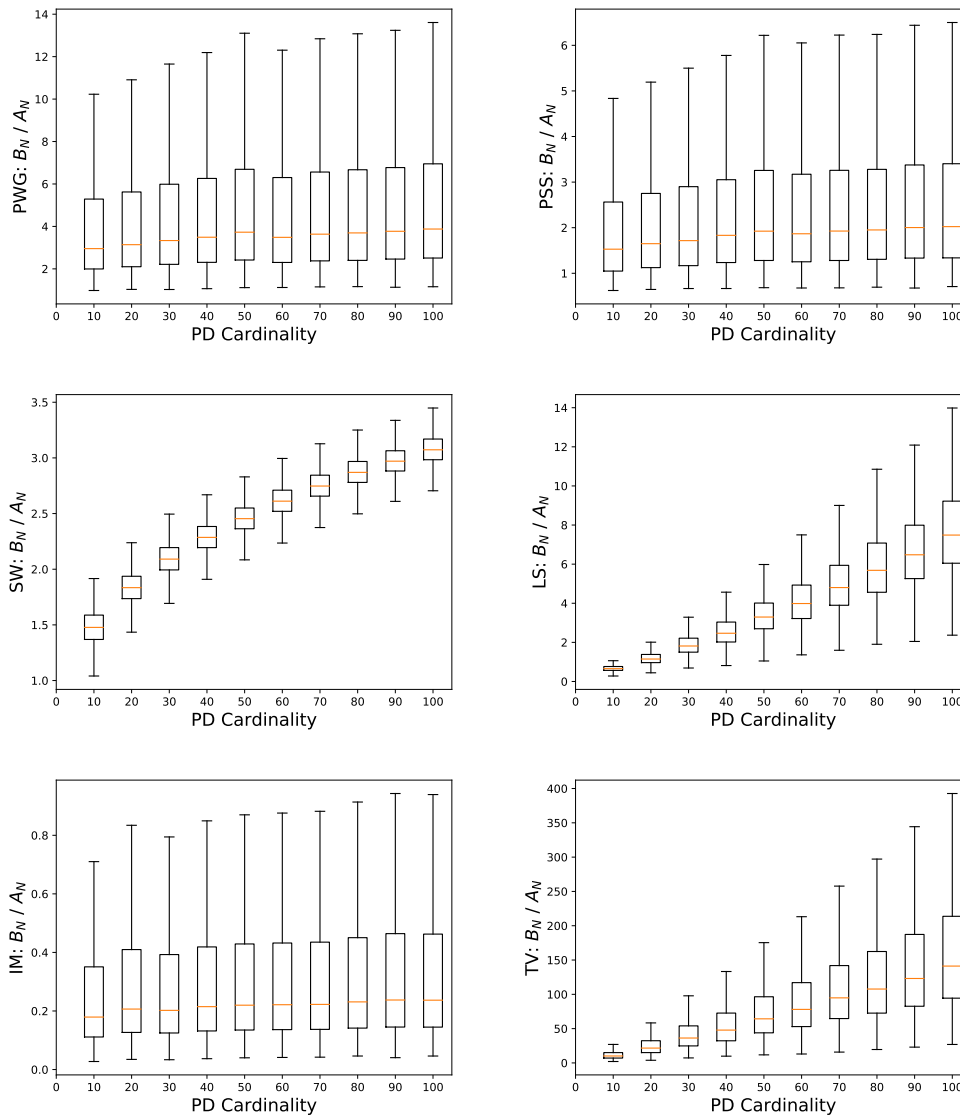
$$\|\Phi_{\text{SW}}(\text{Dg}) - \Phi_{\text{SW}}(\text{Dg}')\| = \sqrt{2(1 - e^{-\text{SW}(\text{Dg}, \text{Dg}')})}$$

Hence, we added the square root of the SW distance in our experiment. All feature maps were computed with the `sklearn-tda` library<sup>1</sup>, which uses `Hera`<sup>2</sup> [15] as a backend to compute the first diagram distances  $d_1$  between pairs of persistence diagrams. These ratios are then displayed as boxplots in Figure 5.

It is clear from Figure 5 that the extreme values of these ratios (the upper tail of the ratio distributions) increase with the cardinality of the persistence diagrams, as expected from Theorem 3.5. This is especially interesting in the case of the Sliced Wasserstein distance since the question whether the lower bound that was proved in [6], which increases with the number of points in the diagrams, was tight or not, i.e., if a lower bound which is oblivious

<sup>1</sup> [https://github.com/MathieuCarriere/sklearn\\_tda](https://github.com/MathieuCarriere/sklearn_tda)

<sup>2</sup> [https://bitbucket.org/grey\\_narn/hera](https://bitbucket.org/grey_narn/hera)



■ **Figure 5** Boxplots of the ratios between distances induced by various feature maps and the first diagram distance  $d_1$ .

to the number of points could be derived, is still open. Hence, it seems from Figure 5 that this is not the case empirically. It is also interesting to notice that the divergence speed of these ratios differ from a feature map to another. More precisely, it seems like the metric distortion bounds increase linearly with the cardinalities for the TV and LS feature maps and the Sliced Wasserstein distance, while it is increasing at a much lower speed for the other feature maps.

## 6 Conclusion

In this article, we provided two important theoretical results about the embedding of persistence diagrams in separable Hilbert spaces, which is a common technique in TDA to feed machine learning algorithms with persistence diagrams. Indeed, most of the recent



attempts have defined feature maps for persistence diagrams into Hilbert spaces and showed these maps were stable with respect to the first diagram distance, and conjectured whether a lower bound holds as well or not. In this work, we proved that this is never the case if the Hilbert space is finite dimensional, and that such a lower bound has to go to zero with the number of points for most other feature maps in the literature. We also provided experiments that confirm this result, by showing a clear increase of the metric distortion with the number of points for persistence diagrams generated uniformly in the unit upper half-square.

---

## References

- 1 Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence Images: A Stable Vector Representation of Persistent Homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017.
- 2 Ulrich Bauer and Michael Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *Journal of Computational Geometry*, 6(2):162–191, 2015.
- 3 Christian Berg, Jens Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Springer, 1984.
- 4 Peter Bubenik. Statistical Topological Data Analysis using Persistence Landscapes. *Journal of Machine Learning Research*, 16:77–102, 2015.
- 5 Gunnar Carlsson. Topology and Data. *Bulletin of the American Mathematical Society*, 46:255–308, 2009.
- 6 Mathieu Carrière, Marco Cuturi, and Steve Oudot. Sliced Wasserstein Kernel for Persistence Diagrams. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.
- 7 Mathieu Carrière, Steve Oudot, and Maks Ovsjanikov. Stable Topological Signatures for Points on 3D Shapes. *Computer Graphics Forum*, 34, 2015.
- 8 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. Springer, 2016.
- 9 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of Persistence Diagrams. *Discrete and Computational Geometry*, 37(1):103–120, 2007.
- 10 Barbara Di Fabio and Massimo Ferri. Comparing Persistence Diagrams Through Complex Vectors. In *Image Analysis and Processing – ICIAP 2015*, pages 294–305, 2015.
- 11 Herbert Edelsbrunner and John Harer. *Computational Topology: an introduction*. AMS Bookstore, 2010.
- 12 Juha Heinonen. *Lectures on Analysis on Metric Spaces*. Springer, 2001.
- 13 Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep Learning with Topological Signatures. In *Advances in Neural Information Processing Systems 30*, pages 1633–1643. Curran Associates, Inc., 2017.
- 14 Sara Kališnik. Tropical Coordinates on the Space of Persistence Barcodes. *Foundations of Computational Mathematics*, 2018.
- 15 Michael Kerber, Dmitriy Morozov, and Arnur Nigmatov. Geometry Helps to Compare Persistence Diagrams. *Journal of Experimental Algorithmics*, 22:1.4:1–1.4:20, September 2017.
- 16 Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka. Persistence Weighted Gaussian Kernel for Topological Data Analysis. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2004–2013, 2016.
- 17 Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka. Kernel Method for Persistence Diagrams via Kernel Embedding and Weight Factor. *Journal of Machine Learning Research*, 18(189):1–41, 2018.
- 18 Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems*, 27(12):124007, 2011.
- 19 Steve Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*. Number 209 in Mathematical Surveys and Monographs. American Mathematical Society, 2015.

- 20 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A Stable Multi-Scale Kernel for Topological Machine Learning. *CoRR*, abs/1412.6821, 2014. [arXiv:1412.6821](#).
- 21 Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A Stable Multi-Scale Kernel for Topological Machine Learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- 22 James C. Robinson. *Dimensions, Embeddings, and Attractors*, volume 186 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 2010.
- 23 Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Fréchet Means for Distributions of Persistence Diagrams. *Discrete and Computational Geometry*, 52(1):44–70, 2014.



# Convex Polygons in Cartesian Products

**Jean-Lou De Carufel**

School of Electrical Engineering and Computer Science, University of Ottawa, Canada  
jdecaruf@uottawa.ca

**Adrian Dumitrescu**

Department of Computer Science, University of Wisconsin-Milwaukee, USA  
dumitres@uwm.edu

**Wouter Meulemans**

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands  
w.meulemans@tue.nl

**Tim Ophelders**

Department of Computational Mathematics, Science and Engineering,  
Michigan State University, East Lansing, MI, USA  
ophelder@egr.msu.edu

**Claire Pennarun**

LIRMM, CNRS & Université de Montpellier, France  
claire.pennarun@gmail.com

**Csaba D. Tóth**

Department of Mathematics, California State University Northridge, Los Angeles, CA, USA  
Department of Computer Science, Tufts University, Medford, MA, USA  
csaba.toth@csun.edu

**Sander Verdonschot**

School of Computer Science, Carleton University, Ottawa, ON, Canada  
sander@cg.scs.carleton.ca

---

## Abstract

We study several problems concerning convex polygons whose vertices lie in a Cartesian product of two sets of  $n$  real numbers (for short, *grid*). First, we prove that every such grid contains a convex polygon with  $\Omega(\log n)$  vertices and that this bound is tight up to a constant factor. We generalize this result to  $d$  dimensions (for a fixed  $d \in \mathbb{N}$ ), and obtain a tight lower bound of  $\Omega(\log^{d-1} n)$  for the maximum number of points in convex position in a  $d$ -dimensional grid. Second, we present polynomial-time algorithms for computing the longest convex polygonal chain in a grid that contains no two points with the same  $x$ - or  $y$ -coordinate. We show that the maximum size of such a convex polygon can be efficiently approximated up to a factor of 2. Finally, we present exponential bounds on the maximum number of convex polygons in these grids, and for some restricted variants. These bounds are tight up to polynomial factors.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Erdős–Szekeres theorem, Cartesian product, convexity, polyhedron, recursive construction, approximation algorithm

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.22

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1812.11332>.

**Funding** *Wouter Meulemans*: Partially supported by the Netherlands eScience Center (NLeSC, 027.015.G02)

*Csaba D. Tóth*: Supported in part by the NSF awards CCF-1422311 and CCF-1423615.

**Acknowledgements** This work was initiated at the 2017 Fields Workshop on Discrete and Computational Geometry (Carleton University, Ottawa, ON, July 31–August 4, 2017).



© Jean-Lou De Carufel, Adrian Dumitrescu, Wouter Meulemans, Tim Ophelders,  
Claire Pennarun, Csaba D. Tóth, and Sander Verdonschot;  
licensed under Creative Commons License CC-BY

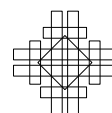
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 22; pp. 22:1–22:17



Leibniz International Proceedings in Informatics

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



## 1 Introduction

Can a convex polygon  $P$  in the plane be reconstructed from the projections of its vertices to the coordinate axes? Assuming that no two vertices of  $P$  share the same  $x$ - or  $y$ -coordinate, we arrive at the following problem: given two sets,  $X$  and  $Y$ , each containing  $n$  real numbers, does the Cartesian product  $X \times Y$  support a convex polygon with  $n$  vertices? We say that  $X \times Y$  *contains* a polygon  $P$  if every vertex of  $P$  is in  $X \times Y$ ; and  $X \times Y$  *supports*  $P$  if it contains  $P$  and no two vertices of  $P$  share an  $x$ - or  $y$ -coordinate. For short, we call the Cartesian product  $X \times Y$  an  $n \times n$  *grid*.

Not every  $n \times n$  grid supports a convex  $n$ -gon. This is the case already for  $n = 5$  (Figure 1). Several interesting questions arise: can we decide efficiently whether an  $n \times n$ -grid supports a convex  $n$ -gon? How can we find the largest  $k$  such that it supports a convex  $k$ -gon? What is the largest  $k$  such that *every*  $n \times n$  grid supports a convex  $k$ -gon? How many convex polygons does an  $n \times n$  grid support, or contain? We initiate the study of these questions for convex polygons, and their higher dimensional variants for convex polyhedra.

**Our results.** We first show that every  $n \times n$  grid supports a convex polygon with  $(1 - o(1)) \log n$  vertices<sup>1</sup>; this bound is tight up to a constant factor: there are  $n \times n$  grids that do not support convex polygons with more than  $4(\lceil \log n \rceil + 1)$  vertices. We generalize our upper and lower bounds to higher dimensions, and show that every  $d$ -dimensional Cartesian product  $\prod_{i=1}^d Y_i$ , where  $|Y_i| = n$  and  $d$  is constant, contains  $\Omega(\log^{d-1} n)$  points in convex position; this bound is also tight apart from constant factors (Section 2). Next, we present polynomial-time algorithms to find a maximum supported convex polygon that is  $x$ - or  $y$ -monotone. We show how to efficiently approximate the maximum size of a supported convex polygon up to a factor of two (Section 3). Finally, we present tight asymptotic bounds for the maximum number of convex polygons supported by an  $n \times n$  grid (Section 4).

**Related work.** Erdős and Szekeres proved, as one of the first Ramsey-type results in combinatorial geometry [16], that for every  $k \in \mathbb{N}$ , a sufficiently large point set in the plane in general position contains  $k$  points in convex position. The minimum cardinality of a point set that guarantees  $k$  points in convex position is known as the Erdős–Szekeres number,  $f(k)$ . They proved that  $2^{k-2} + 1 \leq f(k) \leq \binom{2k-4}{k-2} + 1 = 4^{k(1-o(1))}$ , and conjectured that the lower bound is tight [14]. The current best upper bound, due to Suk [29], is  $f(k) \leq 2^{k(1+o(1))}$ . In other words, every set of  $n$  points in general position in the plane contains  $(1 - o(1)) \log n$  points in convex position, and this bound is tight up to lower-order terms.

In dimension  $d \geq 3$ , the asymptotic growth rate of the Erdős–Szekeres number is not known. By the Erdős–Szekeres theorem, every set of  $n$  points in general position in  $\mathbb{R}^d$  contains  $\Omega(\log n)$  points in convex position (it is enough to find points whose projections onto a generic plane are in convex position). For every constant  $d \geq 2$ , Károlyi and Valtr [19] and Valtr [30] constructed  $n$ -element sets in general position in  $\mathbb{R}^d$  in which no more than  $O(\log^{d-1} n)$  points are in convex position. Both constructions are recursive, and one of them is related to high-dimensional Horton sets [30]. These bounds are conjectured to be optimal apart from constant factors. Our results establish the same  $O(\log^{d-1} n)$  upper bound for Cartesian products  $\prod_{i=1}^d Y_i$ , where  $|Y_i| = n$ , for which it is tight apart from constant factors. However our results do not improve the bounds for points in general position.

---

<sup>1</sup> Throughout this paper all logarithms are in base 2.

Algorithmically, one can find a largest convex cap in a given set of  $n$  points in  $\mathbb{R}^2$  in  $O(n^2 \log n)$  time by dynamic programming [10], and a largest subset in convex position in  $O(n^3)$  time [7, 10]. The same approach can be used for counting the number of convex polygons contained in a given point set [20]. While this approach applies to grids, it is unclear how to include the restriction that each coordinate is used at most once. On the negative side, finding a largest subset in convex position in a point set in  $\mathbb{R}^d$  for  $d \geq 3$  was recently shown to be NP-hard [15].

There has been significant interest in *counting* the number of convex polygons in various point sets. Answering a question of Hammer, Erdős [13] proved that every set of  $n$  points in general position in  $\mathbb{R}^2$  contains  $\exp(\Theta(\log^2 n))$  subsets in convex position, and this bound is the best possible. Bárány and Pach [2] showed that the number of convex polygons in an  $n \times n$  section of the integer lattice is  $\exp(O(n^{2/3}))$ . Bárány and Vershik [3] generalized this bound to  $d$ -dimensions and showed that there are  $\exp(O(n^{(d-1)/(d+1)}))$  convex polytopes in an  $n \times \dots \times n$  section of  $\mathbb{Z}^d$ . Note that the exponent is sublinear in  $n$  for every  $d \geq 2$ . We prove that an  $n \times n$  Cartesian product can contain  $\exp(\Theta(n))$  convex polygons, significantly more than integer grids; our bounds are tight up to polynomial factors.

Motivated by integer programming and geometric number theory, lattice polytopes (whose vertices are in  $\mathbb{Z}^d$ ) have been intensely studied; refer to [1, 4]. However, results for lattices do not extend to arbitrary Cartesian products. Recently, several deep results have been established for Cartesian products in incidence geometry and additive combinatorics [23, 24, 25, 28], while the analogous statements for points sets in general position remain elusive.

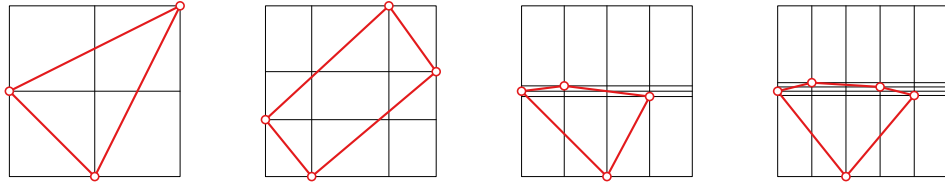
**Definitions.** A polygon  $P$  in  $\mathbb{R}^2$  is *convex* if all of its internal angles are strictly smaller than  $\pi$ . A point set in  $\mathbb{R}^2$  is in *convex position* if it is the vertex set of a convex polygon; and it is in *general position* if no three points are collinear. Similarly, a polyhedron  $P$  in  $\mathbb{R}^d$  is *convex* if it is the convex hull of a finite set of points. A point set in  $\mathbb{R}^d$  is in *convex position* if it is the vertex set of a convex polytope; and it is in *general position* if no  $d + 1$  points lie on a hyperplane. In  $\mathbb{R}^d$ , we say that the  $x_d$ -axis is *vertical*, hyperplanes orthogonal to  $x_d$  are *horizontal*, and understand the above-below relationship with respect to the  $x_d$ -axis. Let  $\mathbf{e}_d$  be a standard basis vector parallel to the  $x_d$ -axis. A point set  $P$  in  $\mathbb{R}^d$  is *full-dimensional* if no hyperplane contains  $P$ .

We consider special types of convex polygons. Let  $P$  be a convex polygon with vertices  $((x_1, y_1), \dots, (x_k, y_k))$  in clockwise order. We say that  $P$  is a *convex cap* if the  $x$ - or  $y$ -coordinates are strictly monotonic, and a *convex chain* if both the  $x$ - and  $y$ -coordinates are strictly monotonic. We distinguish four types of convex caps (resp., chains) based on the monotonicity of the coordinates as follows:

- *convex caps* come in four types  $\{\curvearrowright, \curvearrowleft, \smile, \frown\}$ . We have
  - $P \in \curvearrowright$  if and only if  $(x_i)_{i=1}^k$  strictly increases;
  - $P \in \curvearrowleft$  if and only if  $(y_i)_{i=1}^k$  strictly increases;
  - $P \in \smile$  if and only if  $(x_i)_{i=1}^k$  strictly decreases;
  - $P \in \frown$  if and only if  $(y_i)_{i=1}^k$  strictly decreases;
- *convex chains* come in four types  $\{\curvearrowright, \curvearrowleft, \smile, \frown\}$ . We have
  - $\curvearrowright = \curvearrowleft \cap \smile, \quad \curvearrowleft = \curvearrowright \cap \frown, \quad \smile = \smile \cap \curvearrowleft, \quad \frown = \frown \cap \smile.$

**Initial observations.** It is easy to see that for  $n = 3, 4$ , every  $n \times n$  grid supports a convex  $n$ -gon. However, there exists a  $5 \times 5$  grid that does not support any convex pentagon (cf. Fig. 1). Interestingly, every  $6 \times 6$  grid supports a convex pentagon.

► **Lemma 1.** *Every  $6 \times 6$  grid  $X \times Y$  supports a convex polygon of size at least 5.*



■ **Figure 1** Maximum-size supported convex polygons of respective sizes 3, 4, 4, and 5 in  $n \times n$  grids, where  $n$  is between 3 and 6.

**Proof.** Let  $X' = X \setminus \{\min(X), \max(X)\}$  and  $Y' = Y \setminus \{\min(Y), \max(Y)\}$ . The  $4 \times 4$  grid  $X' \times Y'$  supports a convex chain  $P'$  of size 3 between two opposite corners of  $X' \times Y'$ . Then one  $x$ -coordinate  $x' \in X'$  and one  $y$ -coordinate  $y' \in Y'$  are not used by  $P'$ . Without loss of generality, assume that  $P' \in \uparrow$ . Then the convex polygon containing the points of  $P'$  and  $(x', \min(Y))$  and  $(\max(X), y')$  is a supported convex polygon of size 5 on  $X \times Y$ . ◀

## 2 Extremal bounds for convex polytopes in Cartesian products

### 2.1 Lower bounds in the plane

In this section, we show that for every  $n \geq 3$ , every  $n \times n$  grid supports a convex polygon with  $\Omega(\log n)$  vertices. The results on the Erdős–Szekeres number cannot be used directly, since they crucially use the assumption that the given set of points is in general position. An  $n \times n$  section of the integer lattice is known to contain  $\Theta(n)$  points in general position [12], and this number is conjectured to be  $\frac{\pi}{\sqrt{3}}n(1 + o(1))$  [17, 31]. However, this result does not apply to arbitrary Cartesian products. It is worth noting that higher dimensional variants for the integer lattice are poorly understood: it is known that an  $n \times n \times n$  section of  $\mathbb{Z}^3$  contains  $\Theta(n^2)$  points no three of which are collinear [22], but no similar statements are known in higher dimensions. We use a recent result from incidence geometry.

► **Lemma 2.** (Payne and Wood [21]) *Every set of  $N$  points in the plane with at most  $\ell$  collinear,  $\ell \leq O(\sqrt{N})$ , contains a set of  $\Omega(\sqrt{N/\log \ell})$  points in general position.*

► **Lemma 3.** *Every  $n \times n$  grid supports a convex polygon of size  $(1 - o(1)) \log n$ .*

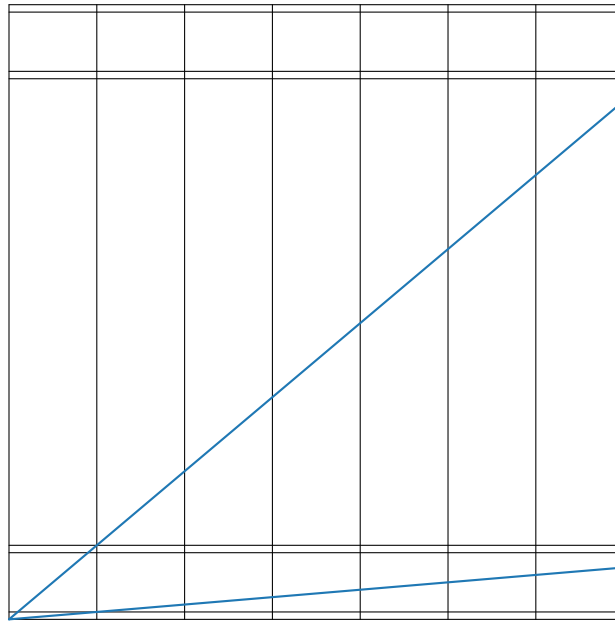
**Proof.** Every  $n \times n$  grid contains a set of  $\Omega(\sqrt{n^2/\log n}) = \Omega(n/\sqrt{\log n})$  points in general position by applying Lemma 2 with  $N = n^2$  and  $\ell = n$ . Discarding points with the same  $x$ - or  $y$ -coordinate reduces the size by a factor at most  $\frac{1}{4}$ , so this asymptotic bound also holds when coordinates in  $X$  and  $Y$  are used at most once. By Suk’s result [29], the grid supports a convex polygon with at least  $(1 - o(1))(\log(n/\sqrt{\log n})) = (1 - o(1)) \log n$  vertices. ◀

### 2.2 Upper bounds in the plane

For the upper bound, we construct  $n \times n$  Cartesian products that do not support large convex chains. For  $n = 8$ , such a grid is depicted in Figure 2.

► **Lemma 4.** *For every  $n \in \mathbb{N}$ , there exists an  $n \times n$  grid that contains at most  $4(\lceil \log n \rceil + 1)$  points in convex position.*

**Proof.** Let  $g(n)$  be the maximum integer such that for all  $n$ -element sets  $X, Y \subset \mathbb{R}$ , the grid  $X \times Y$  supports a convex polygon of size  $g(n)$ ; clearly  $g(n)$  is nondecreasing. Let  $k$  be the minimum integer such that  $n \leq 2^k$ ; thus  $\lceil \log n \rceil \leq k$  and  $g(n) \leq g(2^k)$ . We show that  $g(2^k) \leq 4(k + 1)$  and thereby establish that  $g(n) \leq 4(k + 1)$ .



■ **Figure 2** An  $8 \times 8$  grid with no convex chains of size greater than  $4 = \log 8 + 1$ , where  $X = \{0, 1, \dots, 7\}$  and  $Y = \{0, 1, 16, 17, 256, 257, 272, 273\}$ . Two lines determined by grid points are drawn in blue.

Assume w.l.o.g. that  $n = 2^k$ , and let  $X = \{0, \dots, n - 1\}$ . For a  $k$ -bit integer  $m$ , let  $m_i$  be the bit at its  $i$ -th position, such that  $m = \sum_{i=0}^{k-1} m_i 2^i$ . Let  $Y = \{\sum_{i=0}^{k-1} m_i (2n)^i \mid 0 \leq m \leq n - 1\}$  (see Fig. 2). Both  $X$  and  $Y$  are symmetric:  $X = \{\max(X) - x \mid x \in X\}$  and  $Y = \{\max(Y) - y \mid y \in Y\}$ . Thus, it suffices to show that no convex chain  $P \in \mathcal{C}$  of size greater than  $k + 1$  exists.

Consider two points,  $p = (x, y)$  and  $p' = (x', y')$ , in  $X \times Y$  such that  $x < x'$  and  $y < y'$ . Assume  $y = \sum_{i=0}^{k-1} m_i (2n)^i$  and  $y' = \sum_{i=0}^{k-1} m'_i (2n)^i$ . The slope of the line spanned by  $p$  and  $p'$  is  $\text{slope}(p, p') = \frac{\sum_{i=0}^{k-1} (m'_i - m_i) (2n)^i}{x' - x}$ . Let  $j$  be the largest index such that  $m_j \neq m'_j$ . Then  $y < y'$  implies  $m_j < m'_j$ , and we can bound the slope as follows:

$$\text{slope}(p, p') \geq \frac{(2n)^j - \sum_{i=0}^{j-1} (2n)^i}{x' - x} > \frac{(2n)^j - 2(2n)^{j-1}}{n - 1} = 2 \cdot (2n)^{j-1},$$

$$\text{slope}(p, p') \leq \frac{\sum_{i=0}^j (2n)^i}{x' - x} \leq \frac{\sum_{i=0}^j (2n)^i}{1} = \frac{(2n)^{j+1} - 1}{2n - 1} < 2 \cdot (2n)^j.$$

Hence,  $\text{slope}(p, p') \in I_j = (2 \cdot (2n)^{j-1}, 2 \cdot (2n)^j)$ . Let us define the family of intervals  $I_0, I_1, \dots, I_{k-1}$  analogously, and note that these intervals are pairwise disjoint. Suppose that some convex chain  $P \in \mathcal{C}$  contains more than  $k + 1$  points. Since the slopes of the first  $k + 1$  edges of  $P$  decrease monotonically, by the pigeonhole principle, there must be three consecutive vertices  $p = (x, y)$ ,  $p' = (x', y')$ , and  $p'' = (x'', y'')$  of  $P$  such that both  $\text{slope}(p, p')$  and  $\text{slope}(p', p'')$  are in the same interval, say  $I_j$ . Assume that  $y = \sum_{i=0}^{k-1} m_i (2n)^i$ ,  $y' = \sum_{i=0}^{k-1} m'_i (2n)^{i+1}$ , and  $y'' = \sum_{i=0}^{k-1} m''_i (2n)^{i+1}$ . Then  $j$  is the largest index such that  $m_j \neq m'_j$ , and also the largest index such that  $m'_j \neq m''_j$ . Because  $m < m' < m''$ , we have  $m_j < m'_j < m''_j$ , which is impossible since each of  $m_j, m'_j$  and  $m''_j$  is either 0 or 1.



Hence,  $X \times Y$  does not contain any convex chain in  $\nearrow$  of size greater than  $k+1$ . Analogously, every convex chain in  $\searrow$ ,  $\swarrow$ , or  $\nwarrow$  has at most  $k+1$  vertices. Consequently,  $X \times Y$  contains at most  $4(k+1)$  points in convex position. ◀

### 2.3 Upper bounds in higher dimensions

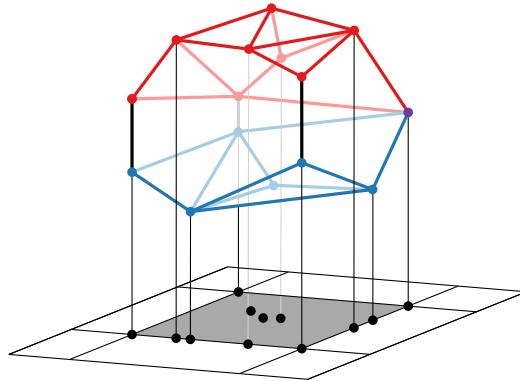
We construct Cartesian products in  $\mathbb{R}^d$ , for  $d \geq 3$ , that match the best known upper bound  $O(\log^{d-1} n)$  for the Erdős–Szekeres numbers in  $d$ -dimensions for points in general position. Our construction generalizes the ideas from the proof of Lemma 4 to  $d$ -space.

► **Theorem 5.** *Let  $d \geq 2$  be fixed. For every  $n \geq 2$ , there exist  $n$ -element sets  $Y_i \subseteq \mathbb{R}$  for  $i = 1, \dots, d$ , such that the Cartesian product  $Y = \prod_{i=1}^d Y_i$  contains at most  $O(\log^{d-1} n)$  points in convex position.*

**Proof.** We construct point sets recursively. For  $d = 2$ , the result follows from Lemma 4. For  $d \geq 3$  and  $0 \leq i \leq j$ , we define  $S_d(i, j)$  as a Cartesian product of  $d$  sets, where the first  $d-1$  sets have  $2^j$  elements and the last set has  $2^i$  elements. We then show that  $S_d(i, j)$  does not contain the vertex set of any full-dimensional convex polyhedron with more than  $2^{d^2-d+1} \cdot i \cdot j^{d-2}$  vertices (there is no restriction on lower-dimensional convex polyhedra).

To initialize the recursion, we define boundary values as follows: For every  $j \geq 0$ , let  $S_2(j, j)$  be the  $2^j \times 2^j$  grid defined in the proof of Lemma 4 that does not contain more than  $4(j+1)$  points in convex position. Note that every line that contains 3 or more points from  $S_2(j, j)$  is axis-parallel (this property was not needed in the proof of Lemma 4). Assume now that  $d \geq 3$ , and  $S_{d-1}(j, j)$  has been defined for all  $j \geq 0$ ; and for all  $k = 1, \dots, d$ , every  $k$ -dimensional flat containing  $2^k + 1$  or more points is parallel to at least one coordinate axis. We now construct  $S_d(i, j)$  for all  $0 \leq i \leq j$  as follows.

Let  $S_d(0, j) = S_{d-1}(j, j) \times \{0\}$ . For  $i = 1, \dots, j$ , we define  $S_d(i, j)$  as the disjoint union of two translates of  $S_d(i-1, j)$ . Specifically, let  $S_d(i, j) = A \cup B$ , where  $A = S_d(i-1, j)$  and  $B = A + \lambda_d^i \mathbf{e}_d$ , where  $\lambda_d^i > 0$  is sufficiently large (as specified below) and algebraically independent from the coordinates of  $S_d(i-1, j)$ , and for all  $k = 1, \dots, d$ , every  $k$ -dimensional flat containing  $2^k + 1$  or more points is parallel to at least one coordinate axis. .



■ **Figure 3** A polyhedron  $\text{conv}(P)$  in  $\mathbb{R}^3$ , whose projection  $\text{conv}(P)^{\text{proj}}$  is a rectangle. Seven points in  $P$  are projected onto the four vertices of  $\text{conv}(P)^{\text{proj}}$ . Overall the *silhouette* of  $P$  contains 12 points. Red (blue) vertices are upper (lower); the purple point is both upper and lower.

Let  $P \subset S_d(i, j)$  be a full-dimensional set in convex position. The orthogonal projection of  $\text{conv}(P)$  to the horizontal hyperplane  $x_d = 0$  is a convex polytope in  $\mathbb{R}^{d-1}$  that we denote by  $\text{conv}(P)^{\text{proj}}$ ; refer to Fig. 3. The *silhouette* of  $P$  is the subset of vertices whose orthogonal

projection to  $x_d = 0$  lies on the boundary of  $\text{conv}(P)^{\text{proj}}$ . Since no three points in  $P$  are collinear, at most two points in  $P$  are projected to the same point. A point  $p \in P$  is an upper (resp., lower) vertex if  $P$  lies in the closed halfspace below (resp., above) some tangent hyperplane of  $\text{conv}(P)$  at  $p$  (a point in  $P$  may be both upper and lower vertex).

We prove, by double induction on  $d$  and  $i$ , the following:

▷ **Claim 6.** If  $P \subset S_d(i, j)$  is a full-dimensional set in convex position, then  $P$  contains at most  $2^{d(d-1)} \cdot i \cdot j^{d-2}$  upper (resp., lower) vertices of  $\text{conv}(P)$ .

For  $d = 2$  and  $i = j$ , this holds by definition (cf. Lemma 4). For  $i = 0$ , the set  $S_d(0, j) = S_{d-1}(j, j) \times \{0\}$  lies in a horizontal hyperplane in  $\mathbb{R}^d$ , and so it is not full-dimensional, hence the claim vacuously holds. By induction,  $S_{d-1}(j, j)$  contains at most  $2^{(d-1)(d-2)} \cdot j \cdot j^{d-3}$  upper (resp., lower) vertices in  $\mathbb{R}^{d-1}$ , hence  $S_d(0, j)$  has at most  $2 \cdot 2^{(d-1)(d-2)} \cdot j \cdot j^{d-3}$  extreme points in  $\mathbb{R}^d$ . Assume that  $d \geq 3$ ,  $1 = i \leq j$ , and the claim holds for  $S_{d-1}(j, j)$ . We prove the claim for  $S_d(i, j)$ . The set  $S_d(1, j)$  is the disjoint union of  $A = S_d(0, j)$  and  $B = S_d(0, j) + \lambda_d^1 \mathbf{e}_d$ . Every upper (resp., lower) vertex of  $S_d(1, j)$  is an extreme vertex in  $A$  or  $B$ , hence  $S_d(1, j)$  contains at most  $4 \cdot 2^{(d-1)(d-2)} \cdot j^{d-2} = 2^{d^2-3d+4} \cdot j^{d-2} < 2^{d^2-d} \cdot 1 \cdot j^{d-2}$  upper (resp., lower) vertices, as required, where we used that  $d \geq 3$ .

In the general case, we assume that  $d \geq 3$ ,  $2 \leq i \leq j$ , and the claim holds for  $S_{d-1}(j, j)$  and  $S_d(i-1, j)$ . We prove the claim for  $S_d(i, j)$ . Recall that  $S_d(i, j)$  is the disjoint union of two translates of  $S_d(i-1, j)$ , namely  $A = S_d(i-1, j)$  and  $B = S_d(i-1, j) + \lambda_d^i \mathbf{e}_d$ . Let  $P \subset S_d(i, j)$  be a full-dimensional set in convex position. We partition the upper vertices in  $P$  as follows. Let  $P_0 \subset P$  be the set of upper vertices whose orthogonal projections to  $x_d = 0$  are vertices of  $\text{conv}(P)^{\text{proj}}$ . For  $k = 1, \dots, d-1$ , let  $P_k \subset P$  be the set of upper vertices whose orthogonal projections to  $x_d = 0$  lie in the relative interior of a  $k$ -face of  $\text{conv}(P)^{\text{proj}}$ . By construction, only axis-aligned faces can contain interior points. For the  $(d-1)$ -dimensional polytope  $\text{conv}(P)^{\text{proj}}$ , the axis-aligned  $k$ -faces ( $k = 1, \dots, d-1$ ) can be partitioned into  $\binom{d-1}{k}$  equivalence classes, based on the set of parallel coordinate axes.

The orthogonal projection of  $S_d(i, j)$  to  $x_d = 0$  is  $S_{d-1}(j, j)$ , and the orthogonal projection of  $P_0$ , denoted  $P_0^{\text{proj}}$ , is the vertex set of a  $(d-1)$ -dimensional convex polyhedron in  $S_{d-1}(j, j)$ . By induction,  $|P_0| \leq 2 \cdot 2^{(d-1)(d-2)} \cdot j \cdot j^{d-3} = 2^{d^2-3d+3} j^{d-2}$ . We show the following.

▷ **Claim 7.** For every axis-aligned face  $F$  of  $\text{conv}(P)^{\text{proj}}$ , the set of upper vertices that project to the interior of  $F$  is contained in either  $A$  or  $B$ .

Let  $F$  be an axis-aligned  $k$ -face of  $\text{conv}(P)^{\text{proj}}$  for  $k \in \{1, 2, \dots, d-1\}$ . Let  $P(F) \subset P$  be the set of upper vertices whose orthogonal projections lie in the interior of  $F$ , and let  $P(\partial F)$  be the set of upper vertices whose orthogonal projections lie in the boundary of  $F$ . Let  $P(\partial F)^{\text{proj}}$  be the orthogonal projection of  $P(\partial F)$  to the hyperplane  $x_d = 0$ . Consider the point set  $P' = P(\partial F)^{\text{proj}} \cup P(F)$ , and observe that if  $P(F) \neq \emptyset$ , then it is a vertex set of a  $(k+1)$ -dimensional polytope in which all vertices are upper. It remains to show that  $P(F) \subseteq A$  or  $P(F) \subseteq B$ . Suppose, for the sake of contradiction, that  $P(F)$  contains points from both  $A$  and  $B$ . Let  $p_a$  be a vertex in  $P(F)$  with the maximum  $x_d$ -coordinate. The 1-skeleton of  $\text{conv}(P')$  contains a  $x_d$ -monotonically decreasing path from  $p_a$  to an  $x_d$ -minimal vertex in  $P'$ . Let  $p_b$  be the neighbor of  $p_a$  along such a path. Then  $p_b \in B$  by the choice of  $p_a$ . Every hyperplane containing  $p_a$  and  $p_b$ , in particular the tangent hyperplane of  $P'$  containing the edge  $p_a p_b$ , partitions  $P(\partial F)^{\text{proj}}$ , which is a contradiction.

We can now finish the proof of Claim 6. By induction on  $S_d(i-1, j)$ , we have

$$\begin{aligned} |P_k| &\leq \binom{d-1}{k} \cdot 2^{(d-k-1)(d-k-2)+1} j^{d-k-1} \cdot 2^{k(k-1)} \cdot (i-1) \cdot j^{k-1} \\ &\leq \binom{d-1}{k} \cdot 2^{(d-1)(d-2)} \cdot (i-1) \cdot j^{d-2}. \end{aligned}$$

Altogether, the number of upper vertices is

$$\sum_{k=0}^{d-1} |P_k| \leq 2^{d^2-3d+3} j^{d-2} + \sum_{k=1}^{d-1} \binom{d-1}{k} 2^{(d-1)(d-2)} \cdot (i-1) \cdot j^{d-2} < 2^{d(d-1)} \cdot i \cdot j^{d-2},$$

as required, where we used the binomial theorem and the inequality  $d \geq 3$ .  $\blacktriangleleft$

## 2.4 Lower bounds in higher dimensions

The proof technique in Section 2.1 is insufficient for establishing a lower bound of  $\Omega(\log^{d-1} n)$  for  $d \geq 3$ . Whereas a  $d$ -dimensional  $n \times \dots \times n$  grid contains  $\Omega(n^d)$  points in general position for some  $\delta = \delta(d) > 0$  [6], the current best lower bound on the number of points in convex position in any set of  $n$  points in general position in  $\mathbb{R}^d$  is  $\Omega(\log n)$ ; the conjectured value is  $\Omega(\log^{d-1} n)$ . Instead, we rely on the structure of Cartesian products and induction on  $d$ . Our main result in this section is the following.

► **Theorem 8.** *Every  $d$ -dimensional Cartesian product  $\prod_{i=1}^d Y_i$ , where  $|Y_i| = n$  and  $d$  is fixed, contains  $\Omega(\log^{d-1} n)$  points in convex position.*

We say that a strictly increasing sequence of real numbers  $A = (a_1, \dots, a_n)$ , has the *monotone differences* property (for short,  $A$  is *MD*), if either  $a_{i+1} - a_i > a_i - a_{i-1}$ , or  $a_{i+1} - a_i < a_i - a_{i-1}$  for  $i = 2, \dots, n-1$ . Furthermore, the sequence  $A$  is *r-MD* for some  $r > 1$  if either  $a_{i+1} - a_i \geq r(a_i - a_{i-1})$ , or  $a_{i+1} - a_i \leq (a_i - a_{i-1})/r$  for  $i = 2, \dots, n-1$ .

A finite set  $X \subseteq \mathbb{R}$  is *MD* (resp., *r-MD*) if its elements arranged in increasing order form an MD (resp., *r-MD*) sequence. These sequences are intimately related to convexity: a strictly increasing sequence  $A = (a_1, \dots, a_n)$  is MD if and only if there exists a monotone (increasing or decreasing) convex function  $f: \mathbb{R} \rightarrow \mathbb{R}$  such that  $a_i = f(i)$  for all  $i = 1, \dots, n$ . MD sets have been studied in additive combinatorics [11, 18, 23, 27].

We first show that every  $n$ -element set  $X \subseteq \mathbb{R}$  contains an MD subset of size  $\Omega(\log n)$ , and this bound is the best possible (Lemma 9). In contrast, every  $n$ -term arithmetic progression contains an MD subsequence of  $\Theta(\sqrt{n})$  terms: for example  $(0, \dots, n-1)$  contains the subsequence  $(i^2 : i = 0, \dots, \lfloor \sqrt{n-1} \rfloor)$ . We then show that for constant  $d \geq 2$ , the  $d$ -dimensional Cartesian product of MD sets of size  $n$  contains  $\Theta(n^{d-1})$  points in convex position. The combination of these results immediately implies that every  $n \times \dots \times n$  Cartesian product in  $\mathbb{R}^d$  contains  $\Omega(\log^{d-1} n)$  points in convex position.

The following lemma gives a lower bound for MD sequences. It is known that a monotone sequence of  $n$  reals contains a 2-MD sequence (satisfying the so-called *doubling differences condition* [26]) of size  $\Omega(\log n)$  [5, Lemma 4.1]; see also [8] for related recent results.

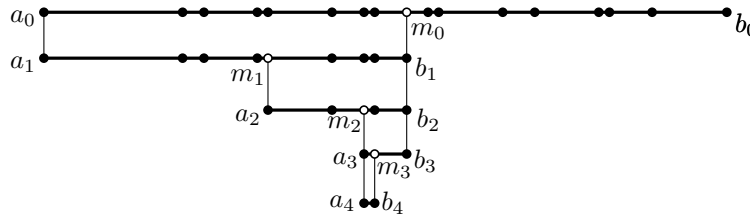
► **Lemma 9.** *Every set of  $n$  real numbers contains an MD subset of size  $\lfloor (\log n)/2 \rfloor + 1$ . For every  $n \in \mathbb{N}$ , there exists a set of  $n$  real numbers in which the size of every MD subset is at most  $\lceil \log n \rceil + 1$ .*

**Proof.** Let  $X = (x_0, \dots, x_{n-1})$  be a strictly increasing sequence. Assume w.l.o.g. that  $n = 2^\ell + 1$  for some  $\ell \in \mathbb{N}$ . We construct a sequence of nested intervals

$$[a_0, b_0] \supset [a_1, b_1] \supset \dots \supset [a_\ell, b_\ell]$$

such that the endpoints of the intervals are in  $X$  and the lengths of the intervals decrease by factors of 2 or higher, that is,  $b_i - a_i \leq (b_{i-1} - a_{i-1})/2$  for  $i = 1, \dots, \ell$ .

We start with the interval  $[a_0, b_0] = [x_0, x_{n-1}]$ ; and for every  $i = 0, \dots, \ell - 1$ , we divide  $[a_i, b_i]$  into two intervals at the median, and recurse on the shorter interval.



■ **Figure 4** A sequence  $X$  of 17 elements and nested intervals  $[a_0, b_0] \supset \dots \supset [a_4, b_4]$ .

By partitioning  $[a_i, b_i]$  at the median, the algorithm maintains the invariant that  $[a_i, b_i]$  contains  $2^{\ell-i} + 1$  elements of  $X$ . Note that for every  $i = 1, \dots, \ell$ , we have either  $(a_{i-1} = a_i$  and  $b_{i-1} < b_i)$  or  $(a_{i-1} = a_i$  and  $b_i < b_{i-1})$ . Consequently, the sequences  $A = (a_0, a_1, \dots, a_\ell)$  and  $B = (b_\ell, b_{\ell-1}, \dots, b_0)$  both increase (not necessarily strictly), and at least one of them contains at least  $1 + \ell/2$  distinct terms. Assume w.l.o.g. that  $A$  contains at least  $1 + \ell/2$  distinct terms. Let  $C = (c_0, \dots, c_k)$  be a maximal strictly increasing subsequence of  $A$ . Then  $k \geq \ell/2 = \lfloor (\log n)/2 \rfloor$ .

We show that  $C$  is an MD sequence. Let  $i \in \{1, \dots, k - 1\}$ . Assume that  $c_i = a_j = \dots = a_{j'}$  for consecutive indices  $j, \dots, j'$ . Then  $c_{i-1} = a_{j-1}$ ,  $c_i = a_j$ , and  $c_{i+1} = a_{j'+1}$ . By construction,  $c_i \in [a_{j-1}, b_{j-1}] = [c_{i-1}, b_j]$  such that  $c_i - c_{i-1} \geq b_j - c_i$ . Similarly,  $c_{i+1} \in [a_{j'}, b_{j'}] = [c_i, b_{j'}]$  such that  $c_{i+1} - c_i \geq b_{j'} - c_{i+1}$ . However,  $[a_j, b_j] \subset [a_{j-1}, b_{j-1}]$ . As required, this yields

$$c_{i+1} - c_i = a_{j'+1} - a_j < b_j - a_j \leq \frac{b_{j-1} - a_{j-1}}{2} \leq a_j - a_{j-1} = c_i - c_{i-1}.$$

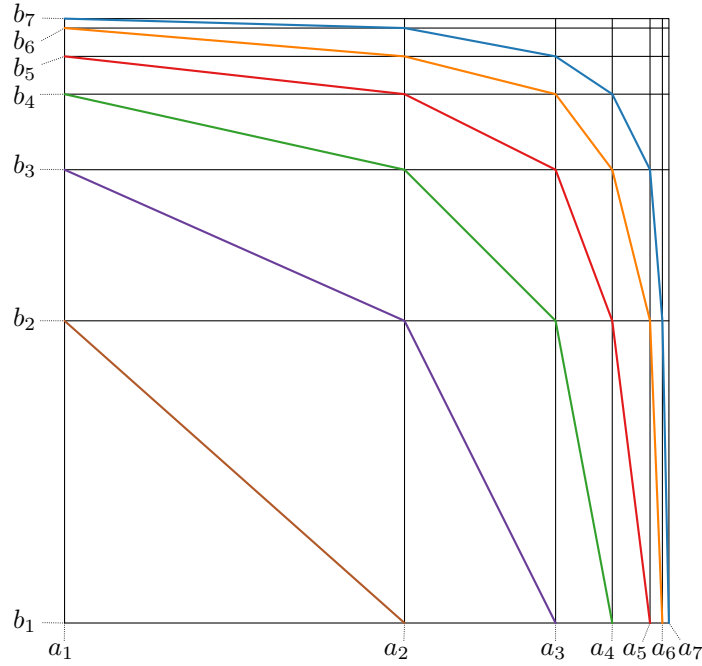
The upper bound construction is the point set  $Y$  defined in the proof of Lemma 4, for which every chain in  $\sphericalangle$  or  $\swarrow$  supported by  $\{0, \dots, n - 1\} \times Y$  has at most  $\lceil \log n \rceil + 1$  vertices. Let  $\{b_0, \dots, b_{\ell-1}\} \subset Y$  be an MD subset such that  $b_0 < \dots < b_{\ell-1}$ . Then  $\{(i, b_i) : i = 0, \dots, \ell - 1\} \subset X \times Y$  is in  $\sphericalangle$  or  $\swarrow$ . Consequently, every MD subset of  $Y$  has at most  $\lceil \log n \rceil + 1$  terms, as claimed. ◀

We show how to use Lemma 9 to establish a lower bound in the plane. While this approach yields worse constant coefficients than Lemma 3, its main advantage is that it generalizes to higher dimensions (see Lemma 12 below).

► **Lemma 10.** *The Cartesian product of two MD sets, each of size  $n$ , supports  $n$  points in convex position.*

**Proof.** Let  $A = \{a_1, \dots, a_n\}$  and  $B = \{b_1, \dots, b_n\}$  be MD sets such that  $a_i < a_{i+1}$  and  $b_i < b_{i+1}$  for  $i = 1, \dots, n - 1$ . We may assume, by applying a reflection if necessary, that  $a_{i+1} - a_i < a_i - a_{i-1}$  and  $b_{i+1} - b_i < b_i - b_{i-1}$ , for  $i = 2, \dots, n - 1$  (see Fig. 5).

We define  $P \subset A \times B$  as the set of  $n$  points  $(a_i, b_j)$  such that  $i + j = n + 1$ . By construction, every horizontal (vertical) line contains at most one point in  $P$ . Since the differences  $a_i - a_{i-1}$  are positive and strictly decrease in  $i$ ; and the differences  $b_{\ell-i} - b_{\ell-i-1}$  are negative and their absolute values strictly increase in  $i$ , the slopes  $(b_{\ell-i} - b_{\ell-i-1}) / (a_i - a_{i-1})$  strictly decrease, which proves the convexity of  $P$ . ◀



■ **Figure 5** A  $7 \times 7$  grid  $\{a_1, \dots, a_7\} \times \{b_1, \dots, b_7\}$ , where the differences between consecutive  $x$ -coordinates (resp.,  $y$ -coordinates) decrease by factors of 2 or higher. The point sets  $\{(0, 0)\} \cup \{(a_i, b_j) : i + j = k\}$ , for  $k = 2, \dots, 8$ , form nested convex chains.

► **Lemma 11.** *The Cartesian product of three MD sets, each of size  $n$ , contains  $\binom{n+1}{2}$  points in convex position.*

**Proof.** Let  $A = \{a_1, \dots, a_n\}$ ,  $B = \{b_1, \dots, b_n\}$ , and  $C = \{c_1, \dots, c_n\}$  be MD sets, where the elements are labeled in increasing order. We may assume, by applying a reflection in the  $x$ -,  $y$ -, or  $z$ -axis if necessary, that  $a_{i+1} - a_i < a_i - a_{i-1}$ ,  $b_{i+1} - b_i < b_i - b_{i-1}$ , and  $c_{i+1} - c_i < c_i - c_{i-1}$ , for  $i = 2, \dots, n-1$ . For  $i, j, k \in \{1, \dots, n\}$ , let  $p_{i,j,k} = (a_i, b_j, c_k) \in A \times B \times C$ . We now let  $P = \{p_{i,j,k} : i+j+k = n+2\}$ . It is clear that  $|P| = \sum_{i=1}^n i = \binom{n+1}{2}$ . We let  $P' = P \cup \{p_{1,1,1}\}$  and show that the points in  $P'$  are in convex position.

By Lemma 10, the points in  $P'$  lying in the planes  $x = a_1$ ,  $y = b_1$ , and  $z = c_1$  are each in convex position. These convex  $(n+1)$ -gons are faces of the convex hull of  $P$ , denoted  $\text{conv}(P)$ . We show that the remaining faces of  $\text{conv}(P)$  are the triangles  $T'_{i,j,k}$  spanned by  $p_{i,j,k}$ ,  $p_{i,j+1,k-1}$ , and  $p_{i+1,j,k-1}$ ; and the triangles  $T''_{i,j,k}$  spanned by  $p_{i,j,k}$ ,  $p_{i,j-1,k+1}$ , and  $p_{i-1,j,k+1}$ .

The projection of these triangles to the  $xy$ -plane is shown in Fig. 5. By construction, the union of these faces is homeomorphic to a sphere. It suffices to show that the dihedral angle between any two adjacent triangles is convex. Without loss of generality, consider the triangle  $T'_{i,j,k}$ , which is adjacent to at most three other triangles:  $T''_{i+1,j,k-1}$ ,  $T''_{i,j+1,k-1}$ , and  $T''_{i+1,j+1,k-2}$  (if they exist). Consider first the triangles  $T'_{i,j,k}$  and  $T''_{i+1,j+1,k-2}$ . They share the edge  $p_{i+1,j-1,k+1}p_{i-1,j+1,k+1}$ , which lies in the plane  $z = c_{k+1}$ . The orthogonal projections of these triangles to this plane are congruent, however their extents in the  $z$ -axis are  $c_{i+1} - c_i$  and  $c_i - c_{i-1}$ , respectively. Since  $c_{i+1} - c_i < c_i - c_{i-1}$ , their dihedral angle is convex. Similarly, the dihedral angles between  $T'_{i,j,k}$  and  $T''_{i+1,j,k-1}$  (resp.,  $T''_{i,j+1,k-1}$ ) is convex because  $a_{i+1} - a_i < a_i - a_{i-1}$  and  $b_{i+1} - b_i < b_i - b_{i-1}$ . ◀

The proof technique of Lemma 11 generalizes to higher dimensions (details are provided in the full version):

► **Lemma 12.** *For every constant  $d \geq 2$ , the Cartesian product of  $d$  MD sets, each of size  $n$ , contains  $\Omega(n^{d-1})$  points in convex position.*

Now Theorem 8 follows from Lemma 9 and Lemma 12.

### 3 Algorithms

In this section, we describe polynomial-time algorithms for (i) finding convex chains and caps of maximum size; and (ii) approximating the maximum size of a convex polygon; where these structures are *supported* by a given grid. The main challenge is to ensure that the vertices of the convex polygon (resp., cap or chain) have distinct  $x$ - and  $y$ -coordinates. The coordinates of a point  $p \in X \times Y$  are denoted by  $x(p)$  and  $y(p)$ .

As noted in Section 1, efficient algorithms are available for finding a largest convex polygon or convex cap *contained* in a planar point set. Edelsbrunner and Guibas [10, Thm. 5.1.2] use the dual line arrangement of  $N$  points in the plane and dynamic programming to find the maximum size of a convex cap in  $\curvearrowright$  in  $O(N^2)$  time and  $O(N)$  space; the same bounds hold for  $\smile$ ,  $\frown$ , and  $\supset$ . A longest convex cap can be also returned in  $O(N^2 \log N)$  time and  $O(N \log N)$  space. It is straightforward to adapt their algorithm to find the maximum size of a convex cap in  $\curvearrowright$ , and report a longest such chain within the same time and space bounds. Since  $x$ - and  $y$ -coordinates do not repeat in a convex chain, we obtain the following.

► **Theorem 13.** *In a given  $n \times n$  grid, the maximum size of a supported convex chain can be computed in  $O(n^4)$  time and  $O(n^2)$  space; and a supported convex chain of maximum size can be computed in  $O(n^4 \log n)$  time and  $O(n^2 \log n)$  space.*

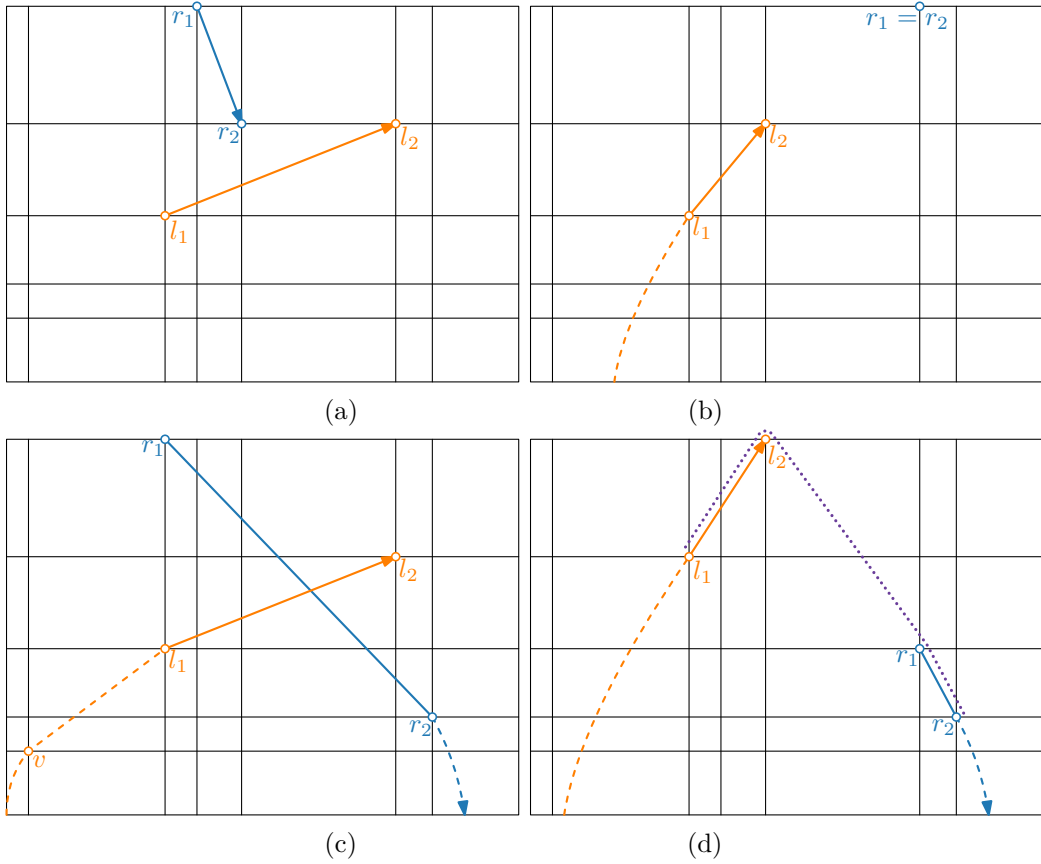
We make use of the following general observation:

► **Observation 14.** *If a supported convex polygon  $P$  is in a set  $\curvearrowright$ ,  $\supset$ ,  $\smile$ ,  $\frown$ ,  $\curvearrowleft$ ,  $\supset$ ,  $\smile$ , or  $\frown$ , then every subsequence of  $P$  is in the same set. That is, these classes are hereditary.*

#### 3.1 Convex caps

For computing the maximum size of a convex cap in  $\curvearrowright$ , we need to be careful to use each  $y$ -coordinate at most once. We design an algorithm that finds the maximum size of *two* convex chains that use distinct  $y$ -coordinates by dynamic programming. Specifically, for two edges  $l = (l_1, l_2)$  and  $r = (r_1, r_2)$ , we compute the maximum total size  $C(l, r)$  of a pair of chains  $A \in \curvearrowright$  and  $B \in \supset$  such that their vertices use distinct  $y$ -coordinates and such that the last two vertices of  $A$  are  $l_1$  and  $l_2$  (or  $A = (l_1)$  if  $l_1 = l_2$ ), and the first two vertices of  $B$  are  $r_1$  and  $r_2$  (or  $B = (r_1)$  if  $r_1 = r_2$ ). We use the dynamic programming algorithm of [10] to find  $L(p_1, p_2)$  (resp.,  $R(p_1, p_2)$ ), the size of a largest convex chain  $P$  in  $\curvearrowright$  (resp.,  $\supset$ ), ending (resp., starting) with vertices  $p_1$  and  $p_2$ , or  $P = (p_1)$  if  $p_1 = p_2$ .

The desired quantity  $C(l, r)$  can be computed by dynamic programming. By Observation 14, we can always safely eliminate the highest vertex of the union of the two chains, to find a smaller subproblem, as this vertex cannot be (implicitly) part of the optimal solution to a subproblem. In particular, if  $l$  is a single vertex and it is highest, we can simply use the value of  $R(r_1, r_2)$ , incrementing it by one for the one vertex of  $l$ . Analogously, we handle the case if  $r$  is or both  $l$  and  $r$  are a single vertex. The interesting case is when both chains end in an edge. Here, we observe that we can easily check whether  $l$  and  $r$  use unique coordinates.



■ **Figure 6** Illustration for the cases of  $C(l, r)$ . (a) Invalid configuration, as  $y(r_2) = y(l_2)$ . (b)  $r$  is a single point above  $l_2$ , so we look for the longest chain ending in  $l$ . (c) Removing the topmost point (from  $l$  in this case), testing all valid possible  $v$  to find the longest chain. Note that the left and right chain may not complete to a cap – this is checked separately. (d) We only need to test whether  $l$  and  $r$  make a cap (purple dotted line) to check whether the  $C(l, r)$  entry should be considered.

If not, then this subproblem is invalid; otherwise, we may find a smaller subproblem by eliminating the highest vertex and comparing all possible subchains that could lead to it.

With the reasoning above, we obtain the recurrence below; see Fig. 6(a–c) for illustration. The first case eliminates invalid edges, and edge pairs that use a  $y$ -coordinate more than once. In all remaining cases, we assume that  $l \in \nearrow$ ,  $r \in \searrow$ , and  $l$  and  $r$  use distinct  $y$ -coordinates.

$$C(l, r) = \begin{cases} -\infty & \text{if } l_1 \neq l_2 \text{ and } l \notin \nearrow, \text{ or} \\ & r_1 \neq r_2 \text{ and } r \notin \searrow, \text{ or} \\ & \{y(l_1), y(l_2)\} \cap \{y(r_1), y(r_2)\} \neq \emptyset \\ 2 & \text{otherwise, if } l_1 = l_2 \text{ and } r_1 = r_2 \\ L(l_1, l_2) + 1 & \text{otherwise, if } r_1 = r_2 \text{ and } y(l_2) < y(r_1) \\ R(r_1, r_2) + 1 & \text{otherwise, if } l_1 = l_2 \text{ and } y(l_2) > y(r_1) \\ \max_{(v, l_1, l_2) \in \nearrow \text{ or } v = l_1} C((v, l_1), r) + 1 & \text{otherwise, if } y(l_2) > y(r_1) \\ \max_{(r_1, r_2, v) \in \searrow \text{ or } v = r_1} C(l, (r_2, v)) + 1 & \text{otherwise, } y(l_2) < y(r_1). \end{cases}$$

Let  $E = (X \times Y)^2$  denote the number of pairs (edges) in the grid, from which we take  $l$  and  $r$ . As  $|E| = O(n^4)$ , we can compute  $C(l, r)$  for all  $l$  and  $r$  in  $O(|E|^2 |X \times Y|) = O(n^{10})$  time and  $O(|E|^2) = O(n^8)$  space. With  $C(l, r)$ , we can easily find the size of a maximum size cap  $P$  in  $\nearrow$ , using the observation below, and analogous observations for the special case  $k = 1$  and/or  $\ell = 1$  (see Fig. 6(d)).

► **Observation 15.** If  $A = (a_1, \dots, a_k) \in \curvearrowright$  and  $B = (b_1, \dots, b_\ell) \in \curvearrowleft$  with  $k \geq 2, \ell \geq 2$  and  $(a_{k-1}, a_k, b_1, b_2) \in \curvearrowright$  and  $A$  and  $B$  use distinct  $y$ -coordinates, then  $(a_1, \dots, a_k, b_1, \dots, b_\ell)$  lies in  $\curvearrowright$  and has size  $k + \ell$ .

Note that the condition  $(a_{k-1}, a_k, b_1, b_2) \in \curvearrowright$  implies that the  $x$ -coordinates are disjoint.

► **Theorem 16.** For a given  $n \times n$  grid, a supported convex cap of maximum size can be computed in  $O(n^{10})$  time and  $O(n^8)$  space.

Although computing the maximum size of a supported convex polygon remains elusive, we can easily devise a constant-factor approximation algorithm by eliminating duplicate coordinates as described in Section 3.3 of the full version.

#### 4 The maximum number of convex polygons

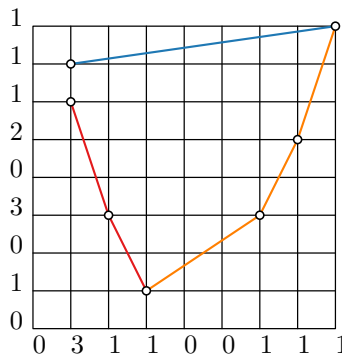
Let  $F(n)$  be the maximum number of convex polygons that can be present in an  $n \times n$  grid, with no restriction on the number of times each coordinate is used. Let  $G(n)$  be this number where all  $2n$  grid lines are used (i.e., each grid line contains at least one vertex of the polygon). Let  $\bar{F}(n)$  and  $\bar{G}(n)$  be the corresponding numbers where each grid line is used at most once (so  $\bar{F}(n)$  counts the maximum number of supported convex polygons). By definition, we have  $F(n) \geq G(n) \geq \bar{G}(n)$  and  $F(n) \geq \bar{F}(n) \geq \bar{G}(n)$  for all  $n \geq 2$ . We prove the following theorem, in which the  $\Theta^*(.)$  notation hides polynomial factors in  $n$ .

► **Theorem 17.** The following bounds hold:

$$F(n) = \Theta^*(16^n), \quad \bar{F}(n) = \Theta^*(9^n), \quad G(n) = \Theta^*(9^n), \quad \bar{G}(n) = \Theta^*(4^n).$$

##### 4.1 Upper bounds

We first prove that  $F(n) = O(n \cdot 16^n)$  by encoding each convex polygon in a unique way, so that the total number of convex polygons is bounded by the total number of encodings. Recall that a convex polygon  $P$  can be decomposed into four convex chains  $\curvearrowright_P, \curvearrowleft_P, \swarrow_P, \searrow_P$ , with only extreme vertices of  $P$  appearing in multiple chains. Let  $\zeta_P = \curvearrowright_P \cup \curvearrowleft_P$  and  $\smile_P = \swarrow_P \cup \searrow_P$ . To encode  $P$ , we assign the following number to each of the  $2n$  grid lines  $\ell$  (see Fig. 7): 0 if  $\ell$  is not incident on any vertex of  $P$ , 3 if  $\ell$  is incident on multiple vertices of  $P$ , 1 if  $\ell$  is incident on one vertex of  $P$  and that vertex lies on  $\zeta_P$  if  $\ell$  is horizontal, or on  $\smile_P$  if  $\ell$  is vertical, and 2 otherwise. We also record the index of the horizontal line containing the leftmost vertex of  $P$  (pick the topmost of these if there are multiple leftmost points).



■ **Figure 7** Encoding the grid lines.



## 22:14 Convex Polygons in Cartesian Products

Since each of the  $2n$  grid lines is assigned one of 4 possible values, and there are  $n$  horizontal lines, the total number of encodings is  $O(n \cdot 4^{2n}) = O(n \cdot 16^n)$ . All that is left to show is that each encoding corresponds to at most one convex polygon.

First, observe that if  $P$  is a convex chain, say in  $\nearrow$ , then the set of grid lines containing a vertex of  $P$  uniquely defines  $P$ : since both coordinates change monotonically, the  $i$ -th vertex of  $P$  must be the intersection of the  $i$ -th horizontal and vertical lines. So all we need to do to reconstruct  $P$  is to identify the set of lines that make up each convex chain.

Since we know the location of the (topmost) leftmost vertex of  $P$ , we know where  $\nearrow_P$  starts. Every horizontal line above this point labelled with a 1 or 3 must contain a vertex of  $\nearrow_P$ ; let  $k$  be the number of such lines. Since the  $x$ -coordinates are monotonic as well,  $\nearrow_P$  ends at the  $k$ -th vertical line labelled with a 2 or 3. The next chain,  $\searrow_P$ , starts either at the end of  $\nearrow_P$ , if the horizontal line is labelled with a 1, or at the intersection of this horizontal line with the next vertical line labelled with a 2 or 3, if this horizontal line is labelled with a 3. We can find the rest of the chains in a similar way. Thus,  $F(n) = O(n \cdot 16^n)$ .

The upper bounds for  $\bar{F}(n)$ ,  $G(n)$ , and  $\bar{G}(n)$  are analogous, except that certain labels are excluded. For the number of supported convex polygons  $\bar{F}(n)$ , each grid line is used at most once, which means that the label 3 cannot be used. Thus,  $\bar{F}(n) = O(n \cdot 3^{2n}) = O(n \cdot 9^n)$ . Similarly, for  $G(n)$ , all grid lines contain at least one vertex of the polygon, so the label 0 cannot be used. Therefore  $G(n) = O(n \cdot 3^{2n}) = O(n \cdot 9^n)$ . Finally, for  $\bar{G}(n)$ , every grid line contains exactly one vertex of the polygon, so neither 0 nor 3 can be used as labels. This gives  $\bar{G}(n) = O(n \cdot 2^{2n}) = O(n \cdot 4^n)$  possibilities.

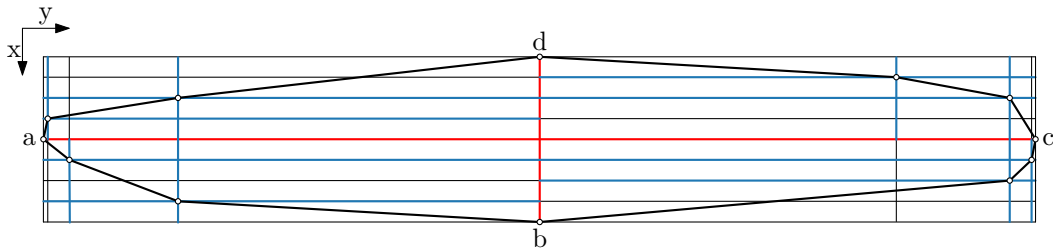
### 4.2 Lower bounds

Assume that  $n = 2m + 3$ , where  $m \in \mathbb{N}$  satisfies suitable divisibility conditions, as needed. All four lower bounds use the same grid, constructed as follows (see Fig. 8).

$$\begin{aligned} X &= \{1, \dots, n-1\} & Y^- &= \{y_1, \dots, y_{m+2}\}, \text{ where } y_i = n^i \\ Y &= Y^- \cup Y^+ & Y^+ &= \{z_1, \dots, z_{m+2}\}, \text{ where } z_i = 2 \cdot y_{m+2} - y_i \end{aligned}$$

Note that this results in an  $(n-1) \times (n-1)$  grid, since  $y_{m+2} = z_{m+2}$ . To obtain an  $n \times n$  grid, we duplicate the median grid lines in both directions and offset them by a sufficiently small distance  $\varepsilon > 0$ . The resulting grid has the property that any three points  $p, q, r$  in the lower half  $X \times Y^-$  with  $x(p) < x(q) < x(r)$  and  $y(p) < y(q) < y(r)$  make a left turn at  $q$ . To see this, suppose that  $y(p) = n^i$ ,  $y(q) = n^j$ , and  $y(r) = n^k$ , for some  $1 \leq i < j < k \leq n$ . Then the slope of  $pq$  is strictly smaller than the slope of  $qr$ , since

$$\text{slope}(qr) = \frac{n^k - n^j}{x(r) - x(q)} \geq \frac{n^{j+1} - n^j}{n-1} = n^j > \frac{n^j - n^i}{1} \geq \frac{n^j - n^i}{x(q) - x(p)} = \text{slope}(pq).$$



■ **Figure 8** The  $n \times n$  grid defined in Section 4.2, with  $n = 9 = 2m + 3$  for  $m = 3$ , before doubling the median lines. The segments (parts of grid lines) incident to vertices are drawn in blue.

Thus, any sequence of points with increasing  $x$ - and  $y$ -coordinates in the lower half is in  $\swarrow$ . By symmetry, such a sequence in the upper half  $X \times Y^+$  is in  $\nearrow$ . Analogously, points with increasing  $x$ -coordinates and decreasing  $y$ -coordinates are in  $\searrow$  if they are in the lower half and  $\nwarrow$  if they are in the upper half.

We first derive lower bounds on  $\bar{G}(n)$  and  $G(n)$  by constructing a large set of convex polygons that use each grid line at least once. Then we use these bounds to derive the bounds on  $\bar{F}(n)$  and  $F(n)$ . The polygons we construct all share the same four extreme vertices, which lie on the intersections of the grid boundary with the duplicated median grid lines. Specifically, the leftmost and rightmost vertices are the intersections of the duplicate horizontal medians with the left and right boundary, and the highest and lowest vertices are the intersections of the duplicate vertical medians with the top and bottom boundary. Since each of these median lines now contains a vertex, we can choose additional vertices from the remaining  $2m$  grid lines in each direction.

To construct each polygon, select  $m/2$  vertical grid lines left of the median to participate in the bottom chain, and do the same right of the median. Likewise, select  $m/2$  horizontal grid lines above and below the median, respectively, to participate in the left chain. The remaining grid lines participate in the other chain (top or right). This results in a polygon with  $m/2$  vertices in each quadrant of the grid (excluding the extreme vertices). The convexity follows from our earlier observations. The total number of such polygons is

$$\binom{m}{\frac{m}{2}}^4 = \Theta\left(\left(m^{-\frac{1}{2}}2^m\right)^4\right) = \Theta(m^{-2}2^{4m}) = \Theta(n^{-2}2^{2n}) = \Theta(n^{-2}4^n) = \Theta^*(4^n).$$

The first step uses the following estimate, which can be derived from Stirling’s formula for the factorial [9]. Let  $0 < \alpha < 1$ , then

$$\binom{n}{\alpha n} = \Theta(n^{-\frac{1}{2}}2^{H(\alpha)n}), \text{ where } H(\alpha) = -\alpha \log \alpha - (1 - \alpha) \log(1 - \alpha).$$

For the lower bound on  $G(n)$ , the only difference is that we now allow grid lines to contain vertices in two chains. We obtain a maximum when we divide the grid lines evenly between the three groups (bottom chain, top chain, both chains). Thus, we select  $m/3$  vertical grid lines left of the median to participate in the bottom chain, another  $m/3$  to participate in the top chain and the remaining  $m/3$  participate in both. We repeat this selection to the right of the median and on both sides of the median horizontal line. As before, this results in a convex polygon with the same number of vertices in each quadrant of the grid – exactly  $2m/3$  this time. The number of such polygons is

$$\begin{aligned} \binom{m}{\frac{m}{3}}^4 \binom{\frac{2m}{3}}{\frac{m}{3}}^4 &= \Theta\left(\left(m^{-\frac{1}{2}}2^{H(\frac{1}{3})m} \cdot m^{-\frac{1}{2}}2^{H(\frac{1}{2})\frac{2m}{3}}\right)^4\right) \\ &= \Theta\left(m^{-4}2^{4m(\log 3 - \frac{2}{3} + \frac{2}{3})}\right) = \Theta(n^{-4}2^{2n \log 3}) = \Theta(n^{-4}9^n) = \Theta^*(9^n). \end{aligned}$$

We translate these bounds to bounds on  $\bar{F}(n)$  and  $F(n)$  in the full version.

### 4.3 The maximum number of weakly convex polygons

A polygon  $P$  in  $\mathbb{R}^2$  is *weakly convex* if all of its internal angles are less than or equal to  $\pi$ . We summarize the bounds we obtain in the following (details are provided in the full version):

► **Theorem 18.** *Let  $W(n)$  denote the maximum number of weakly convex polygons that can be present in an  $n \times n$  grid. Then  $W(n) = \Omega(16^n)$  and  $W(n) = O^*(16^n)$ .*

## References

- 1 Imre Bárány. Random points and lattice points in convex bodies. *Bulletin of the American Mathematical Society*, 45:339–365, 2008.
- 2 Imre Bárány and János Pach. On the number of convex lattice polygons. *Combinatorics, Probability and Computing*, 1(4):295–302, 1992.
- 3 Imre Bárány and Anatoly Moiseevich Vershik. On the number of convex lattice polytopes. *Geometric and Functional Analysis*, 2(4):381–393, 1992.
- 4 Alexander Barvinok. Lattice points and lattice polytopes. In Jacob E. Goodman, Joseph O’Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 7, pages 185–210. CRC Press, Boca Raton, FL, 3rd edition, 2017.
- 5 Boris Bukh and Jiří Matoušek. Erdős–Szekeres-type statements: Ramsey function and decidability in dimension 1. Manuscript, 2012. Available from: <https://arxiv.org/abs/1207.0705>.
- 6 Jean Cardinal, Csaba D. Tóth, and David R. Wood. A note on independent hyperplanes and general position subsets in  $d$ -space. *Journal of Geometry*, 108(1):33–43, 2017.
- 7 Václav Chvátal and G. T. Klincksiek. Finding largest convex subsets. *Congressus Numerantium*, 29:453–460, 1980.
- 8 David Conlon, Jacob Fox, János Pach, Benny Sudakov, and Andrew Suk. Ramsey-type results for semi-algebraic relations. *Transactions of the American Mathematical Society*, 366:5043–5065, 2014.
- 9 Adrian Dumitrescu, André Schulz, Adam Sheffer, and Csaba D. Tóth. Bounds on the maximum multiplicity of some common geometric graphs. *SIAM Journal on Discrete Mathematics*, 27(2):802–826, 2013.
- 10 Herbert Edelsbrunner and Leonidas J. Guibas. Topologically Sweeping an Arrangement. *Journal of Computer and System Sciences*, 38(1):165–194, 1989.
- 11 György Elekes, Melvyn B. Nathanson, and Imre Z. Ruzsa. Convexity and Sumsets. *Journal of Number Theory*, 83(2):194–201, 2000.
- 12 Paul Erdős. Appendix, in Klaus F. Roth, On a problem of Heilbronn. *Journal of the London Mathematical Society*, 26:198–204, 1951.
- 13 Paul Erdős. Some more problems on elementary geometry. *Gazette of the Australian Mathematical Society*, 5(2):52–54, 1978.
- 14 Paul Erdős and György Szekeres. On some extremum problems in elementary geometry. *Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Mathematica*, 3-4:53–62, 1960/1961.
- 15 Panos Giannopoulos, Christian Knauer, and Daniel Werner. On the Computational Complexity of Erdős–Szekeres and Related Problems in  $\mathbb{R}^3$ . In *Proc. 21st European Symposium on Algorithms*, volume 8125 of *LNCS*, pages 541–552. Springer, 2013.
- 16 Ronald L. Graham. Euclidean Ramsey theory. In Jacob E. Goodman, Joseph O’Rourke, and Csaba D. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 11, pages 281–297. CRC Press, Boca Raton, FL, 3rd edition, 2017.
- 17 Richard K. Guy and Patrick A. Kelly. The No-Three-in-Line-Problem. *Canadian Mathematical Bulletin*, 11:527–531, 1968.
- 18 Norbert Hegyvári. On consecutive sums in sequences. *Acta Mathematica Hungarica*, 48(1-2):193–200, 1986.
- 19 Gyula Károlyi and Pavel Valtr. Configurations in  $d$ -space without large subsets in convex position. *Discrete & Computational Geometry*, 30(2):277–286, 2003.
- 20 Joseph S.B. Mitchell, Günter Rote, Gopalakrishnan Sundaram, and Gerhard Woeginger. Counting convex polygons in planar point sets. *Information Processing Letters*, 56(1):45–49, 1995.
- 21 Michael S. Payne and David R. Wood. On the general position subset selection problem. *SIAM Journal on Discrete Mathematics*, 27(4):1727–1733, 2013.
- 22 Attila Pór and David R. Wood. No-three-in-line-in-3D. *Algorithmica*, 47(4):481–488, 2007.

- 23 Orit E. Raz, Micha Sharir, and Ilya D. Shkredov. On the number of unit-area triangles spanned by convex grids in the plane. *Computational Geometry: Theory and Applications*, 62:25–33, 2017.
- 24 Orit E. Raz, Micha Sharir, and József Solymosi. Polynomials vanishing on grids: The Elekes–Rónyai problem revisited. *American Journal of Mathematics*, 138(4):1029–1065, 2016.
- 25 Orit E. Raz, Micha Sharir, and Frank De Zeeuw. Polynomials vanishing on Cartesian products: The Elekes–Szabó theorem revisited. *Duke Mathematical Journal*, 165(18):3517–3566, 2016.
- 26 David Alan Rosenthal. *The classification of the order indiscernibles of real closed fields and other theories*. PhD thesis, University of Wisconsin–Madison, 1981.
- 27 Tomasz Schoen and Ilya D. Shkredov. On Sumsets of Convex Sets. *Combinatorics, Probability and Computing*, 20(5):793–798, 2011.
- 28 Ryan Schwartz, József Solymosi, and Frank de Zeeuw. Extensions of a result of Elekes and Rónyai. *Journal of Combinatorial Theory, Series A*, 120(7):1695–1713, 2013.
- 29 Andrew Suk. On the Erdős–Szekeres convex polygon problem. *Journal of the American Mathematical Society*, 30:1047–1053, 2017.
- 30 Pavel Valtr. Sets in  $\mathbb{R}^d$  with no large empty convex subsets. *Discrete Mathematics*, 108(1–3):115–124, 1992.
- 31 Eric W. Weisstein. No-Three-in-a-Line-Problem. online, 2005. Available from: <http://mathworld.wolfram.com/No-Three-in-a-Line-Problem.html>.



# Smallest $k$ -Enclosing Rectangle Revisited

Timothy M. Chan

Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA  
tmc@illinois.edu

Sariel Har-Peled

Dept. of Computer Science, University of Illinois at Urbana-Champaign, USA  
sariel@illinois.edu

---

## Abstract

Given a set of  $n$  points in the plane, and a parameter  $k$ , we consider the problem of computing the minimum (perimeter or area) axis-aligned rectangle enclosing  $k$  points. We present the first near quadratic time algorithm for this problem, improving over the previous near- $O(n^{5/2})$ -time algorithm by Kaplan et al. [23]. We provide an almost matching conditional lower bound, under the assumption that (min, +)-convolution cannot be solved in truly subquadratic time. Furthermore, we present a new reduction (for either perimeter or area) that can make the time bound sensitive to  $k$ , giving near  $O(nk)$  time. We also present a near linear time  $(1 + \varepsilon)$ -approximation algorithm to the minimum area of the optimal rectangle containing  $k$  points. In addition, we study related problems including the 3-sided, arbitrarily oriented, weighted, and subset sum versions of the problem.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Geometric optimization, outliers, approximation algorithms, conditional lower bounds

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.23

**Related Version** A full version is available at <https://arxiv.org/abs/1903.06785>.

**Funding** *Timothy M. Chan*: Supported in part by NSF AF award CCF-1814026.

*Sariel Har-Peled*: Supported in part by NSF AF award CCF-1421231.

## 1 Introduction

Given a set  $P$  of  $n$  points in the plane, and a parameter  $k$ , consider the problem of computing the smallest area/perimeter axis-aligned rectangle that contains  $k$  points of  $P$ . (Unless stated otherwise, rectangles are axis-aligned by default.) This problem and its variants have a long history. Eppstein and Erickson [18] studied an exhaustive number of variants of this problem for various shapes.

For the minimum perimeter variant, the first work on this problem seems to be Aggarwal et al. [1], who showed a brute force algorithm with running time  $O(n^3)$ . Recently, Kaplan et al. [23] gave an algorithm with running time  $O(n^{5/2} \log^2 n)$  that works for both minimum perimeter and area.

Several works derived algorithms with running time sensitive to  $k$ , the number of points in the shape. Aggarwal et al. [1] showed an algorithm for the minimum perimeter with running time  $O(k^2 n \log n)$ . This was improved to  $O(n \log n + k^2 n)$  by Eppstein and Erickson [18] or alternatively by Datta et al. [16]. Kaplan et al.'s algorithm [23] for the  $k$ -insensitive case, coupled with these previous techniques [18, 16], results in an  $O(n \log n + nk^{3/2} \log^2 k)$  running time, which is currently the state of the art.

Known techniques [18, 16] reduce the problem to solving  $O(n/k)$  instances of size  $O(k)$ . These reductions work only for the perimeter case, not the area case – in particular, there are incorrect attributions in the literature to results on the minimum area rectangle – see the introduction of de Berg et al. [17] for details. De Berg et al. described an algorithm with



© Timothy M. Chan and Sariel Har-Peled;

licensed under Creative Commons License CC-BY

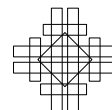
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 23; pp. 23:1–23:15

Leibniz International Proceedings in Informatics



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



running time  $O(n \log^2 n + nk^2 \log n)$  for minimum area. Both de Berg et al. [17] and Kaplan et al. [23] left as an open question whether there is a reduction from the minimum-area problem to about  $\tilde{O}(n/k)$  instances of size  $O(k)$ , where  $\tilde{O}$  hides<sup>1</sup> polynomial factors in  $\log n$  and  $1/\varepsilon$ . Such a reduction would readily imply an improved algorithm.

### Our results

We revisit the above problems and provide significantly improved algorithms:

1. **Exact smallest  $k$ -enclosing rectangle.** In Section 2.1 we describe an algorithm for the minimum  $k$ -enclosing rectangle (either area or perimeter) with running time  $O(n^2 \log n)$  (see Theorem 2). It is based on a new divide-and-conquer approach, which is arguably simpler than Kaplan et al.'s algorithm. Known reductions mentioned above then lead to an  $O(n \log n + nk \log k)$ -time algorithm for computing the minimum perimeter rectangle.
2.  **$k$ -sensitive running time for smallest area.** In Section 2.2 we describe a reduction of the minimum-area problem to  $O(\frac{n}{k} \log \frac{n}{k})$  instances of size  $O(k)$  (see Theorem 8). Our reduction uses *shallow cutting* for 3-sided rectangular ranges [22] and is conceptually simple.

Plugging this the aforementioned new  $O(n^2 \log n)$ -time algorithm leads to  $O(nk \log \frac{n}{k} \log k)$  time algorithm for computing the minimum area  $k$ -enclosing rectangle (see Corollary 9). Thus, our new result strictly improves upon both Kaplan et al.'s and de Berg et al.'s results for all  $k$ , from constant to  $\Theta(n)$ .

The smallest enclosing rectangle problem is amenable to sampling. Kaplan et al. used samples in an approximation algorithm, with running time  $\tilde{O}(n/k)$ , that computes a rectangle containing at least  $(1 - \varepsilon)k$  points of a prescribed perimeter, where  $k$  is the maximum number of points in any such rectangle. Similarly, using relative approximations [21], de Berg et al. [17] showed an algorithm that computes, in  $\tilde{O}(n)$  time, a rectangle containing  $\geq (1 - \varepsilon)k$  points, where  $k$  is the maximum number of points in any rectangle of a prescribed area. The “dual” problem, of approximating the minimum area rectangle containing  $k$  points seems harder, since sampling does not directly apply to it.

1. **Approximating the area of the smallest  $k$ -enclosing rectangle.** In Section 2.3, we present an approximation algorithm that computes, in  $O(n \log n)$  expected time, a rectangle containing  $k$  points of area  $\leq (1 + \varepsilon)\alpha^*$ , for a constant  $\varepsilon \in (0, 1)$ , where  $\alpha^*$  is the smallest-area of such a rectangle (see Theorem 13).

We next present a flotilla of related results:

1. **3-sided smallest  $k$ -enclosing rectangle.** In Section 3.1 we (slightly) speed up the exact algorithm for the 3-sided rectangles case (i.e., rectangles that must have their bottom edge on the  $x$ -axis). The running time is  $O(n^2/2^{\Omega(\sqrt{\log n})})$ , and is obtained using known results on the *(min, +)-convolution* problem [7, 28] (see Theorem 16).
2. **Arbitrarily oriented smallest  $k$ -enclosing rectangle.** In Section 3.2 we briefly consider the variant where the rectangle may not be axis-aligned. We show that this problem can be solved in  $O(n^3 \log n + n^3 k/2^{\Omega(\sqrt{\log k})})$  time, slightly improving a previous result of  $O(n^3 k)$  [15] when  $k$  is not too small.

---

<sup>1</sup> We reserve the right, in the future, to use the  $\tilde{O}$  to hide any other things we do not like.

3. Minimum-weight  $k$ -enclosing rectangle. In Section 3.3 we show how to extend our  $O(n^2 \log n)$ -time algorithm to the related problem of finding a minimum-weight rectangle that contains  $k$  points, for  $n$  given weighted points in the plane (see Theorem 17).
4. Subset sum for  $k$ -enclosing rectangle. In Section 3.4, we study the problem of finding a rectangle that contains  $k$  points and has a prescribed weight  $W$  (or as close as one can get to it). The running time of the new algorithm is  $O(n^{5/2} \log n)$  (see Theorem 19).
5. Conditional lower bound. In Section 3.5, we prove that our near quadratic algorithm for exact minimum (perimeter or area)  $k$ -enclosing rectangle is near optimal up to an arbitrarily small polynomial factor, under a “popular” conjecture that the (min,+)-convolution problem cannot be solved in truly subquadratic time [14].

## 2 Smallest $k$ -enclosing rectangle

### 2.1 An exact near-quadratic algorithm

Our  $O(n^2 \log n)$ -time algorithm for minimum  $k$ -enclosing rectangles is based on divide-and-conquer. It has some similarity with an  $O(n^2)$ -time divide-and-conquer algorithm by Barbay et al. [5] for a different problem (finding the minimum-weight rectangle for  $n$  weighted points in the plane, without any  $k$ -enclosing constraint), but the new algorithm requires more ingenuity.

We start with a semi-dynamic data structure for a 1D subproblem:

► **Lemma 1.** *Given a set  $P$  of  $n$  points in 1D with  $q$  marked points, and an integer  $k$ , we can maintain an  $O(q^2)$ -space data structure, with  $O(n \log n + nq)$  preprocessing time, that supports the following operations:*

- report the shortest interval containing  $k$  points of  $P$  in  $O(q)$  time;
- delete a marked point in  $O(q)$  time;
- unmark a marked point in  $O(q)$  time.

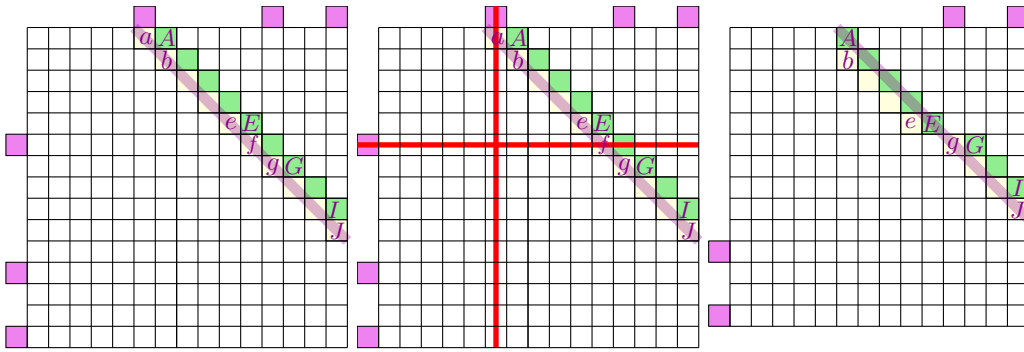
**Proof.** Sort the points  $P$ , and let  $p_1, \dots, p_n$  be resulting order. Consider the (implicit) matrix  $M = P - P$ . Formally, the entry  $M_{ij}$  is  $p_j - p_i$  (we are interested only in the top right part of this matrix) – such an entry can be computed in  $O(1)$  time directly from the sorted point set. The optimal quantity of interest is the minimum on the  $k$ th diagonal; that is,  $\alpha(M) = \min_i M_{i, i+k-1}$ . When a marked point get deleted, this corresponds to deleting a row and a column of  $M$  – the quantity of interest remains the minimum along the  $k$ th diagonal. Such a deletion, as far as a specific entry of the top right of the matrix is concerned, either (i) removes it, (ii) keeps it in its place, (iii) shift it one diagonal down as its moves left, or (iv) keep it on the same diagonal as it shifts both up and left (see Figure 1).

In particular, any sequence of at most  $q$  deletions of elements can shift an entry in the matrix at most  $q$  diagonals down. This implies that we need to keep track only of the  $k, \dots, k+q$  diagonals of this matrix. To do better, observe that if we track the elements of an original diagonal of interest, the deletions can fragment the diagonal into at most  $O(q)$  groups, where each group still appear as contiguous run of the original diagonal.

To this end, let a **fragment** of a diagonal be either (i) a singleton entry that appears in a row or column of a marked point, or (ii) a maximum contiguous portion of the diagonal which does not touch any singleton entries from (i). It is easy to verify that the  $k$ th diagonal of the matrix at any given point in time is made out of a sequence of at most  $3k$  fragments, where each fragment is an original fragment of one of the diagonals in the range  $k, \dots, k+q$ .

As such, instead of storing all the elements of a fragment, we only maintain the minimum entry of the fragment (together with the information of what pairs of points it corresponds to). After this compression, a diagonal of interest can be represented as a linked list of  $O(q)$





■ **Figure 1** Behold the matrix!

fragment summaries. In the preprocessing stage, the algorithm computes this representation for the  $k$  to  $k + q$  diagonals (using this representation). This requires  $O(q^2)$  space, and  $O(nq)$  time.

A deletion of a marked point then corresponds to taking a contiguous block of linked fragments at the  $i$ th list and moving it to list  $i - 1$ , doing this surgery for  $i = k, \dots, k + q$ . The blocks being moved start and end in singleton entries that correspond to the deleted point. We also need to remove these two singleton elements, and merge the two adjacent fragment summaries that are no longer separated by a singleton. This surgery for all the  $q + 1$  lists of interest can be done in  $O(q)$  time (we omit the tedious but straightforward details).

A query corresponds to scanning the  $k$ th diagonal and reporting the minimum value stored along it. An unmarking operation corresponds to merging two fragment summaries and the singleton separating them into a single fragment summary, and doing this for all the  $q + 1$  lists. Both operations clearly can be done in  $O(q)$  time. ◀

► **Theorem 2.** *Given a set  $P$  of  $n$  points in the plane and an integer  $k$ , one can compute, in  $O(n^2 \log n)$  time, the smallest-area/perimeter axis-aligned rectangle enclosing  $k$  points.*

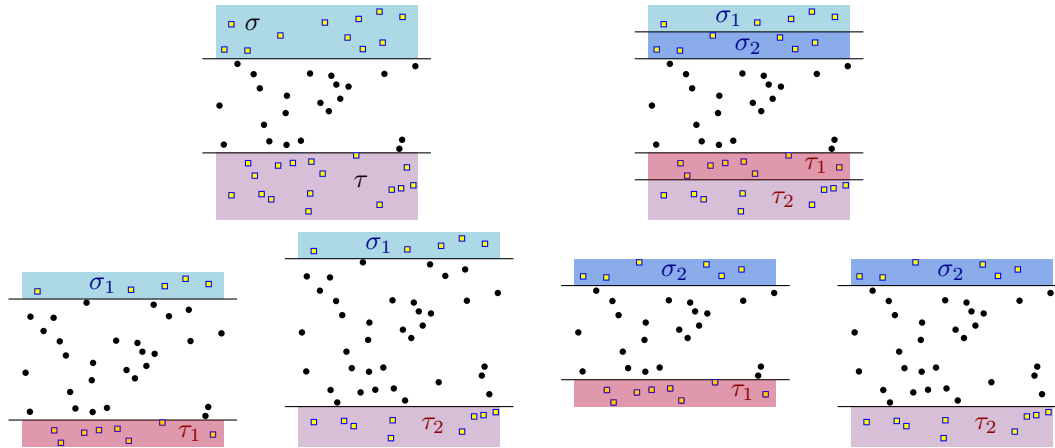
**Proof.** We do divide-and-conquer by  $y$ -coordinates. Given a set  $P$  of  $n$  points in the plane, and horizontal slabs  $\sigma$  and  $\tau$ , each containing  $q$  points of  $P$ , we describe a recursive algorithm to find a smallest  $k$ -enclosing axis-aligned rectangle containing  $P$ , under the restriction that the top edge is inside  $\sigma$  and the bottom edge is inside  $\tau$ . It is assumed that either  $\sigma$  is completely above  $\tau$ , or  $\sigma = \tau$ . It is also assumed that all points above  $\sigma$  or below  $\tau$  have already been deleted from  $P$ . There can still be a large number of points in  $P - (\sigma \cup \tau)$  (recursion will lower  $q$  but not necessarily  $n$ ). We will not explicitly store the points in  $P - (\sigma \cup \tau)$ , but rather “summarize” the points in an  $O(q^2)$ -space structure. Namely, we assume that the  $x$ -coordinates of  $P$  are maintained in the 1D data structure  $\mathcal{S}$  of Lemma 1, where the marked points are the  $O(q)$  points in  $P \cap (\sigma \cup \tau)$ .

The algorithm proceeds as follows:

1. If  $q = 1$ , then report the answer by querying  $\mathcal{S}$  in  $O(1)$  time. Else:
2. Divide  $\sigma$  into two horizontal subslabs  $\sigma_1$  and  $\sigma_2$ , each containing  $q/2$  points of  $P$ . Likewise divide  $\tau$  into  $\tau_1$  and  $\tau_2$ .
3. For each  $i, j \in \{1, 2\}$ , recursively solve the problem for the slabs  $\sigma_i$  and  $\tau_j$ ;<sup>2</sup> to prepare for the recursive call, make a copy of  $\mathcal{S}$ , delete the (marked) points in  $P \cap (\sigma \cup \tau)$  above

<sup>2</sup> If  $\sigma = \tau$ , one of the four recursive calls is unnecessary.

$\sigma_i$  or below  $\tau_j$ , and unmark the remaining points in  $P \cap (\sigma \cup \tau) - (\sigma_i \cup \tau_j)$ , as shown in Figure 2. The time needed for these  $O(q)$  deletions and unmarkings, and for copying  $\mathcal{S}$ , is  $O(q^2)$  (we emphasize that this bound is independent of  $n$ ).



■ **Figure 2** Divide et impera for the problem at hand.

The running time satisfies the recurrence

$$T(n, q) = 4T(n, q/2) + O(q^2),$$

with  $T(n, 1) = O(1)$ , which gives  $T(n, q) = O(q^2 \log q)$ . Initially,  $\sigma = \tau$  is the entire plane, with  $q = n$ ; the data structure  $\mathcal{S}$  can be preprocessed in  $O(n^2)$  time. Thus, the total running time is  $O(n^2 \log n)$ . ◀

One can readily get an algorithm with  $k$ -sensitive running time for the perimeter case, by reducing the problem into  $O(n/k)$  instance of size  $O(k)$ . This reduction is well known [18, 16] in this case – approximate the smallest enclosing disk containing  $k$  points in  $O(n \log n)$  time, partition the plane into a grid with side length proportional to the radius of this disk, and then solve the problem for each cluster (i.e.,  $3 \times 3$  group of grid cells) that contains at least  $k$  points of  $P$ , using our above algorithm. We thus get the following.

► **Corollary 3.** *Given a set  $P$  of  $n$  points in the plane and an integer  $k$ , one can compute, in  $O(n \log n + nk \log k)$  time, the smallest-perimeter axis-aligned rectangle enclosing  $k$  points of  $P$ .*

The  $O(n \log n)$  term can be eliminated in the word RAM model, using a randomized linear-time algorithm for approximate smallest  $k$ -enclosing disk [20] (which requires integer division and hashing).

A similar reduction for the minimum-area case is more challenging, and was left as an open problem in previous work [23]. The difficulty arises because the optimal-area rectangle may be long and thin, with side length potentially much bigger than the radius of the minimum  $k$ -enclosing disk. Nonetheless, we show that such a reduction is possible (with an extra logarithmic factor) in the next subsection.

## 2.2 $k$ -sensitive running time for smallest area

Our starting point is a shallow cutting lemma for 3-sided ranges [22]. (It can be viewed as an orthogonal variant of Matoušek’s shallow cutting lemma for halfspaces [24].)

► **Lemma 4** ([22]). Given a set  $P$  of  $n$  points in the plane, lying above a horizontal line  $\ell$ , and a parameter  $k$ , one can compute a family  $\mathcal{F}$  of at most  $2 \lceil n/k \rceil$  subsets of  $P$ , each of size at most  $6k$ . The collection of sets can be computed in  $O(n)$  time if the  $x$ -coordinates have been pre-sorted. For any axis-aligned rectangle  $R$  with its bottom edge lying on  $\ell$ , that contains less than  $k$  points of  $P$ , we have  $P \cap R \subseteq PA$  for some  $PA \in \mathcal{F}$ .

► **Definition 5.** Let  $\mathcal{R}$  be the set of all axis-aligned rectangles in the plane. A **scoring function** is a function  $f : \mathcal{R} \rightarrow \mathbb{R}$ , with the following properties:

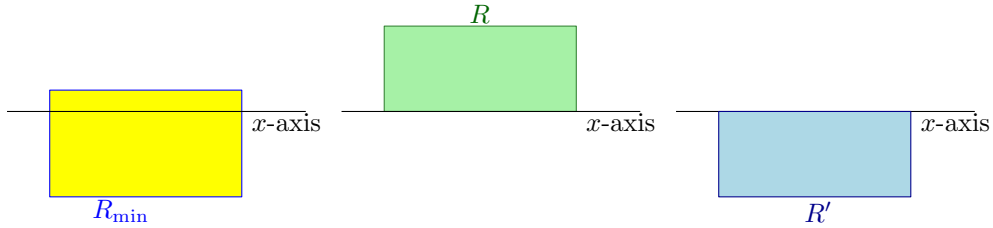
1. Translation invariant:  $\forall p \in \mathbb{R}^2$ , we have  $f(p + R) = f(R)$ .
2. Monotonicity:  $\forall R, R' \in \mathcal{R}$ , such that  $R \subseteq R'$ , we have that  $f(R) \leq f(R')$ .

Functions that satisfy the above definition include area, perimeter, diameter, and enclosing radius of a rectangle.

For a set  $U \subseteq \mathbb{R}^d$ , let  $\uparrow U = \{(x, -y) \mid (x, y) \in U\}$  be the **reflection** of  $U$  through the  $x$ -axis. Similarly, let  $|U|_y = \{(x, |y|) \mid (x, y) \in U\}$  be the **folding** of  $U$  through the  $x$ -axis.

► **Lemma 6.** Given a set  $P$  of  $n$  points in the plane, a parameter  $k$ , and a scoring function  $f$ , let  $R_{\min}$  be the minimum score axis-aligned rectangle that contains  $k$  points of  $P$ , and intersects the  $x$ -axis. Then, one can compute a family  $\mathcal{F}$  of  $O(n/k)$  subsets of  $P$ , such that (i) each set of  $\mathcal{F}$  is of size  $\leq 12k$ , and (ii)  $R_{\min} \cap P \subseteq PA$ , for some  $PA \in \mathcal{F}$ .

**Proof.** Let  $P' = |P|_y$  be the “folding” of  $P$  over the  $x$ -axis, and let  $\mathcal{F}'$  be the cover of  $P'$  by sets of size  $\leq 12k$ , as computed by Lemma 4 for rectangles containing at most  $2k$  points. Let  $\mathcal{F}$  be the corresponding family of sets for  $P$ . We claim that  $\mathcal{F}$  has the desired property.



Let  $R = |R_{\min}|_y$  be the folding of  $R_{\min}$ , and let  $R' = \uparrow R$ . Observe that  $f(R) = f(R') \leq f(R_{\min})$  because of the translation invariance of  $f$ , and monotonicity of  $f$ .

If  $|R \cap P| > k$  then one can shrink it so that it contains only  $k$  points of  $P$ , but this would imply that  $R_{\min}$  is not the minimum, a contradiction. The case that  $|R' \cap P| > k$  leads to a similar contradiction. We conclude that  $R \cup R'$  contains at most  $2k$  points of  $P$ . Implying that  $R = |R_{\min}|_y$  contains at most  $2k$  points of  $P'$ . As such, there is a set  $PA' \in \mathcal{F}'$  that contains  $R \cap R'$ . Now, let  $PA$  be the corresponding set in  $\mathcal{F}$  to  $PA'$ . Since  $R_{\min} \subseteq R \cup R'$ , it follows that  $R_{\min} \cap P \subseteq (R \cup R') \cap P \subseteq PA$ , as desired. ◀

► **Lemma 7.** Given a set  $P$  of  $n$  points in the plane, a parameter  $k$ , and a scoring function  $f$ , let  $R_{\min}$  be the minimum score rectangle that contains  $k$  points of  $P$ . One can compute, in  $O(n \log n)$  time, a family  $\mathcal{F}$  of  $O(\frac{n}{k} \log \frac{n}{k})$  subsets of  $P$ , such that (i) each subset of  $\mathcal{F}$  is of size  $\leq 12k$ , and (ii)  $R_{\min} \cap P \subseteq PA$ , for some  $PA \in \mathcal{F}$ .

**Proof.** Find a horizontal line  $\ell$  that splits  $P$  evenly, and compute the family of Lemma 6. Now recurse on the points above  $\ell$  and the points below  $\ell$ . The recursion bottoms out when the number of points is  $\leq 12k$ . The correctness is by now standard – as soon as a recursive call picks a line that stabs the optimal rectangle, the family generated for this line contains the desired set. The  $x$ -coordinates need to be pre-sorted just once at the beginning. ◀

► **Theorem 8.** Let  $P$  be a set of  $n$  points in the plane,  $k$  be a parameter,  $f$  be a scoring function for rectangles, and let `alg` be an algorithm that computes, in  $T_{\text{alg}}(m)$  time, the axis-aligned rectangle containing  $k$  points in a set of  $m$  points that minimizes  $f$ . Then one can compute the rectangle containing  $k$  points of  $P$  that minimizes  $f$ , in time  $O(n \log n + T_{\text{alg}}(12k) \frac{n}{k} \log \frac{n}{k})$ .

**Proof.** Compute the family of sets  $\mathcal{F}$  using Lemma 7, and then apply `alg` to each set in this family. ◀

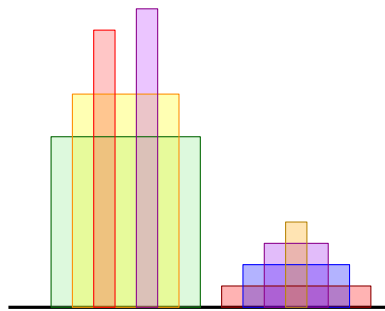
Combining Theorem 2 with Theorem 8 gives the following.

► **Corollary 9.** Given a set  $P$  of  $n$  points in the plane and an integer  $k$ , one can compute, in  $O(nk \log \frac{n}{k} \log k)$  time, the smallest-area axis-aligned rectangle enclosing  $k$  points of  $P$ .

For the case when  $t = n - k$  is very small (i.e., finding the smallest enclosing axis-aligned rectangle with  $t$  outliers), there is an easy reduction (for both perimeter and area) yielding  $O(n + T_{\text{alg}}(4t))$  time [26, 2], by keeping the  $t$  leftmost/rightmost/topmost/bottommost points. Immediately from Theorem 2, we get  $O(n + t^2 \log t)$  running time.

### 2.3 An approximation algorithm for smallest area

In this subsection, we give an efficient approximation algorithm for the smallest-area  $k$ -enclosing rectangle problem. The smallest perimeter case is straightforward to approximate, by grid rounding, but the area case is tougher, again because the optimal rectangle may be long and thin.



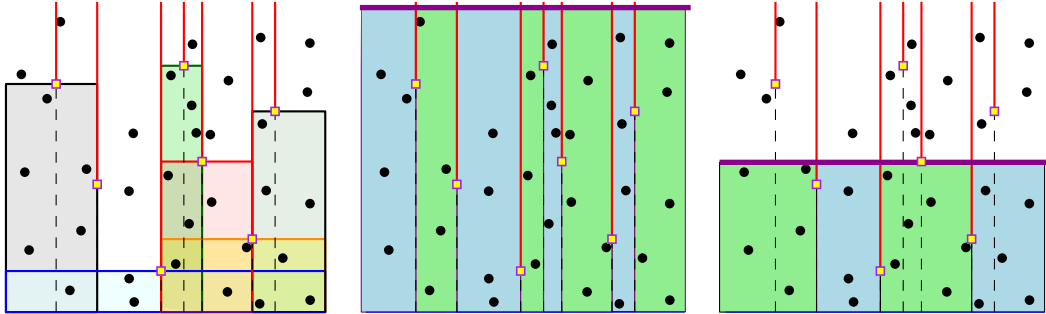
► **Definition 10.** A *laminar* family of 3-sided rectangles is a collection  $\mathcal{R}$  of axis-aligned rectangles with the bottom edges lying on the  $x$ -axis, such that for every pair of rectangles  $[a, b] \times [0, c]$  and  $[a', b'] \times [0, c']$  in  $\mathcal{R}$ , one of the following is true:

- $[a, b] \cap [a', b'] = \emptyset$ , or
- $[a, b] \subseteq [a', b']$  and  $c > c'$ , or
- $[a', b'] \subseteq [a, b]$  and  $c' > c$ .

Standard range trees can answer orthogonal range counting queries (counting the number of points inside rectangular ranges) in logarithmic time per query (this has been improved to  $O(\sqrt{\log n})$  in the offline setting by Chan and Pătraşcu [12]). The following lemma shows how to achieve constant time per query in the offline laminar special case, which will be useful later in our approximation algorithm.

► **Lemma 11.** Let  $P$  be a set of  $n$  points, and let  $\mathcal{R}$  be a laminar family of  $O(n)$  3-sided rectangles in the plane. Suppose that we are given a **designated point** on the top edge of each rectangle in  $\mathcal{R}$ , and the  $x$ - and  $y$ -coordinates of all the designated points and all the points of  $P$  have been pre-sorted. Then we can count, for each rectangle  $R \in \mathcal{R}$ , the number of points of  $P$  inside the rectangle, in  $O(n)$  total time.

**Proof.** We describe a sweep algorithm, with a horizontal sweep line  $\ell$  moving downward. Let  $X$  denote the set of  $x$ -coordinates of all points of  $P$  and all designated points of the rectangles of  $\mathcal{R}$ . Consider the union of  $R \cap \ell$  over all  $R \in \mathcal{R}$ ; it can be expressed as a union of disjoint intervals. Let  $\mathcal{I}_\ell$  be the  $x$ -projection of these disjoint intervals. Store the following collection  $\Gamma_\ell$  of disjoint sets in a *union-find* data structure [27]: for each interval  $I \in \mathcal{I}_\ell$ , define the set  $X \cap I$ , and for each  $a \in X$  not covered by  $\mathcal{I}_\ell$ , define the singleton set  $\{a\}$ . Create a linked list  $L_\ell$  containing these sets in  $\Gamma_\ell$  ordered by  $x$ . For each set in  $\Gamma_\ell$ , we store a count of the number of points of  $P$  below  $\ell$  with  $x$ -coordinates inside the set.



Suppose that the sweep line  $\ell$  hits the top edge of a rectangle  $R$  with  $x$ -projection  $[a, b]$ . By definition of a laminar family, any interval in  $\mathcal{I}_\ell$  that intersects  $[a, b]$  must be contained in  $[a, b]$ . We find the set in  $\Gamma_\ell$  that contains the  $x$ -coordinate of the designated point of  $R$ . From this set, we walk through the list  $L_\ell$  in both directions to find all sets contained in  $[a, b]$ , and replace these sets with their union in  $\Gamma_\ell$  and  $L_\ell$ . The count for the new set is the sum of the counts of the old sets; this also gives the output count for the rectangle  $R$ .

Next, suppose that the sweep line  $\ell$  hits a point  $p \in P$ . We find the set in  $\Gamma_\ell$  that contains the  $x$ -coordinate of  $p$ , and decrement its count. (For example, if the set is a singleton, its count changes from 1 to 0.)

The entire sweep performs  $O(n)$  union and find operations. Gabow and Tarjan [19] gave a linear-time union-find algorithm for the special case where the “union tree” is known in advance; their algorithm is applicable, since the union tree here is just a path of the elements ordered by  $x$ . ◀

We first solve the approximate decision problem:

► **Lemma 12.** *Given a set  $P$  of  $n$  points in the plane, a value  $\alpha$ , and parameters  $k$  and  $\varepsilon \in (0, 1)$ , one can either compute a  $k$ -enclosing axis-aligned rectangle  $R'$  such that  $\text{area}(R') \leq (1 + O(\varepsilon))\alpha$ , or conclude that the smallest-area  $k$ -enclosing axis-aligned rectangle has area greater than  $\alpha$ . The running time of the algorithm is  $O(\varepsilon^{-3} \log \varepsilon^{-1} \cdot n \log n)$ .*

**Proof.** It is sufficient to solve the problem for the case where the rectangle must intersect a horizontal line  $\ell$  in  $O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n)$  time, assuming that the  $x$ - and  $y$ -coordinates of the given points  $P$  have been pre-sorted. Then standard divide-and-conquer by  $y$ -coordinates gives an  $O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n)$ -time algorithm for the general problem. Pre-sorting needs to be done only once at the beginning.

Without loss of generality, assume that  $\ell$  is the  $x$ -axis. Suppose there exists a rectangle  $R^*$  intersecting  $\ell$  that contains at least  $k$  points and has area at most  $\alpha$ . By symmetry, we may assume that  $R^*$  has greater area above  $\ell$  than below, and that the top edge passes through some input point  $p = (p_x, p_y) \in P$ . Then the height  $h^*$  of  $R^*$  is between  $p_y$  and  $2p_y$ , and the width  $w^*$  is at most  $\alpha/p_y$ .

Without loss of generality, assume that all  $x$ -coordinates are in  $[0, 1/3]$ . Define a one-dimensional **quadtrees interval** (also known as a *dyadic interval*) to be an interval of the form  $[\frac{m}{2^i}, \frac{m+1}{2^i}]$ . It is known that every interval of length  $w < 1/3$  is contained in a quadtree interval of length  $O(w)$  after shifting the interval by one of two possible values  $s \in \{0, 1/3\}$  (this is a special case of a shifting lemma for  $d$ -dimensional quadtrees [6, 8]). Thus, suppose that  $[p_x - \alpha/p_y, p_x + \alpha/p_y]$  is contained in the interval  $[\frac{m}{2^i} + s, \frac{m+1}{2^i} + s]$ , where the length  $\frac{1}{2^i}$  is equal to the smallest power of 2 greater than  $c\alpha/p_y$ , for some constant  $c$ . (Note that  $i$  is a nondecreasing function of  $p_y$ .) Without loss of generality, assume that  $1/\varepsilon = 2^E$  for an integer  $E$ . Define a family of  $O(1/\varepsilon^3)$  **canonical rectangles** of the form

$$[\frac{m}{2^i} + \frac{j}{2^{i+E}} + s, \frac{m}{2^i} + \frac{j'}{2^{i+E}} + s] \times [-j''\varepsilon p_y, p_y]$$

over all possible indices  $j, j', j'' \in \{0, \dots, 1/\varepsilon\}$  such that  $p_x \in [\frac{m}{2^i} + \frac{j}{2^{i+E}} + s, \frac{m}{2^i} + \frac{j'}{2^{i+E}} + s]$ .

By rounding,  $R^*$  is contained in a canonical rectangle  $R'$  with height at most  $h^* + O(\varepsilon)p_y \leq (1 + O(\varepsilon))h^*$  and width at most

$$w^* + O(\varepsilon)\alpha/p_y \leq (1 + O(\varepsilon))\alpha/h^*,$$

and thus area at most  $(1 + O(\varepsilon))\alpha$ . So, it suffices to count the number of points inside each canonical rectangle and return the smallest area among those rectangles containing at least  $k$  points.

To speed up range counting, observe that for canonical rectangles with the same  $j, j', j''$ , the same  $s \in \{0, 1/3\}$ , and the same value for  $(i \bmod E)$ , the portion of the rectangles above (resp. below)  $\ell$  forms a laminar family. This is because: (i) in the  $x$ -projections, if a pair of intervals intersects, one interval must be contained in the other; (ii) as the height of the 3-sided rectangle increases,  $p_y$  increases, and so  $i$  can only increase (or stay the same), and so the width of the rectangle can only decrease (or stay the same). Thus, we can apply Lemma 11 to compute the counts of the points inside each rectangle, for all canonical rectangles with a fixed  $j, j', j'', s$  and  $(i \bmod E)$ , in  $O(n)$  time (for each canonical rectangle, we can use a point  $(p_x, p_y) \in P$  on the top edge, and a corresponding point  $(p_x, -j''\varepsilon p_y)$  on the bottom edge, as the designated points). The number of choices for  $j, j', j'', s$  and  $(i \bmod E)$  is  $O((1/\varepsilon)^3 \log(1/\varepsilon))$ . ◀

► **Theorem 13.** *Given a set  $P$  of  $n$  points in the plane, and parameters  $k$  and  $\varepsilon \in (0, 1)$ , one can compute a  $k$ -enclosing rectangle  $R'$  such that  $\text{area}(R') \leq (1 + \varepsilon)\text{opt}(P, k)$ , where  $\text{opt}(P, k)$  is the area of the smallest axis-aligned rectangle containing  $k$  points of  $P$ . The expected running time of the algorithm is  $O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n)$ .*

**Proof.** We can use known techniques for reducing optimization problems to decision problems. We give a self-contained description of one approach based on Chan’s randomized technique [9].

Let  $b$  be a sufficiently large constant. Divide the plane into  $b$  columns (vertical slabs) each containing  $n/b$  points. Similarly divide the plane into  $b$  rows (horizontal slabs) each containing  $n/b$  points. These steps take linear time by invoking a selection algorithm  $O(b)$  times. For each quadruple  $\tau = (c, c', r, r')$  where  $c$  and  $c'$  are columns (with  $c$  left of  $c'$  or  $c = c'$ ) and  $r$  and  $r'$  are rows (with  $r$  below  $r'$  or  $r = r'$ ), consider the subproblem of finding the smallest-area rectangle containing  $k$  points of  $P$ , subject to the extra constraints that the left edge of the rectangle lies in  $c$ , the right edge lies in  $c'$ , the bottom edge lies in  $r$ , and the top edge lies in  $r'$ . To solve this subproblem, it suffices to consider the at most  $4n/b$  points in  $P \cap (c \cup c' \cup r \cup r')$ . To ensure that the extra constraints are satisfied, we add  $4n/b$  copies of the four intersection points formed by the right boundary of  $c$ , the left boundary of  $c'$ , the

## 23:10 Smallest $k$ -Enclosing Rectangle Revisited

top boundary of  $r$ , and the bottom boundary of  $r'$ ; and we add  $16n/b$  to  $k$ . (Straightforward modifications can be made in the special case when  $c = c'$  or  $r = r'$ .) Let  $P_\tau$  be the resulting point set of size at most  $20n/b$  points, and  $k_\tau$  be the resulting value of  $k$ . We thus have

$$\text{opt}(P, k) = \min_{\tau} \text{opt}(P_\tau, k_\tau).$$

To compute an approximation to the minimum, we consider the at most  $b^4$  quadruples in *random order*  $\tau_1, \tau_2, \dots$  and keep track of an approximate minimum  $\alpha$  with the invariant that  $\alpha \leq \min\{\text{opt}(P_{\tau_1}, k_{\tau_1}), \dots, \text{opt}(P_{\tau_{i-1}}, k_{\tau_{i-1}})\} < (1 + \varepsilon)\alpha$  after the  $(i - 1)$ th iteration. Let  $\varepsilon'$  be such that  $(1 + \varepsilon')^2 = 1 + \varepsilon$ ; note that  $\varepsilon' = \Theta(\varepsilon)$ . At the  $i$ th iteration, we run the approximate decision procedure for  $P_{\tau_i}$  twice, at values  $\alpha$  and  $\alpha/(1 + \varepsilon')$ , which allows us to conclude one of the following:

- $\text{opt}(P_{\tau_i}, k_{\tau_i}) \geq \alpha$ . In this case, we can continue to the next iteration and the invariant is maintained.
- $\alpha/(1 + \varepsilon') \leq \text{opt}(P_{\tau_i}, k_{\tau_i}) < (1 + \varepsilon')\alpha$ . In this case, we reset  $\alpha$  to  $\alpha/(1 + \varepsilon')$  and the invariant is maintained.
- $\text{opt}(P_{\tau_i}, k_{\tau_i}) < \alpha$ . In this case, we recursively compute an approximation  $\alpha_i$  to the quantity  $\text{opt}(P_{\tau_i}, k_{\tau_i})$ , satisfying  $\alpha_i \leq \text{opt}(P_{\tau_i}, k_{\tau_i}) < (1 + \varepsilon)\alpha_i$ . We reset  $\alpha$  to  $\alpha_i$  and the invariant is maintained.

We land in the third case only if  $\text{opt}(P_{\tau_i}, k_{\tau_i})$  is the smallest among the  $i$  values  $\text{opt}(P_{\tau_1}, k_{\tau_1}), \dots, \text{opt}(P_{\tau_i}, k_{\tau_i})$ , which happens with probability at most  $1/i$ . Thus, the expected number of recursive calls is bounded by the  $(b^4)$ th Harmonic number  $\sum_{i=1}^{b^4} 1/i < \ln(b^4) + 1$ . The expected running time satisfies the recurrence

$$T(n) \leq (4 \ln b + 1)T(20n/b) + O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n),$$

which gives  $T(n) = O((1/\varepsilon)^3 \log(1/\varepsilon) \cdot n \log n)$  when  $b = 1000$ , for example. ◀

### 3 Extensions

#### 3.1 3-sided smallest $k$ -enclosing rectangle

In this subsection, we give a slightly faster algorithm for the 3-sided variant of the problem, finding the smallest-area/perimeter rectangle enclosing  $k$  points, under the restriction that the bottom edge lies on the  $x$ -axis. The improvement uses the latest result on the  $(\min, +)$ -convolution problem, and is interesting in view of a reduction in Section 3.5 in the reverse direction, establishing essentially an equivalence of the 3-sided problem to  $(\min, +)$ -convolution.

► **Problem 14.**  $(\min, +)$ -Convolution. Given real numbers  $a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}$ , compute  $c_\ell = \min_{i=0}^{\ell} (a_i + b_{\ell-i})$  for all  $\ell = 0, \dots, 2n - 2$ .

Let  $T_{\text{convol}}(n)$  be the time complexity of the  $(\min, +)$ -convolution problem. As observed by Bremner et al. [7], the problem can be reduced to  $(\min, +)$ -matrix multiplication, and using the current best result by Williams [28] (derandomized by Chan and Williams [13]),  $T_{\text{convol}}(n) = O(n^2/2^{\Omega(\sqrt{\log n})})$ . We use  $(\min, +)$ -convolution to speed up the preprocessing time of the 1D data structure from Section 2.1.

► **Lemma 15.** *The preprocessing time in Lemma 1 can be reduced to  $O((n/q)T_{\text{convol}}(q) + q^3)$ .*



**Proof.** Divide the  $n \times n$  matrix  $M$  vertically into  $n/q$  submatrices  $M_1, \dots, M_{n/q}$  each of dimension  $n \times q$ . For each submatrix  $M_i$ , we consider the portions of the diagonals  $k, \dots, k + q$  that are within  $M_i$  – each such portion will be called a **chunk**. We precompute the minimum of the entries in each chunk. For a fixed  $i$ , this is equivalent to computing  $\min_{q_i < j \leq q(i+1)} (p_{j+\ell-1} - p_j)$  for all  $\ell \in \{k, \dots, k + q\}$ . Notice that after some massaging of the sequence (negating, reversing, and padding), this computation can be reduced to  $(\min, +)$ -convolution over  $O(q)$  elements, and can thus be done in  $O(T_{\text{convol}}(q))$  time. The total time over all  $i$  is  $O((n/q)T_{\text{convol}}(q))$ .

Recall that in the preprocessing algorithm in Lemma 1, we need to compute the minimum of each fragment in the  $k, \dots, k + q$  diagonals. Each fragment can be decomposed into some number of disjoint chunks plus  $O(q)$  extra elements. Over all  $O(q)$  diagonals, there are  $O(q^2)$  fragments and  $O(n/q \cdot q) = O(n)$  chunks in total. Thus, we can compute the minima of all fragments in  $O(q^2 \cdot q + n/q \cdot q) = O(q^3 + n)$  time, after the above precomputation of the minima of all chunks. ◀

▶ **Theorem 16.** *Given a set  $P$  of  $n$  points in the plane and integer  $k$ , one can compute, in  $O(n^2/2^{\Omega(\sqrt{\log n})})$  time, the smallest-area/perimeter axis-aligned rectangle enclosing  $k$  points of  $P$ , under the restriction that the bottom edge lies on the  $x$ -axis.*

**Proof.** Divide the plane into  $n/q$  horizontal slabs each containing  $q$  points, for some parameter  $q$  to be set later.

Take such a slab  $\sigma$ . We solve the subproblem of finding a smallest  $k$ -enclosing axis-aligned rectangle under the restriction that the top edge is in  $\sigma$  and the bottom edge is on the  $x$ -axis. To this end, we first delete all points above  $\sigma$  or below the  $x$ -axis. We build the 1D data structure  $\mathcal{S}$  in the lemma for the  $x$ -coordinates of the surviving points, where the marked points are the  $q$  points in  $\sigma$ . The preprocessing time is  $O((n/q)T_{\text{convol}}(q) + q^3)$ . Then for each point  $p \in \sigma$ , we can compute a smallest  $k$ -enclosing axis-aligned rectangle where the top edge has  $p$ 's  $y$ -coordinate and bottom edge is on the  $x$ -axis, by making a copy of  $\mathcal{S}$ , deleting all points in  $\sigma$  above  $p$ , and querying  $\mathcal{S}$ . The time needed for the  $O(q)$  deletions, and for copying  $\mathcal{S}$ , is  $O(q^2)$ . The total time over all  $p \in \sigma$  is  $O(q^3)$ .

We return the minimum (by area or perimeter) of all the rectangles found. The overall running time over all  $n/q$  slabs  $\sigma$  is

$$O((n/q) \cdot ((n/q)T_{\text{convol}}(q) + q^3)).$$

With  $T_{\text{convol}}(q) = O(q^2/2^{\Omega(\sqrt{\log q})})$ , we can set  $q = n^{1/3}$ , for example, and obtain the final time bound  $O(n^2/2^{\Omega(\sqrt{\log q})})$ . ◀

For  $k$ -sensitive bounds, we can apply the shallow cutting technique from Section 2.2 (which is easier for 3-sided rectangles) and obtain an  $O(n \log n + nk/2^{\Omega(\sqrt{\log k})})$  time bound.

### 3.2 Arbitrarily oriented smallest $k$ -enclosing rectangle

We briefly consider the problem of computing a smallest-area/perimeter arbitrarily oriented rectangle (not necessarily axis-aligned) enclosing  $k$  points. The optimal rectangle is defined by 5 points, with one edge containing 2 points  $p_1^*$  and  $p_2^*$ . Given a fixed choice of  $p_1^*$  and  $p_2^*$ , we can use a rotation and translation to make  $p_1^*p_2^*$  lie on the  $x$ -axis and thereby obtain a 3-sided axis-aligned rectangle problem, which can be solved in  $O(n \log n + nk/2^{\Omega(\sqrt{\log k})})$  time. Exhaustively trying all pairs  $p_1^*p_2^*$  then gives  $O(n^3 \log n + n^2k/2^{\Omega(\sqrt{\log k})})$  total time.



### 3.3 Minimum-weight $k$ -enclosing rectangle

Our  $O(n^2 \log n)$ -time algorithm can be adapted to solve the following related problem. (Without the  $k$  constraint, the problem has an  $O(n^2)$ -time algorithm [5].)

► **Theorem 17.** *Given a set  $P$  of  $n$  points in the plane each with a real weight, and an integer  $k$ , one can compute, in  $O(n^2 \log n)$  time, the axis-aligned rectangle enclosing  $k$  points minimizing the total weight of the points inside.*

**Proof.** We follow the same approach as in Section 2.1, with the following differences in the data structure of Lemma 1. For every fragment we maintain the minimum weight solution. Using prefix sums, the entry  $M_{i,j}$  in the matrix contains the total weight of the elements from  $i$  to  $j$ . As before, we break the  $q + 1$  diagonals of entry into fragments, where each fragment summary maintains the minimum weight encountered.

A deletion of a marked point  $p$  of weight  $w$  would result in an insertion of a fixup entry, of value  $-w$  into a linked list of a diagonal where  $p$  appeared as a singleton (when crossing a column of  $p$ ), and a fixup entry of value  $+w$  when encountering the row column of  $p$ . The real value of a fragment is the value stored in the fragment plus the total sum of the fixups appearing before it in the linked list of its diagonal. As such, during query the real value can be computed in  $O(q)$  time overall, as this list is being scanned. When we merge two adjacent fragments separated by a singleton, we should increase the later fragment by the fixup value at the singleton before taking the minimum. Clearly, all the operations can be implemented in  $O(q)$  time.

Now, we can use the divide-and-conquer algorithm in the proof of Theorem 2 with no change. ◀

As an application, we can solve the following problem: given  $n$  points in the plane each colored red or blue, and an integer  $k$ , find an axis-aligned rectangle enclosing exactly  $k$  points minimizing the number of red points inside. This is a special case of the problem in the above theorem, where the red points have weight 1 and blue points have weight 0, and can thus be solved in  $O(n^2 \log n)$  time.

Similarly, we can solve for other variants of the red/blue problem, for example, finding a  $k$ -enclosing rectangle maximizing (or minimizing) the number of red points, or finding a  $k$ -enclosing rectangle with exactly a given number  $k_r$  of red points. (For the latter, the following observation allows us to reduce the 1D subproblem to querying for the maximum and minimum: given a set  $P$  of red/blue points in 1D and a value  $k$ , let  $K_r$  denote the set of all possible values  $k_r$  for which there exists an interval containing  $k$  points of  $P$  and exactly  $k_r$  red points; then  $K_r$  forms a contiguous range of integers, and thus contains all numbers between  $\min(K_r)$  and  $\max(K_r)$ .)

### 3.4 Subset sum for $k$ -enclosing rectangle

A more challenging variant of the weighted problem is to find a rectangle enclosing exactly  $k$  points with total weight exactly  $W$  (similar to subset sum), or more generally, find an axis-aligned rectangle enclosing exactly  $k$  points with total weight closest to  $W$ .

We use a different approach, using a 1D data structure that is static but can “plan for” a small number of deletions.

► **Lemma 18.** *Given a set  $P$  of  $n$  points in 1D and integers  $k$  and  $q$ , we can build a static data structure, with  $O(nq \log n)$  preprocessing time, that supports the following type of queries in  $O(q \log n)$  time: for any subset  $D \subset P$  of at most  $q$  points and any weight  $W$ , find an interval containing  $k$  points of  $P - D$  with weight closest to  $W$ .*

**Proof.** See full version [10]. ◀

► **Theorem 19.** *Given  $n$  points in the plane each with a real weight, and given a real number  $W$  and an integer  $k$ , one can compute, in  $O(n^{5/2} \log n)$  time, an axis-aligned rectangle enclosing exactly  $k$  points with total weight closest to  $W$ .*

**Proof.** See full version [10]. ◀

We can further improve the running time for small  $k$ :

► **Theorem 20.** *Given  $n$  points in the plane each with a real weight, and given a real number  $W$  and an integer  $k$ , one can compute, in  $O(n^2 \sqrt{k} \log k)$  time, an axis-aligned rectangle enclosing exactly  $k$  points with total weight closest to  $W$ .*

**Proof.** See full version [10]. ◀

As an application, we can solve the following problem: given  $n$  colored points in the plane with  $d$  different colors, and integers  $k_1, \dots, k_d$ , with  $k_1 + \dots + k_d = k$ , find an axis-aligned rectangle enclosing exactly  $k_i$  points of the  $i$ th color. The problem was proposed by Barba et al. [4], who gave an  $O(n^2 k)$ -time algorithm. (It may be viewed as a geometric variant of the jumbled or histogram indexing problem for strings [11].) It is a special case of the problem from Theorem 19: we can give points with color  $i$  a weight of  $M^i$  for a sufficiently large  $M$ , e.g.,  $M = n + 1$ , and set the target to  $W = \sum_{i=1}^d k_i M^i$ . Since weights require  $O(d \log n)$  bits, each addition has  $O(d)$  cost, and so the running time becomes  $O(dn^2 \sqrt{k} \log k)$ . The weights can be reduced to  $O(\log n)$  bits by randomized hashing (for example, by randomly selecting  $M$  from  $\{0, \dots, p-1\}$  and working with numbers modulo  $p$  for an  $O(\log n)$ -bit prime  $p$ ), since there are only polynomially (i.e.,  $O(n^4)$ ) many combinatorially different rectangles. This way, the running time can be reduced to  $O(n^2 \sqrt{k} \log k)$  – this improves Barba et al.’s result.

### 3.5 Conditional lower bounds

We can prove that the smallest-perimeter  $k$ -enclosing axis-aligned rectangle problem do not have truly subquadratic (i.e.,  $O(n^{2-\delta})$ ) algorithms, under the conjecture that  $(\min, +)$ -convolution does not have a truly subquadratic algorithm. Our proof holds for the 3-sided version of the problem, which complements nicely with our upper bound in Section 3.1 using  $(\min, +)$ -convolution.

We describe a reduction from the following decision problem, which Cygan et al. [14] showed does not have a truly subquadratic algorithm under the  $(\min, +)$ -convolution conjecture.

► **Problem 21.  $(\min, +)$ -Convolution Decision.** Given real numbers  $a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}$ , and  $c_0, \dots, c_{n-1}$ , decide whether

$$\forall \ell : c_\ell \leq \min_{i+j=\ell} (a_i + b_j).$$

► **Theorem 22.** *If there is a  $T(n)$ -time algorithm for computing the smallest-perimeter/area axis-aligned rectangle enclosing  $k$  points for a given set of  $n$  points in the plane and a given number  $k$  (with or without the constraint that the bottom edge lies on the  $x$ -axis), then there is an  $O(T(O(n)))$ -time algorithm for Problem 21.*

**Proof.** See full version [10]. ◀

A similar reduction holds for the minimum-weight  $k$ -enclosing rectangle problem from Theorem 17:

► **Theorem 23.** *If there is a  $T(n)$ -time algorithm for computing the minimum-weight axis-aligned rectangle enclosing  $k$  points for a given set of  $n$  weighted points in the plane and number  $k$  (with or without the constraint that the bottom edge lies on the  $x$ -axis), then there is an  $O(T(O(n)))$ -time algorithm for Problem 21.*

**Proof.** See full version [10]. ◀

A near-quadratic conditional lower bound for the minimum-weight rectangle problem without the  $k$  constraint was given by Backurs et al. [3] (under a different “popular” conjecture about the complexity of maximum-weight clique).

We can similarly prove that the subset-sum variant of the  $k$ -enclosing rectangle problem from Theorem 19 (or its 3-sided variant) does not have truly subquadratic algorithms, under the conjecture that the *convolution-3SUM* problem (given real numbers  $a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}, c_0, \dots, c_{n-1}$ , decide whether  $c_\ell = a_i + b_{\ell-i}$  for some  $i$  and  $\ell$ ) does not have a truly subquadratic algorithm (which is known to be true under the conjecture that 3SUM for integers does not have a truly subquadratic algorithm [25]).

---

## References

- 1 Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri. Finding  $k$  points with minimum diameter and related problems. *J. Algorithms*, 12(1):38–56, 1991. doi:10.1016/0196-6774(91)90022-Q.
- 2 Rossen Atanassov, Prosenjit Bose, Mathieu Couture, Anil Maheshwari, Pat Morin, Michel Paquette, Michiel H. M. Smid, and Stefanie Wuhrer. Algorithms for optimal outlier removal. *J. Discrete Algorithms*, 7(2):239–248, 2009. doi:10.1016/j.jda.2008.12.002.
- 3 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. In *Proc. 43rd Int. Colloq. Automata Lang. Prog. (ICALP)*, pages 81:1–81:13, 2016.
- 4 Luis Barba, Stephane Durocher, Robert Fraser, Ferran Hurtado, Saeed Mehrabi, Debajyoti Mondal, Jason Morrison, Matthew Skala, and Mohammad Abdul Wahid. On  $k$ -enclosing objects in a coloured point set. In *Proc. 25th Canad. Conf. Comput. Geom. (CCCG)*, 2013. URL: [http://cccg.ca/proceedings/2013/papers/paper\\_35.pdf](http://cccg.ca/proceedings/2013/papers/paper_35.pdf).
- 5 Jérémy Barbay, Timothy M. Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in  $O(n^2)$  time (and better). *Inform. Process. Lett.*, 114(8):437–445, 2014. doi:10.1016/j.ipl.2014.03.007.
- 6 Marshall W. Bern. Approximate closest-point queries in high dimensions. *Inf. Process. Lett.*, 45(2):95–99, 1993. doi:10.1016/0020-0190(93)90222-U.
- 7 David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Pătraşcu, and Perouz Taslakian. Necklaces, convolutions, and  $X + Y$ . *Algorithmica*, 69(2):294–314, 2014. doi:10.1007/s00453-012-9734-3.
- 8 Timothy M. Chan. Approximate nearest neighbor queries revisited. *Discrete & Computational Geometry*, 20(3):359–373, 1998. doi:10.1007/PL00009390.
- 9 Timothy M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22(4):547–567, 1999. doi:10.1007/PL00009478.
- 10 Timothy M. Chan and Sarel Har-Peled. Smallest  $k$ -enclosing rectangle revisited. *CoRR*, abs/1903.06785, 2019. arXiv:1903.06785.
- 11 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. 47th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 31–40, 2015. doi:10.1145/2746539.2746568.
- 12 Timothy M. Chan and Mihai Pătraşcu. Counting inversions, offline orthogonal range counting, and related problems. In *Proc. 21st ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 161–173, 2010. doi:10.1137/1.9781611973075.15.

- 13 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov–Smolensky. In *Proc. 27th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.
- 14 Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Włodarczyk. On problems equivalent to  $(\min, +)$ -convolution. In *Proc. 44th Int. Colloq. Automata Lang. Prog. (ICALP)*, volume 80 of *LIPICs*, pages 22:1–22:15, 2017. doi:10.4230/LIPICs.ICALP.2017.22.
- 15 Sandip Das, Partha P. Goswami, and Subhas C. Nandy. Smallest  $k$ -point enclosing rectangle and square of arbitrary orientation. *Inform. Process. Lett.*, 94(6):259–266, 2005. doi:10.1016/j.ipl.2005.02.013.
- 16 Amitava Datta, Hans-Peter Lenhof, Christian Schwarz, and Michiel H. M. Smid. Static and dynamic algorithms for  $k$ -point clustering problems. *J. Algorithms*, 19(3):474–503, 1995. doi:10.1006/jagm.1995.1048.
- 17 Mark de Berg, Sergio Cabello, Otfried Cheong, David Eppstein, and Christian Knauer. Covering many points with a small-area box. *CoRR*, abs/1612.02149, 2016. arXiv:1612.02149.
- 18 David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete Comput. Geom.*, 11:321–350, 1994. URL: <http://jeffe.cs.illinois.edu/pubs/small.html>.
- 19 Harold N. Gabow and Robert Endre Tarjan. A linear-time algorithm for a special case of disjoint set union. *J. Comput. Sys. Sci.*, 30(2):209–221, 1985. doi:10.1016/0022-0000(85)90014-5.
- 20 Sariel Har-Peled and Soham Mazumdar. Fast algorithms for computing the smallest  $k$ -enclosing disc. *Algorithmica*, 41(3):147–157, 2005. URL: [https://sarielhp.org/p/03/min\\_disk/](https://sarielhp.org/p/03/min_disk/).
- 21 Sariel Har-Peled and Micha Sharir. Relative  $(p, \varepsilon)$ -approximations in geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011. doi:10.1007/s00454-010-9248-1.
- 22 Allan Grønlund Jørgensen and Kasper Green Larsen. Range selection and median: Tight cell probe lower bounds and adaptive data structures. In *Proc. 22nd ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 805–813, 2011. doi:10.1137/1.9781611973082.63.
- 23 Haim Kaplan, Sasanka Roy, and Micha Sharir. Finding axis-parallel rectangles of fixed perimeter or area containing the largest number of points. In *Proc. 26th Annu. Euro. Sympos. Alg. (ESA)*, volume 87 of *LIPICs*, pages 52:1–52:13, 2017. doi:10.4230/LIPICs.ESA.2017.52.
- 24 Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.
- 25 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd Annu. ACM Sympos. Theory Comput. (STOC)*, pages 603–610, 2010. doi:10.1145/1806689.1806772.
- 26 Michael Segal and Klara Kedem. Enclosing  $k$  points in the smallest axis parallel rectangle. *Inform. Process. Lett.*, 65(2):95–99, 1998. doi:10.1016/S0020-0190(97)00212-3.
- 27 Robert Endre Tarjan. Efficiency of a good but not linear set union algorithm. *J. Assoc. Comput. Mach.*, 22(2):215–225, 1975. doi:10.1145/321879.321884.
- 28 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proc. 46th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.



# Dynamic Geometric Data Structures via Shallow Cuttings

Timothy M. Chan

Department of Computer Science, University of Illinois at Urbana-Champaign, USA  
tmc@illinois.edu

---

## Abstract

We present new results on a number of fundamental problems about dynamic geometric data structures:

1. We describe the first fully dynamic data structures with sublinear amortized update time for maintaining (i) the number of vertices or the volume of the convex hull of a 3D point set, (ii) the largest empty circle for a 2D point set, (iii) the Hausdorff distance between two 2D point sets, (iv) the discrete 1-center of a 2D point set, (v) the number of maximal (i.e., skyline) points in a 3D point set. The update times are near  $n^{11/12}$  for (i) and (ii),  $n^{7/8}$  for (iii) and (iv), and  $n^{2/3}$  for (v). Previously, sublinear bounds were known only for restricted “semi-online” settings [Chan, SODA 2002].
2. We slightly improve previous fully dynamic data structures for answering extreme point queries for the convex hull of a 3D point set and nearest neighbor search for a 2D point set. The query time is  $O(\log^2 n)$ , and the amortized update time is  $O(\log^4 n)$  instead of  $O(\log^5 n)$  [Chan, SODA 2006; Kaplan et al., SODA 2017].
3. We also improve previous fully dynamic data structures for maintaining the bichromatic closest pair between two 2D point sets and the diameter of a 2D point set. The amortized update time is  $O(\log^4 n)$  instead of  $O(\log^7 n)$  [Eppstein 1995; Chan, SODA 2006; Kaplan et al., SODA 2017].

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

**Keywords and phrases** dynamic data structures, convex hulls, nearest neighbor search, closest pair, shallow cuttings

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.24

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.08387>.

**Funding** *Timothy M. Chan*: Work supported in part by NSF Grant CCF-1814026.

**Acknowledgements** I thank Sariel Har-Peled for discussions on other problems that indirectly led to the results of this paper. Thanks also to Mitchell Jones for discussions on range searching for points in convex position.

## 1 Introduction

**Background.** *Dynamic* data structures that can support insertions and deletions of data have been a fundamental topic in computational geometry since the beginning of the field. For example, in 1981 an early landmark paper by Overmars and van Leeuwen [25] presented a fully dynamic data structure for *2D convex hulls* with  $O(\log n)$  query time and  $O(\log^2 n)$  update time; the  $\log^2 n$  bound was later improved in a series of work [7, 6, 12] for various basic types of hull queries, e.g., finding extreme points along given directions.

One of the key results in the area is the author’s fully dynamic data structure for *3D convex hulls* [10], which was the first to achieve polylogarithmic query and update time for basic types of hull queries. The original solution required  $O(\log^2 n)$  query time for extreme point queries, and  $O(\log^6 n)$  amortized update time. (A previous solution by Agarwal and



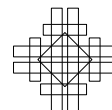
© Timothy M. Chan;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 24; pp. 24:1–24:13

Leibniz International Proceedings in Informatics



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



Matoušek [4] had  $O(n^\varepsilon)$  query or update time for an arbitrarily small constant  $\varepsilon > 0$ .) Recently Kaplan et al. [21] noted a small modification of the data structure, improving the update time to  $O(\log^5 n)$ . The result has numerous applications, including dynamic 2D nearest or farthest neighbor search (by the standard lifting map). Another application is dynamic 2D bichromatic closest pair (i.e., computing  $\min_{p \in P} \min_{q \in Q} \|p - q\|$  for two planar point sets  $P$  and  $Q$ ) or dynamic 2D diameter (i.e., computing  $\max_{p \in P} \max_{q \in P} \|p - q\|$  for a planar point set  $P$ ): Eppstein [18] gave a clever, general technique reducing dynamic closest/farthest pair problems to dynamic nearest/farthest neighbor search, which increased the update time by a  $\log^2 n$  factor; when combined with the above, this yielded an  $O(\log^7 n)$  update time bound.

For many other problems, polylogarithmic update time appears more difficult, and getting sublinear update time is already challenging. For example, in SoCG 2001, the author [8] obtained a dynamic data structure for the *width* of a 2D point set with  $O^*(\sqrt{n})$  amortized update time.<sup>1</sup> (Part of the difficulty is that the width problem is neither “decomposable” nor “LP-type”.) Sublinear update time is known for a few other assorted geometric problems, such as *dynamic connectivity* for the intersection graph of geometric objects [14].

In SODA 2002, the author [9] explored still more challenging dynamic geometric problems, including maintaining

- (i) the number of vertices and facets of a 3D convex hull, or its volume,
- (ii) the *largest empty circle* for a 2D point set (with center restricted to be inside a fixed triangle),
- (iii) the *Hausdorff distance* for 2D point sets  $P$  and  $Q$  (i.e., computing  $\max_{q \in Q} \min_{p \in P} \|p - q\|$  for two planar point set), and
- (iv) the *discrete 1-center* of a 2D point set  $P$  (i.e., computing  $\min_{q \in P} \max_{p \in P} \|p - q\|$ ).

The paper [9] obtained sublinear results only for the *insertion-only* case and the *off-line* case (where we are given the entire update sequence in advance), or a generalization of both – the *semi-online* case (as defined by Dobkin and Suri [17], where we are given the deletion time of an element when it is inserted). The update time bounds were  $O^*(n^{7/8})$  for (i) and (ii), and  $O^*(n^{5/6})$  for (iii) and (iv).

None of these four problems are “decomposable”. In particular, problem (i) is nontrivial since known methods such as [10] for 3D convex hull queries do not maintain the global hull explicitly, unlike Overmars and van Leeuwen’s original data structure for 2D convex hulls. Problem (ii) also seems to require explicit maintenance of a 3D convex hull (lifted from the 2D farthest-point Voronoi diagram). Problems (iii) and (iv) are max-min or min-max problems, and lack the symmetry of min-min and max-max problems that enable Eppstein’s technique. For all these problems, the fully dynamic case has remained open.

### New results.

1. We present the first fully dynamic data structures with sublinear update time for Problems (i)–(iv). The amortized update time bounds are  $O^*(n^{11/12})$  for (i) and (ii), and  $O^*(n^{5/6})$  for (iii) and (iv).

The approach is general enough to be applicable to many more problems; for example, we can maintain the number of maximal or “skyline” points (points that are not dominated by other points) in a 3D point set in  $O^*(n^{2/3})$  amortized time.

<sup>1</sup> Throughout the paper, we use the  $O^*$  notation to hide small extra factors that are polylogarithmic, or in some cases,  $o(n^\varepsilon)$  for an arbitrarily small constant  $\varepsilon > 0$ .

2. For basic 3D convex hull queries (e.g., extreme point queries) and 2D nearest neighbor search, as mentioned, Kaplan et al. [21] have lowered the amortized update time of the author's fully dynamic data structure [10], from  $O(\log^6 n)$  to  $O(\log^5 n)$ . We describe a further logarithmic-factor improvement, from  $O(\log^5 n)$  to  $O(\log^4 n)$ .

Although this improvement is admittedly small, the importance of the result stems from its many applications [10]; for example, we can now compute the *convex (or onion) layers* of a 3D point set in  $O(n \log^4 n)$  time, and the *k-level* in an arrangement of planes in 3D in  $O(n \log n + f \log^4 n)$  time where  $f$  is the output size.

3. For bichromatic closest pair and diameter in 2D, combining Eppstein's technique [18] with the above new result on dynamic nearest neighbor search already gives a slightly improved amortized update time of  $O(\log^6 n)$ . We describe a further, more substantial improvement that eliminates the two extra logarithmic factors caused by Eppstein's technique [18]. The new update time bound is  $O(\log^4 n)$ .

Dynamic bichromatic closest pair has applications to other problems. For example, we can now maintain the Euclidean minimum spanning tree of a 2D point set with  $O(\log^6 n)$  amortized update time by using another reduction of Eppstein [18] combined with known results for dynamic minimum spanning trees for graphs [20].

**Techniques.** The common thread in all of our new methods is the use of *shallow cuttings*: Let  $H$  be a set of  $n$  hyperplanes in  $\mathbb{R}^d$ . The *level* of a point  $q$  refers to the number of hyperplanes of  $H$  strictly below  $q$ . A  $(k, K)$ -*shallow cutting* is a collection of cells covering all points of level at most  $k$ , such that each cell intersects at most  $K$  hyperplanes. The *conflict list*  $H_\Delta$  of a cell  $\Delta$  refers to the subset of all hyperplanes of  $H$  intersecting  $\Delta$ .

Matoušek [23] proved the existence of shallow cuttings with small number of cells. Specifically, in 3D, the main lemma can be stated as follows:<sup>2</sup>

► **Lemma 1.** (Shallow Cutting Lemma) *Given a set  $H$  of  $n$  planes in  $\mathbb{R}^3$  and a parameter  $k \in [1, n]$ , there exists a  $(k, O(k))$ -shallow cutting with  $O(n/k)$  cells, where each cell is a "downward" tetrahedron containing  $(0, 0, -\infty)$ . The cutting, together with the conflict lists of all its cells, can be constructed in  $O(n \log n)$  time.*

The construction time was first shown by Ramos [26] with a randomized algorithm. Later, Chan and Tsakalidis [15] obtained the first  $O(n \log n)$ -time deterministic algorithm.

To see how static shallow cuttings may be useful for dynamic geometric data structures, observe that most of the problems considered here are related to the lower envelope of a dynamic set of planes in  $\mathbb{R}^3$  (via duality or the standard lifting transformation). Usually, the bottleneck lies in deletions rather than insertions. Basically, a shallow cutting provides a compact implicit representation of the  $(\leq k)$ -level, which is guaranteed to cover the lower envelope even when up to  $k$  deletions have occurred.

A further idea behind all our solutions is to classify planes into two types, those that intersect few cells of the shallow cutting, and those that intersect many cells. The latter type of planes may be bad in slowing down updates, but the key observation is that there can't be too many bad elements.

The new sublinear solutions to Problems (i)–(iv), described in Sections 2–3, are obtained by incorporating the shallow cutting idea with the previous techniques from [9], based on periodic rebuilding. The entire solution is conceptually not complicated at all, and the

<sup>2</sup> Matoušek's original formulation in  $\mathbb{R}^d$  states the existence of a  $(k, n/r)$ -shallow cutting with  $O(r^{\lfloor d/2 \rfloor} (1 + kr/n)^{\lceil d/2 \rceil})$  cells.



description for Problem (i) fits in under two pages, assuming the availability of known range searching structures. As are typical in other works on data structures with sublinear update time with “funny” exponents, parameters are judiciously chosen to balance several competing costs.

The shallow cutting idea has actually been exploited before in dynamic data structures for basic 3D convex hull queries: Agarwal and Matoušek [4] used shallow cuttings recursively (which caused some loss of efficiency), while the author [10] used a hierarchy of shallow cuttings, for logarithmically many values of  $k$ . The above application of shallow cuttings to Problems (i)–(iv) is even more elementary – we only need a single cutting. (This makes it all the more embarrassing that the idea was missed till now.)

For basic 3D convex hull queries and 2D nearest neighbor search, our improvement is less innovative. Described in Section 4 (which can be read independently of the previous sections), it is based on the author’s original data structure [10], with Kaplan et al.’s logarithmic-factor improvement [21], plus one extra idea to remove a second logarithmic factor: the main observation is that Chan and Tsakalidis’s algorithm for shallow cuttings [15] already constructs an entire hierarchy of  $O(\log n)$  cuttings in  $O(n \log n)$  time, not just a single cutting. However, the hierarchy needed for the data structure in [10] requires some planes be pruned as we go from one cutting to the next, so Chan and Tsakalidis’s algorithm cannot be applied immediately. Still, we show that some nontrivial but technical changes (as explained in the appendix) can fix the problem.

For 2D bichromatic closest pair and diameter, our  $\log^2 n$ -factor improvement, described in Section 5, is a bit more interesting. We still do not know how to improve Eppstein’s general reduction [18] from dynamic closest pair to dynamic nearest neighbor search, but intuitively the blind combination of Eppstein’s technique with the author’s dynamic data structure for 2D nearest neighbor search seems wasteful, since both share some commonalities (both are sophisticated variants of the *logarithmic method* [5], and both handle deletions via re-insertions of elements into smaller subsets). To avoid the redundancy, we show how to directly modify our dynamic data structure for 2D nearest neighbor search to solve the dynamic 2D bichromatic closest pair problem. The resulting modification completely bypasses Eppstein’s “conga line” structure [18, 19], and turns out to cause no increase to the  $O(\log^4 n)$  bound.

## 2 Dynamic 3D Convex Hull Size

We begin with our new sublinear-time fully dynamic data structure for maintaining the number of vertices/facets of the convex hull of a dynamic 3D point set. The solution is based on the use of shallow cuttings (Lemma 1) and the author’s previous semi-online data structure [9].

► **Theorem 2.** *We can maintain the number of vertices, edges, and facets for the convex hull of a dynamic set of  $n$  points in  $\mathbb{R}^3$ , in general position, with  $O^*(n)$  preprocessing time and  $O^*(n^{11/12})$  amortized insertion and deletion time.*

**Proof.** It suffices to maintain the number of convex hull facets, which determines the number of vertices and edges (assuming general position). It suffices to compute the number of upper hull facets, since by symmetry we can compute the number of lower hull facets. We describe our solution in dual space, where the problem is to compute the number of vertices in  $\text{LE}(H)$  for a dynamic set  $H$  of  $n$  planes in  $\mathbb{R}^3$ .

Let  $k$  and  $s$  be parameters to be set later. We divide the update sequence into phases of  $k$  updates each. We maintain a decomposition of the set  $H$  into a deletion-only set  $H_0$  and a small set  $H_{\text{bad}}$  of “bad” planes.

**Preprocessing for each phase.** At the beginning of each phase, we construct a  $(k, O(k))$ -shallow cutting  $\Gamma$  of  $H$  with  $O(n/k)$  cells, together with all their conflict lists, by Lemma 1. We set

$$H_0 = \{h \in H : h \text{ intersects at most } n/s \text{ cells}\} \quad \text{and} \quad H_{\text{bad}} = H - H_0.$$

Since the total conflict list size is  $O(n/k \cdot k) = O(n)$ , we have  $|H_{\text{bad}}| = O(s)$ .

Let  $V_0$  and  $E_0$  be the set of vertices and edges of the portion of  $\text{LE}(H_0)$  covered by  $\Gamma$ , respectively. There are  $O(k)$  such vertices and edges per cell of  $\Gamma$ , and hence,  $|V_0|, |E_0| = O(n/k \cdot k) = O(n)$ . We preprocess  $V_0$  and  $E_0$  in  $O^*(n)$  time by known range searching and intersection searching techniques, so that

- we can count the number of points in  $V_0$  inside a query tetrahedron in  $O^*(n^{2/3})$  time (this is 3D simplex range searching) [22, 11, 2];
- we can count the number of line segments in  $E_0$  intersecting a query triangle in  $O^*(n^{3/4})$  time (as noted in [9], we can first solve the case of lines and query halfplanes in  $\mathbb{R}^3$  using semialgebraic range searching [3] in Plücker space, and then extend the solution for line segments and query triangles by a multi-level data structure [2]).

These data structures can support insertions and deletions of points in  $V_0$  and line segments in  $E_0$  in  $O^*(1)$  time each. In addition, we preprocess  $H_0$  in a known dynamic lower envelope data structure in  $O^*(n)$  time, to support ray shooting queries in  $\text{LE}(H_0)$  in  $O^*(1)$  time and deletions in  $O^*(1)$  time (e.g., see [4] or Section 4). The total preprocessing time per phase is  $O^*(n)$ . Amortized over  $k$  updates, the cost is  $O^*(n/k)$ .

**Inserting a plane  $h$ .** We just insert  $h$  to the list  $H_{\text{bad}}$ . Note that  $|H_{\text{bad}}| = O(s + k)$  at all times, since there are at most  $k$  insertions per phase.

**Deleting a plane  $h$  from  $H_{\text{bad}}$ .** We just remove  $h$  from the list  $H_{\text{bad}}$ .

**Deleting a plane  $h$  from  $H_0$ .** We consider each cell  $\Delta \in \Gamma$  intersected by  $h$ , and compute  $\text{LE}((H_0)_\Delta)$  from scratch in  $O(k \log k)$  time (since  $|(H_0)_\Delta| = O(k)$ ). As the number of cells intersected by  $h$  is at most  $n/s$ , this computation requires  $O^*(kn/s)$  total time. The sets  $V_0$  and  $E_0$  undergo at most  $O(kn/s)$  changes, and their associated data structures can be updated in  $O^*(kn/s)$  time.

**Computing the answer.** To compute the number of vertices of  $\text{LE}(H) = \text{LE}(H_0 \cup H_{\text{bad}})$ , we first construct  $\text{LE}(H_{\text{bad}})$  in  $O((s + k) \log(s + k))$  time, and triangulate all its  $O(s + k)$  faces. For each triangle  $\tau$  in this triangulation:

- we count the number of vertices of  $V_0$  that lie directly below  $\tau$ , in  $O^*(n^{2/3})$  time; and
- we count the number of edges of  $E_0$  that intersect  $\tau$ , in  $O^*(n^{3/4})$  time.

We sum up all these counts. In addition, for each edge of  $\text{LE}(H_{\text{bad}})$ , we test whether it intersects  $\text{LE}(H_0)$  by ray shooting in  $O^*(1)$  time, and increment the count if true. For each vertex of  $\text{LE}(H_{\text{bad}})$ , we test whether it is underneath  $\text{LE}(H_0)$  by vertical ray shooting in  $O^*(1)$  time, and increment the count if true. Note that  $\text{LE}(H)$  is covered by  $\Gamma$  at all times, since there are at most  $k$  deletions per phase. The overall count thus gives the answer. The total time to compute the answer is  $O^*((s + k)n^{3/4})$ .

**Analysis.** The overall amortized update time is

$$O^*(n/k + kn/s + (s+k)n^{3/4}).$$

The theorem follows by setting  $s = k^2$  and  $k = n^{1/12}$ .  $\blacktriangleleft$

The preprocessing time can be made  $O(n \log n)$  and space made  $O(n)$  by increasing the update time by an  $n^\varepsilon$  factor, via known trade-offs for range/intersection searching (with larger-degree partition trees). The method can be deamortized, using existing techniques [24].

The same method can be adapted to maintain the sum or maximum of  $f(v)$  over all vertices  $v$  of  $\text{LE}(H)$ , for a general class of functions  $f$ . Instead of range counting, we store the set  $V_0$  of points for range sum or range maximum queries (which have similar complexity as range counting). For the set  $E_0$  of line segments, the base level of its multi-level data structure requires data structures  $\mathcal{S}_L$  for each canonical subset  $L$  of lines in  $\mathbb{R}^3$ , so that we can return the sum or maximum of  $f(\ell \cap h)$  over all  $\ell \in L$  for a query plane  $h$  in  $O^*(|L|^\alpha)$  time, supporting insertions and deletions in  $L$  in  $O^*(1)$  time. If  $\alpha \leq 3/4$ , the final time bound of our algorithm remains  $O^*(n^{11/12})$ .

► **Theorem 3.** *We can maintain the volume of the convex hull for a dynamic set of  $n$  points in  $\mathbb{R}^3$ , with  $O^*(n)$  preprocessing time and  $O^*(n^{11/12})$  amortized insertion and deletion time.*

**Proof.** Let  $o$  be a fixed point sufficiently far below all the input points. It suffices to maintain the sum of the volume of the tetrahedra  $op_1p_2p_3$  over all upper hull facets  $p_1p_2p_3$ , since by symmetry we can maintain a similar sum for lower hull facets and subtract. We map each point  $p$  to its dual plane  $h_p$ . Then the problem fits in the above framework, with  $f(v)$  equal to the volume of the tetrahedron  $op_1p_2p_3$  for a vertex  $v$  defined by the planes  $h_{p_1}, h_{p_2}, h_{p_3}$ . For a fixed line  $\ell$  defined by the planes  $h_{p_1}$  and  $h_{p_2}$ , observe that  $f(\ell \cap h_p)$  is a linear function over the 3 coordinates of  $p$ , since the volume of  $op_1p_2p$  can be expressed as a determinant. (This assumes that  $op_1p_2$  is oriented clockwise, which we can ensure at the base level of the multi-level data structure.) Thus, we can implement the base structures  $\mathcal{S}_L$  with  $\alpha = 0$ , by simply summing the 4 coefficients of the associated linear functions over all  $\ell \in L$ .  $\blacktriangleleft$

► **Theorem 4.** *We can maintain the largest empty circle of a dynamic set of  $n$  points in  $\mathbb{R}^2$ , under the restriction that the center lies inside a given triangle  $\Delta_0$ , with  $O^*(n)$  preprocessing time and  $O^*(n^{11/12})$  amortized insertion and deletion time.*

**Proof.** By the standard lifting transformation, map each input point  $p = (a, b) \in \mathbb{R}^2$  to the plane  $h_p$  with equation  $z = -2ax - 2by + a^2 + b^2$  in  $\mathbb{R}^3$ . Add 3 near-vertical planes along the edges of  $\Delta_0$ . The largest empty circle problem reduces to finding a vertex  $v = (x, y, z)$  of the lower envelope of these planes, maximizing  $f(v) = x^2 + y^2 + z$ . For a fixed line  $\ell$ , observe that  $f(\ell \cap h_{(a,b)})$  is a fixed-degree rational function (ratio of two polynomials) in the 2 variables  $a$  and  $b$ . We can implement the base structures  $\mathcal{S}_L$  with  $\alpha = 2/3$ , by known techniques for semialgebraic range searching in 3D [4] (applied to the graphs of these bivariate functions).  $\blacktriangleleft$

We can obtain sublinear update time bounds for other similar problems, e.g., maintaining the minimum/maximum-area Delaunay triangle of a dynamic 2D point set. Another application is computing the number of maximal points, also called “skyline points” (which are points not dominated by other points), in a dynamic 3D point set:

► **Theorem 5.** *We can maintain the number of maximal points in a dynamic set  $P$  of  $n$  points in  $\mathbb{R}^3$ , with  $O^*(n)$  preprocessing time and  $O^*(n^{2/3})$  amortized insertion and deletion time.*

**Proof.** The maximal points are vertices of the upper envelope of orthants  $(-\infty, a] \times (-\infty, b] \times (-\infty, c]$  over all input points  $(a, b, c) \in P$  (the upper envelope is an orthogonal polyhedron). As is well known, an analogue of the shallow cutting lemma holds for such orthants in 3D (in fact, there is a transformation that maps such orthants to halfspaces in 3D); for example, see [13]. The same method can thus be adapted. In fact, it can be simplified. The data structure for  $V_0$  is for orthogonal range searching [16], which has  $O^*(1)$  query and update time. The data structure  $E_0$  is not needed. The overall update time becomes

$$O^*(n/k + kn/s + (s + k)).$$

The theorem follows by setting  $s = k^2$  and  $k = n^{1/3}$ . ◀

We can similarly maintain the volume of a union of  $n$  boxes in  $\mathbb{R}^3$  in the case when all the boxes have a common corner point at the origin (this is called the *hypervolume indicator* problem) with  $O^*(n^{2/3})$  update time (previously, an  $O^*(\sqrt{n})$  bound was known only in the semi-online setting [9]).

### 3 Dynamic 2D Hausdorff Distance

The method in Section 2 can also be adapted to solve the dynamic 2D Hausdorff distance problem:

► **Theorem 6.** *We can maintain the Hausdorff distance between two dynamic sets  $P$  and  $Q$  of at most  $n$  points in  $\mathbb{R}^2$ , with  $O^*(n)$  preprocessing time and  $O^*(n^{8/9})$  amortized insertion and deletion time.*

**Proof.** By the standard lifting transformation, map each point  $p = (a, b) \in P$  to the plane  $h_p$  with equation  $z = -2ax - 2by + a^2 + b^2$  in  $\mathbb{R}^3$ . Let  $H$  be the resulting set of planes. For each point  $q \in Q$ , let  $\lambda_H(q)$  denote the point on  $\text{LE}(H)$  at the vertical line at  $q$ . The problem is to find the maximum of  $f(\lambda_H(q))$  over all  $q \in Q$ , where  $f(x, y, z) = x^2 + y^2 + z$ , for a dynamic set  $H$  of at most  $n$  planes and a dynamic set  $Q$  of at most  $n$  points.

Let  $k$  and  $s$  be parameters to be set later. We divide the update sequence into phases of  $k$  updates each. We maintain a decomposition of the set  $H$  into a deletion-only set  $H_0$  and a small set  $H_{\text{bad}}$  of “bad” planes, and a decomposition of the set  $Q$  into a deletion-only set  $Q_0$  and a small set  $Q_{\text{bad}}$  of “bad” points.

**Preprocessing for each phase.** At the beginning of each phase, we construct a  $(k, O(k))$ -shallow cutting  $\Gamma$  of  $H$  with  $O(n/k)$  cells, together with all their conflict lists, by Lemma 1. We further subdivide the cells to ensure that each cell contains at most  $k$  points of  $Q$  in its  $xy$ -projection; this can be done by  $O(n/k)$  additional vertical plane cuts, so the number of cells remains  $O(n/k)$ . We set

$$H_0 = \{h \in H : h \text{ intersects at most } n/s \text{ cells}\} \quad \text{and} \quad H_{\text{bad}} = H - H_0.$$

Since the total conflict list size is  $O(n/k \cdot k) = O(n)$ , we have  $|H_{\text{bad}}| = O(s)$ .

We set  $Q_0 = Q$ . We compute  $\lambda_{H_0}(q)$  for all  $q \in Q$  in  $O(n \log n)$  time. Let  $\Lambda_0$  be the subset of points in  $\{\lambda_{H_0}(q) : q \in Q\}$  covered by  $\Gamma$ . We preprocess the point set  $\Lambda_0$  in known 3D simplex range searching data structures [22, 11, 2] in  $O^*(n)$  time, to support the following queries in  $O^*(n^{2/3})$  time:

- compute the maximum of  $f(v)$  over all points  $v \in \Lambda_0$  inside a query tetrahedron;

- compute the maximum of  $f(\lambda_{\{h_p\}}(x, y))$  over all points  $v = (x, y, z) \in \Lambda_0$  inside a query tetrahedron for a query plane  $h_p$ ; note that maximizing  $f(\lambda_{\{h_p\}}(x, y))$  is equivalent to maximizing the distance from  $(x, y)$  to  $p$  (so we can use a 2-level data structure, combining simplex range searching with 2D farthest neighbor searching).

The data structures can support insertions and deletions of points in  $\Lambda_0$  in  $O^*(1)$  time each. In addition, we preprocess  $H_0$  in a known dynamic lower envelope data structure in  $O^*(n)$  time, to support ray shooting queries in  $\text{LE}(H_0)$  in  $O^*(1)$  time and deletions in  $O^*(1)$  time (e.g., see [4] or Section 4).

**Inserting a plane  $h$  to  $H$  or a point  $q$  to  $Q$ .** We just insert  $h$  to the list  $H_{\text{bad}}$  or  $q$  to the list  $Q_{\text{bad}}$ . Note that  $|H_{\text{bad}}| = O(s + k)$  and  $|Q_{\text{bad}}| = O(k)$  at all times.

**Deleting a plane  $h$  from  $H_{\text{bad}}$  or a point  $q$  from  $Q_{\text{bad}}$ .** We just remove  $h$  from the list  $H_{\text{bad}}$  or  $q$  from the list  $Q_{\text{bad}}$ .

**Deleting a point  $q$  from  $Q_0$ .** We just remove  $\lambda_{H_0}(q)$  from the set  $\Lambda_0$  in  $O^*(1)$  time.

**Deleting a plane  $h$  from  $H_0$ .** We consider each cell  $\Delta \in \Gamma$  intersected by  $h$ , and compute  $\lambda_{(H_0)_\Delta}(q)$  for all  $q \in Q$  in the  $xy$ -projection of  $\Delta$  from scratch in  $O(k \log k)$  time (since  $\Delta$  is intersected by  $O(k)$  planes in  $H$  and contains  $O(k)$  points of  $Q$  in its  $xy$ -projection). As the number of cells intersected by  $h$  is at most  $n/s$ , this computation takes  $O^*(kn/s)$  total time. The set  $\Lambda_0$  undergoes at most  $O(kn/s)$  changes, and its associated data structures can be updated in  $O^*(kn/s)$  time.

**Computing the answer.** To compute the maximum of  $f(\lambda_H(q))$  over all  $q \in Q$ , we first construct  $\text{LE}(H_{\text{bad}})$  in  $O((s + k) \log(s + k))$  time, and triangulate all its  $O(s + k)$  faces. For each triangle  $\tau$  in this triangulation:

- We compute the maximum of  $f(v)$  over all  $v = (x, y, z) \in \Lambda_0$  that lie directly below  $\tau$ , in  $O^*(n^{2/3})$  time. Note that for all such  $v$ , the  $\lambda_H(x, y) = \lambda_{H_0}(x, y) = v$ .
- We let  $h$  be the plane through  $\tau$  and compute the maximum of  $f(\lambda_{\{h\}}(x, y))$  over all  $v = (x, y, z) \in \Lambda_0$  that lie directly above  $\tau$ , in  $O^*(n^{2/3})$  time. Note that for all such  $v$ ,  $\lambda_H(x, y) = \lambda_{\{h\}}(x, y)$ .

In addition, for each  $q \in Q_{\text{bad}}$ , we compute  $\lambda_H(q)$  by vertical ray shooting in  $\text{LE}(H_0)$  and  $\text{LE}(H_{\text{bad}})$  in  $O^*(1)$  time; we take the maximum of  $f(\lambda_H(q))$  for these points. Note that  $\text{LE}(H)$  is covered by  $\Gamma$  at all times, since there are at most  $k$  deletions per phase. The overall maximum thus gives the answer. The total time to compute the answer is  $O^*((s + k)n^{2/3})$ .

**Analysis.** The overall amortized update time is

$$O^*(n/k + kn/s + (s + k)n^{2/3}).$$

The theorem follows by setting  $s = k^2$  and  $k = n^{1/9}$ . ◀

We can similarly solve the dynamic 2D discrete 1-center problem, by switching lower with upper envelopes and maximum with minimum:

► **Theorem 7.** *We can maintain the discrete 1-center of a dynamic set of  $n$  points in  $\mathbb{R}^2$ , with  $O^*(n)$  preprocessing time and  $O^*(n^{8/9})$  amortized insertion and deletion time.*

It is possible to slightly improve the  $O^*(n^{8/9})$  bound to  $O^*(n^{5/6})$  in the preceding two theorems: the key observation is that the point set  $\Lambda_0$  is in convex position, and in the convex-position case, the  $O^*(n^{2/3})$  query time for 3D simplex range searching can be improved to  $O^*(\sqrt{n})$ , as shown by Sharir and Zaban [27]. (The same observation also improves the author's previous  $O^*(n^{5/6})$  result to  $O^*(n^{3/4})$  in the semi-online setting.)

It remains open whether the dynamic Hausdorff distance and discrete 1-center problem in dimensions  $d \geq 3$  can similarly be solved in sublinear time. The author's previous paper [9] gave an  $O^*(n^{1-1/(d+1)}(\lceil d/2 \rceil + 1))$ -time algorithm but only in the semi-online setting. In higher dimensions, the size of shallow cuttings becomes too large for the approach to be effective.

#### 4 Dynamic 3D Convex Hull Queries

In this section, we present a slightly improved data structure for extreme point queries for a dynamic 3D convex hull, by combining the author's previous data structure [10] (as refined by Kaplan et al. [21]) with a modification of Chan and Tsakalidis's algorithm for constructing a hierarchy of shallow cuttings [15].

To describe the latter, we need a definition: Given a set  $H$  of  $n$  planes in  $\mathbb{R}^3$  and a collection  $\Gamma_{\text{in}}$  of cells, a  $\Gamma_{\text{in}}$ -restricted  $(k, K)$ -shallow cutting is a collection  $\Gamma_{\text{out}}$  of cells covering  $\{p \in \mathbb{R}^3 : p \text{ is covered by } \Gamma_{\text{in}} \text{ and has level at most } k\}$ , such that each cell in  $\Gamma_{\text{out}}$  intersects at most  $K$  planes. We note that Chan and Tsakalidis's algorithm, with some technical modifications, can prove the following lemma. (The proof requires knowledge of Chan and Tsakalidis's paper, and is deferred to the full paper.)

► **Lemma 8.** *There exist constants  $b, c$ , and  $c'$  such that the following is true: For a set  $H$  of at most  $n$  planes in  $\mathbb{R}^3$  and a parameter  $k \in [1, n]$ , given a  $(-\infty, cbk)$ -shallow cutting<sup>3</sup>  $\Gamma_{\text{in}}$  with at most  $c'n/(bk)$  downward cells, together with their conflict lists, we can construct a  $\Gamma_{\text{in}}$ -restricted  $(k, ck)$ -shallow cutting  $\Gamma_{\text{out}}$  with at most  $c'n/k$  downward cells, together with their conflict lists, in  $O(n + (n/k) \log(n/k))$  deterministic time.*

We now redescribe the author's previous data structure [10] for 3D extreme point queries, with slight changes to incorporate Lemma 8. The redescription uses a recursive form of the logarithmic method [5], which should be a little easier to understand than the original description.

► **Theorem 9.** *We can maintain a set of  $n$  points in  $\mathbb{R}^3$ , with  $O(n \log n)$  preprocessing time,  $O(\log^2 n)$  amortized insertion time, and  $O(\log^4 n)$  amortized deletion time, so that we can answer find the extreme point of the convex hull along any query direction in  $O(\log^2 n)$  time.*

**Proof.** We describe our solution in dual space, where we want to answer vertical ray shooting queries for  $\text{LE}(H)$ , i.e., find the lowest plane of  $H$  at a query vertical line, for a dynamic set  $H$  of  $n$  planes in  $\mathbb{R}^3$ .

**Preprocessing.** Our preprocessing algorithm is given by the pseudocode below (ignoring trivial base cases), with the constants  $b, c, c'$  from Lemma 8:<sup>4</sup>

```

preprocess( $H$ ):
1.  $H_0 = H$ ,  $\Gamma_0 = \{\mathbb{R}^3\}$ ,  $\ell = \log_b n$ 
2. for  $i = 1, \dots, \ell$  do {
3.    $\Gamma_i =$  a  $\Gamma_{i-1}$ -restricted  $(n/b^i, cn/b^i)$ -shallow cutting of  $H_{i-1}$  with at most  $c'b^i$  cells
4.    $H_i = H_{i-1} - \{h \in H : h \text{ intersects more than } 2cc'\ell \text{ cells of } \Gamma_1 \cup \dots \cup \Gamma_i\}$ 
5.   for each  $\Delta \in \Gamma_i$ , compute the conflict list  $(H_i)_\Delta$  and initialize  $k_\Delta = 0$ 
}
```

<sup>3</sup> In a  $(-\infty, k)$ -shallow cutting, the cells are not required to cover any particular region.

<sup>4</sup> Line 4 is where Kaplan et al.'s improvement lies [21]. The original data structure from [10] basically had  $H_i = H_{i-1} - \{h \in H : h \text{ intersects more than } 2cc'\ell \text{ cells of } \Gamma_i\}$ .

## 24:10 Dynamic Geometric Data Structures via Shallow Cuttings

6. preprocess  $H_\ell$  for static vertical ray shooting
7.  $H_{\text{bad}} = H - H_\ell$
8. preprocess( $H_{\text{bad}}$ )

Note that  $\Gamma_{i-1}$  is a  $(-\infty, cn/b^{i-1})$ -shallow cutting of  $H_{i-2}$ , and consequently a  $(-\infty, cn/b^{i-1})$ -shallow cutting of  $H_{i-1}$ , since  $H_{i-1} \subseteq H_{i-2}$ . Given  $\Gamma_{i-1}$  and its conflict lists, we can thus apply Lemma 8 to compute  $\Gamma_i$  and its conflict lists, in  $O(n + b^i \log b^i)$  time. The total time for lines 1–5 is  $O(n \log n + \sum_{i=1}^{\ell} b^i \log b^i) = O(n \log n)$ . Line 6 takes  $O(n \log n)$  time (by a planar point location method [16]).

We claim that  $|H_{\text{bad}}| \leq n/2$ . To see this, consider each  $h \in H_{\text{bad}}$ . Let  $i$  be the index with  $h \in H_{i-1} - H_i$ . Then  $h$  intersects more than  $2cc'\ell$  cells of  $\Gamma_1 \cup \dots \cup \Gamma_i$ ; send a charge from  $h$  to each of these cells. Each cell in  $\Gamma_j$  receives charges only from planes in  $H_{j-1}$  that intersect the cell. Thus, the total number of charges is at least  $2cc'\ell|H_{\text{bad}}|$  and is at most  $\sum_{j=1}^{\ell} cn/b^j \cdot c'b^j = cc'\ell n$ . The claim follows. The preprocessing time thus satisfies the recurrence  $P(n) \leq P(n/2) + O(n \log n)$ , which gives  $P(n) = O(n \log n)$ .

**Inserting a plane  $h$ .** We simply insert  $h$  to  $H_{\text{bad}}$  recursively. When  $|H_{\text{bad}}|$  reaches  $3n/4$ , we rebuild the data structure for  $H$ . It takes  $\Omega(n)$  updates for a rebuild to occur. The amortized insertion time thus satisfies the recurrence  $I(n) \leq I(3n/4) + O(P(n)/n) = I(3n/4) + O(\log n)$ , which gives  $I(n) = O(\log^2 n)$ .

**Deleting a plane  $h$ .** The deletion algorithm is as follows:

- ```
delete( $H, h$ ):
1.  for  $i = 1, \dots, \ell$  do
2.    for each  $\Delta \in \Gamma_i$  with  $h \in (H_i)_\Delta$  do {
3.      increment  $k_\Delta$ 
4.      if  $k_\Delta \geq n/b^{i+1}$  then
5.        for all  $h \in (H_i)_\Delta$  that are still in  $H$  but not yet in  $H_{\text{bad}}$ , insert  $h$  to  $H_{\text{bad}}$ 
6.    }
7.  if  $h \in H_{\text{bad}}$  then delete( $H_{\text{bad}}, h$ )
```

Let  $i$  be the largest index with  $h \in H_i$ . Then  $h$  intersects at most  $2cc'\ell = O(\log n)$  cells of  $\Gamma_1 \cup \dots \cup \Gamma_i$ . Thus, in each deletion, lines 3–5 are executed  $O(\log n)$  times.

In lines 3–5, it takes  $n/b^{i+1}$  increments of  $k_\Delta$  to cause the  $|(H_i)_\Delta| \leq cn/b^i$  planes to be inserted to  $H_{\text{bad}}$ . Thus, each increment triggers  $O(1)$  amortized number of insertions to  $H_{\text{bad}}$ , and so a deletion triggers  $O(\log n)$  amortized number of insertions to  $H_{\text{bad}}$ . The amortized deletion time thus satisfies the recurrence  $D(n) \leq D(3n/4) + O(\log n)I(3n/4) = D(3n/4) + O(\log^3 n)$ , which gives  $D(n) = O(\log^4 n)$ .

**Answering the query for a vertical line  $q$ .** We first answer the query for the static set  $H_\ell$  in  $O(\log n)$  time (by planar point location); if the returned plane has already been deleted, ignore the answer. We then recursively answer the query for  $H_{\text{bad}}$ , and return the lowest of all the planes found. The query time satisfies the recurrence  $Q(n) \leq Q(3n/4) + O(\log n)$ , which gives  $Q(n) = O(\log^2 n)$ .

**Correctness of the query algorithm.** To prove correctness, let  $h^*$  be the lowest plane at  $q$  and  $v^* = h^* \cap q$ . If  $h^* \in H_{\text{bad}}$ , correctness follows by induction. So, assume that  $h^* \notin H_{\text{bad}}$ .

If  $v^*$  is covered by  $\Gamma_\ell$ , say, by the cell  $\Delta \in \Gamma_\ell$ , then either  $v^*$  is on  $\text{LE}(H_\ell)$ , in which case the algorithm would have correctly found  $h^*$ , or some plane in  $(H_\ell)_\Delta$  has been deleted from  $H$ , in which case all active planes of  $(H_\ell)_\Delta$ , including  $h^*$ , would have been inserted to  $H_{\text{bad}}$ .



Otherwise, let  $i$  be an index such that  $v^*$  is not covered by  $\Gamma_i$  but is covered by  $\Gamma_{i-1}$ , say, by the cell  $\Delta \in \Gamma_{i-1}$ . Since  $\Gamma_i$  is a  $\Gamma_{i-1}$ -restricted  $(n/b^i, cn/b^i)$ -shallow cutting of  $H_{i-1}$ , it follows that  $v^*$  must have level more than  $n/b^i$  in  $H_{i-1}$ . In order for  $v^*$  to be the answer, the more than  $n/b^i$  planes of  $H_{i-1}$  below  $v^*$  must have been deleted from  $H$ . But then all active planes of  $(H_{i-1})_\Delta$ , including  $h^*$ , would have been inserted to  $H_{\text{bad}}$ . ◀

By the standard lifting transformation, we obtain:

► **Corollary 10.** *We can maintain a set of  $n$  points in  $\mathbb{R}^2$ , with  $O(n \log n)$  preprocessing time,  $O(\log^2 n)$  amortized insertion time, and  $O(\log^4 n)$  amortized deletion time, so that we can answer find the nearest neighbor to any query point in  $O(\log^2 n)$  time.*

The space usage in the above data structure is  $O(n \log n)$ , but can be improved to  $O(n)$ , by following an idea mentioned in [10] (due to Afshani): instead of storing conflict lists explicitly, generate conflict lists on demand by using a known optimal (static) linear-space data structure for halfspace range reporting [1].

Following [10], we can use the same dynamic data structure to answer other basic types of 3D convex hull queries, e.g., *gift wrapping queries* (finding the two tangents of the hull with a query line outside the hull) in  $O(\log^2 n)$  time and *line-intersection queries* (intersecting the hull with a query line) in  $O(\log^4 n \log^{O(1)} \log n)$  time. The latter corresponds to *3D linear programming queries* in dual space. The dynamic data structure can be adapted to maintain the *smallest enclosing circle* of a 2D point set. Following [12], the dynamic data structure can also be adapted to answer 3D halfspace range reporting queries.

## 5 Dynamic 2D Bichromatic Closest Pair

We now adapt the data structure in Section 4 to solve the dynamic 2D bichromatic closest pair problem:

► **Theorem 11.** *We can maintain the closest pair between two dynamic sets  $P$  and  $Q$  of at most  $n$  points in  $\mathbb{R}^2$ , with  $O(n \log n)$  preprocessing time,  $O(\log^2 n)$  amortized insertion time, and  $O(\log^4 n)$  amortized deletion time.*

**Proof.** By the standard lifting transformation, map each input point  $p = (a, b)$  to the plane  $h_p$  with equation  $z = -2ax - 2by + a^2 + b^2$  in  $\mathbb{R}^3$ . Let  $H = \{h_p : p \in P\}$ . For each point  $q \in Q$ , let  $\lambda_H(q)$  denote the point on  $\text{LE}(H)$  at the vertical line at  $q$ . Let  $J = \{h_q : q \in Q\}$ . For each point  $p \in P$ , define  $\lambda_J(p)$  similarly. We want to compute the minimum of  $f(\lambda_H(q))$  over all  $q \in Q$ , where  $f(x, y, z) = x^2 + y^2 + z$ , which is equivalent to the minimum of  $f(\lambda_J(p))$  over all  $p \in P$ .

**Preprocessing.** We maintain a global heap, whose minimum gives the answer. We modify the  $\text{preprocess}(H)$  algorithm in Section 4:

$\text{preprocess}(H, J)$ :

1. run lines 1–7 of the  $\text{preprocess}(H)$  algorithm on  $H$
2. for each  $h_q \in J$ , add  $f(\lambda_{H_\ell}(q))$  to the heap
3. run lines 1–7 of the  $\text{preprocess}(H)$  algorithm but with  $H$ 's replaced by  $J$ 's
4. for each  $h_p \in H$ , add  $f(\lambda_{J_\ell}(p))$  to the heap
5.  $\text{preprocess}(H_{\text{bad}}, J_{\text{bad}})$

As in Section 4, the preprocessing time satisfies the recurrence  $P(n) \leq P(n/2) + O(n \log n)$ , which gives  $P(n) = O(n \log n)$ .



**Inserting a plane  $h_p$  to  $H$ .** We recursively insert  $h_p$  to  $H_{\text{bad}}$ . We also compute  $\lambda_{J_\ell}(p)$  in  $O(\log n)$  time (by planar point location), and add  $f(\lambda_{J_\ell}(p))$  to the heap.

When  $|H_{\text{bad}}|$  or  $|J_{\text{bad}}|$  reaches  $3n/4$ , we rebuild the data structure for  $H$  and  $J$ . It takes  $\Omega(n)$  updates for a rebuild to occur. The amortized insertion time thus satisfies the recurrence  $I(n) \leq I(3n/4) + O(\log n) + O(P(n)/n) = I(3n/4) + O(\log n)$ , which gives  $I(n) = O(\log^2 n)$ .

**Inserting a plane  $h_q$  to  $J$ .** Symmetric to the above.

**Deleting a plane  $h_p$  from  $H$ .** We run lines 1–5 of the `delete( $H, h$ )` algorithm in Section 4 (with  $h = h_p$ ). In the heap, we remove all entries  $f(\lambda_{H_\ell}(q))$  that has  $\lambda_{H_\ell}(q) = \lambda_{\{h_p\}}(q)$ . If  $h_p \in H_{\text{bad}}$ , we further recursively delete  $h_p$  from  $H_{\text{bad}}$ . We also remove  $f(\lambda_{J_\ell}(p))$  from the heap.

For the analysis, we can charge removals of entries from the heap to their corresponding insertions, by amortization. The amortized deletion time thus satisfies the recurrence  $D(n) \leq D(3n/4) + O(\log n)I(3n/4) = D(3n/4) + O(\log^3 n)$ , which gives  $D(n) = O(\log^4 n)$ .

**Deleting a plane  $h_q$  from  $J$ .** Symmetric to the above.

**Correctness.** Let  $p^*q^*$  be the closest pair with  $p^* \in P$  and  $q^* \in Q$ . If both  $h_{p^*} \in H_{\text{bad}}$  and  $h_{q^*} \in J_{\text{bad}}$ , correctness follows by induction. Otherwise, assume without loss of generality that  $h_{p^*} \notin H_{\text{bad}}$ . (The case  $J_{q^*} \notin J_{\text{bad}}$  is symmetric.) Let  $v^* = \lambda_H(q^*)$ . The rest of the correctness argument is essentially identical to that in Section 4:

If  $v^*$  is covered by  $\Gamma_\ell$ , say, by the cell  $\Delta \in \Gamma_\ell$ , then either  $v^*$  is on  $\text{LE}(H_\ell)$ , in which case the algorithm would have included  $f(\lambda_H(q^*))$  in the heap, or some plane in  $(H_\ell)_\Delta$  has been deleted from  $H$ , in which case all active planes of  $(H_\ell)_\Delta$ , including  $h_{p^*}$ , would have been inserted to  $H_{\text{bad}}$ .

Otherwise, let  $i$  be an index such that  $v^*$  is not covered by  $\Gamma_i$  but is covered by  $\Gamma_{i-1}$ , say, by the cell  $\Delta \in \Gamma_{i-1}$ . Since  $\Gamma_i$  is a  $\Gamma_{i-1}$ -restricted  $(n/b^i, cn/b^i)$ -shallow cutting of  $H_{i-1}$ , it follows that  $v^*$  must have level more than  $n/b^i$  in  $H_{i-1}$ . In order for  $v^*$  to be the answer, the more than  $n/b^i$  planes of  $H_{i-1}$  below  $v^*$  must have been deleted from  $H$ . But then all active planes of  $(H_{i-1})_\Delta$ , including  $h_{p^*}$ , would have been inserted to  $H_{\text{bad}}$ . ◀

We can similarly solve the diameter problem, by replacing min with max and lower with upper envelopes:

► **Theorem 12.** *We can maintain the diameter of a dynamic set of  $n$  points in  $\mathbb{R}^2$ , with  $O(n \log n)$  preprocessing time,  $O(\log^2 n)$  amortized insertion time, and  $O(\log^4 n)$  amortized deletion time.*

---

## References

- 1 Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proc. 20th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 180–186, 2009. URL: <http://dl.acm.org/citation.cfm?id=1496770.1496791>.
- 2 Pankaj K. Agarwal. Simplex range searching and its variants: A review. In M. Loebl, J. Nešetřil, and R. Thomas, editors, *Journal through Discrete Mathematics*. Springer, to appear.
- 3 Pankaj K. Agarwal and Jiří Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.*, 11:393–418, 1994. doi:10.1007/BF02574015.
- 4 Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995. doi:10.1007/BF01293483.
- 5 Jon Louis Bentley and James B. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, 1(4):301–358, 1980. doi:10.1016/0196-6774(80)90015-2.
- 6 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proc. 43rd Sympos. Found. Comput. Sci. (FOCS)*, pages 617–626, 2002. doi:10.1109/SFCS.2002.1181985.

- 7 Timothy M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *J. ACM*, 48(1):1–12, 2001. Preliminary version in FOCS 1999. doi:10.1145/363647.363652.
- 8 Timothy M. Chan. A fully dynamic algorithm for planar width. *Discrete Comput. Geom.*, 30(1):17–24, 2003. Preliminary version in SoCG 2001. doi:10.1007/s00454-003-2923-8.
- 9 Timothy M. Chan. Semi-online maintenance of geometric optima and measures. *SIAM J. Comput.*, 32(3):700–716, 2003. Preliminary version in SODA 2002. doi:10.1137/S0097539702404389.
- 10 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. Preliminary version in SODA 2006. doi:10.1145/1706591.1706596.
- 11 Timothy M. Chan. Optimal partition trees. *Discrete Comput. Geom.*, 47(4):661–690, 2012. Preliminary version in SoCG 2010. doi:10.1007/s00454-012-9410-z.
- 12 Timothy M. Chan. Three problems about dynamic convex hulls. *Int. J. Comput. Geom. Appl.*, 22(4):341–364, 2012. Preliminary version in SoCG 2011. doi:10.1142/S0218195912600096.
- 13 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proc. 27th ACM Sympos. Comput. Geom. (SoCG)*, pages 1–10, 2011. doi:10.1145/1998196.1998198.
- 14 Timothy M. Chan, Mihai Pătraşcu, and Liam Roditty. Dynamic connectivity: Connecting to networks and geometry. *SIAM J. Comput.*, 40(2):333–349, 2011. Preliminary version in FOCS 2008. doi:10.1137/090751670.
- 15 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. *Discrete Comput. Geom.*, 56(4):866–881, 2016. Preliminary version in SoCG 2015. doi:10.1007/s00454-016-9784-4.
- 16 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008. URL: <http://www.worldcat.org/oclc/227584184>.
- 17 David P. Dobkin and Subhash Suri. Maintenance of geometric extrema. *J. ACM*, 38(2):275–298, 1991. doi:10.1145/103516.103518.
- 18 David Eppstein. Dynamic Euclidean minimum spanning trees and extrema of binary functions. *Discrete Comput. Geom.*, 13:111–122, 1995. doi:10.1007/BF02574030.
- 19 David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *ACM Journal of Experimental Algorithmics*, 5:1, 2000. doi:10.1145/351827.351829.
- 20 Jacob Holm, Eva Rotenberg, and Christian Wulff-Nilsen. Faster fully-dynamic minimum spanning forest. In *Proc. 23rd European Sympos. Algorithms (ESA)*, pages 742–753, 2015. doi:10.1007/978-3-662-48350-3\_62.
- 21 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proc. 28th ACM–SIAM Sympos. Discrete Algorithms (SODA)*, pages 2495–2504, 2017. doi:10.1137/1.9781611974782.165.
- 22 Jiří Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992. doi:10.1007/BF02293051.
- 23 Jiří Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992. doi:10.1016/0925-7721(92)90006-E.
- 24 Mark H. Overmars. *The Design of Dynamic Data Structures*, volume 156 of *Lecture Notes in Computer Science*. Springer, 1983. doi:10.1007/BFb0014927.
- 25 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23(2):166–204, 1981. doi:10.1016/0022-0000(81)90012-X.
- 26 Edgar A. Ramos. On range reporting, ray shooting and  $k$ -level construction. In *Proc. 15th Sympos. Comput. Geom. (SoCG)*, pages 390–399, 1999. doi:10.1145/304893.304993.
- 27 Micha Sharir and Shai Zaban. Output-sensitive tools for range searching in higher dimensions. Manuscript, 2013. URL: <http://www.cs.tau.ac.il/~michas/shai.pdf>.



# Lower Bounds for Electrical Reduction on Surfaces

**Hsien-Chih Chang**

Duke University, Durham, USA  
hsienchih.chang@duke.edu

**Marcos Cossarini**

Laboratoire d'Analyse et de Mathématiques Appliquées, Université Paris-Est Marne-la-Vallée,  
Champs-sur-Marne, France  
marcos.cossarini@u-pem.fr

**Jeff Erickson**

University of Illinois at Urbana-Champaign, Champaign, USA  
jeffe@illinois.edu

---

## Abstract

We strengthen the connections between *electrical transformations* and *homotopy* from the planar setting – observed and studied since Steinitz – to arbitrary surfaces with punctures. As a result, we improve our earlier lower bound on the number of electrical transformations required to reduce an  $n$ -vertex graph on surface in the worst case [SOCG 2016] in two different directions. Our previous  $\Omega(n^{3/2})$  lower bound applies only to *facial* electrical transformations on plane graphs with *no terminals*. First we provide a stronger  $\Omega(n^2)$  lower bound when the planar graph has two or more terminals, which follows from a quadratic lower bound on the number of homotopy moves in the annulus. Our second result extends our earlier  $\Omega(n^{3/2})$  lower bound to the wider class of *planar* electrical transformations, which preserve the planarity of the graph but may delete cycles that are not faces of the given embedding. This new lower bound follows from the observation that the *defect* of the medial graph of a planar graph is the same for all its planar embeddings.

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

**Keywords and phrases** electrical transformation,  $\Delta$ -Y-transformation, homotopy, tight, defect, SPQR-tree, smoothings, routing set, 2-flipping

**Digital Object Identifier** 10.4230/LIPIcs.SocG.2019.25

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1707.04683>.

**Funding** This work was partially supported by NSF grant CCF-1408763. We also greatly appreciate the support from Labex Bezout.

*Hsien-Chih Chang:* This work was initiated when the author was affiliated with University of Illinois at Urbana-Champaign, USA.

*Marcos Cossarini:* This work is initiated when the author was affiliated with Instituto de Matemática Pura e Aplicada, Brazil.

## 1 Introduction

Consider the following set of local operations performed on any graph:

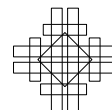
- *Leaf contraction:* Contract the edge incident to a vertex of degree 1.
- *Loop deletion:* Delete the edge of a loop.
- *Series reduction:* Contract either edge incident to a vertex of degree 2.
- *Parallel reduction:* Delete one of a pair of parallel edges.
- *$Y \rightarrow \Delta$  transformation:* Delete a degree-3 vertex and connect its neighbors with three new edges.
- *$\Delta \rightarrow Y$  transformation:* Delete edges of a 3-cycle and join its vertices to a new vertex.



© Hsien-Chih Chang, Marcos Cossarini, and Jeff Erickson;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 25; pp. 25:1–25:16  
Leibniz International Proceedings in Informatics



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



These operations and their inverses, which we call *electrical transformations* following Colin de Verdière et al. [10], have been used for over a century to analyze electrical networks [24]. Steinitz [35, 36] proved that any planar network can be reduced to a single vertex using these operations. Several decades later, Epifanov [16] proved that any planar graph with two special vertices called *terminals* can be similarly reduced to a single edge between the terminals; simpler algorithmic proofs of Epifanov’s theorem were later given by Feo [18], Truemper [40, 41], and Feo and Provan [19]. These results have since been extended to planar graphs with more than two terminals [20, 21, 2, 14] and to some families of non-planar graphs [20, 42]. See Chang’s thesis [5] for a history of the problem.

Despite decades of prior work, the complexity of the reduction process is still poorly understood. Steinitz’s proof implies that  $O(n^2)$  electrical transformations suffice to reduce any  $n$ -vertex planar graph to a single vertex; Feo and Provan’s algorithm reduces any 2-terminal planar graph to a single edge in  $O(n^2)$  steps. While these are the best upper bounds known, several authors have conjectured that they can be improved [20, 19, 2]. Without any restrictions on which transformations are permitted, the only known lower bound is the trivial  $\Omega(n)$ . However, Chang and Erickson recently proved that if all transformations are required to be *facial*, meaning any deleted cycle must be a face of the given embedding, then reducing a plane graph without terminals to a single vertex requires  $\Omega(n^{3/2})$  steps in the worst case [7]. This is obtained by studying the relation between facial electrical transformations and *homotopy moves*, a set of operations performed on the medial graph of the input.

In this paper, we extend our earlier lower bound for electrical transformations in two directions. To this end, first we study multicurves on surfaces under electrical reduction and homotopy moves, which are in one-to-one correspondence with medial graphs. Specifically, in Section 3 we prove that the set of *tight* multicurves under electrical reduction and under homotopy moves is identical. As a consequence, any surface-embedded graph can be reduced without ever increasing its number of edges. Previously such property is only known to hold for plane graphs [27, 7].

Next, we consider plane graphs with two terminals. In this setting, leaf deletions, series reductions, and  $Y \rightarrow \Delta$  transformations that delete terminals are forbidden. We prove in Section 4 that  $\Omega(n^2)$  facial electrical transformations are required in the worst case to reduce a 2-terminal plane graph *as much as possible*. Not every 2-terminal plane graph can be reduced to a single edge between the terminals using only facial electrical transformations. However, we show that any 2-terminal plane graph can be reduced to a unique minimal graph called a *bullseye* using a finite number of facial electrical transformations. Our lower bound ultimately relies on a recent  $\Omega(n^2)$  lower bound on the number of homotopy moves required to tighten a contractible closed curve in the annulus [9].

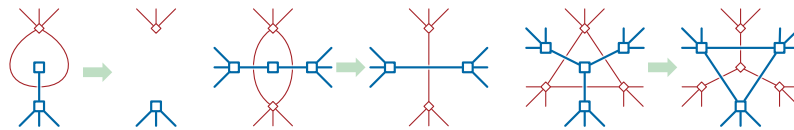
In Section 5, we consider a wider class of electrical transformations that preserve the planarity of the graph, but are not necessarily facial. Our second main result is that  $\Omega(n^{3/2})$  *planar* electrical transformations are required to reduce a planar graph (without terminals) to a single vertex in the worst case. Like our earlier lower bound for *facial* electrical transformations, our proof ultimately reduces to the study of a certain curve invariant, called the *defect*, of the medial graph of a given *unicursal* plane graph  $G$ . A key step in our new proof is the following surprising observation: Although the definition of the medial graph of  $G$  depends on the embedding of  $G$ , the defect of the medial graph is the same for all planar embeddings of  $G$ .

## 2 Background

### 2.1 Types of electrical transformations

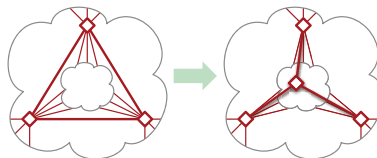
We distinguish between three increasingly general types of electrical transformations in plane graphs: *facial*, *crossing-free*, and *arbitrary*. (For ease of presentation, we assume throughout the paper that plane graphs are actually embedded on the *sphere* instead of the plane.)

An electrical transformation in a graph  $G$  embedded on a surface  $\Sigma$  is *facial* if any deleted cycle is a face of  $G$ . All leaf contractions, series reductions, and  $Y \rightarrow \Delta$  transformations are facial, but loop deletions, parallel reductions, and  $\Delta \rightarrow Y$  transformations may not be facial. Facial electrical transformations form three dual pairs, as shown in Figure 1; for example, any series reduction in  $G$  is equivalent to a parallel reduction in the dual graph  $G^*$ .



■ **Figure 1** Facial electrical transformations in a plane graph  $G$  and its dual  $G^*$ .

An electrical transformation in  $G$  is *crossing-free* if it preserves the embeddability of the underlying graph into the same surface. Equivalently, an electrical transformation is crossing-free if the vertices of the cycle deleted by the transformation are all incident to a common face of  $G$ . All facial electrical transformations are trivially crossing-free, as are all loop deletions and parallel reductions. If the graph embeds in the plane, crossing-free electrical transformations are also called *planar*. The only non-crossing-free electrical transformation is a  $\Delta \rightarrow Y$  transformation whose three vertices are *not* incident to a common face; any such transformation introduces a  $K_{3,3}$ -minor into the graph, connecting the three vertices of the  $\Delta$  to an interior vertex, an exterior vertex, and the new  $Y$  vertex.



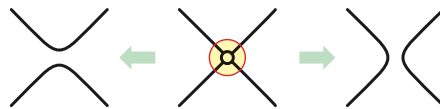
■ **Figure 2** A non-planar  $\Delta \rightarrow Y$  transformation.

### 2.2 Multicurves and medial graphs

A *surface* is a 2-manifold with or without punctures. Formally, a *closed curve* in a surface  $\Sigma$  is a continuous map  $\gamma: S^1 \rightarrow \Sigma$ . A closed curve is *simple* if it is injective. A *multicurve* is a collection of one or more closed curves. We consider only *generic* multicurves, which are injective except at a finite number of (self-)intersections, each of which is a transverse double point. A multicurve is *connected* if its image in the surface is connected; we consider only connected multicurves in this paper. The image of any (non-simple, connected) multicurve has a natural structure as a 4-regular map, whose *vertices* are the self-intersection points of the curves, *edges* are maximal subpaths between vertices, and *faces* are components of complement of the curves in the surface. We do not distinguish between multicurves whose images are combinatorially equivalent maps.

The *medial graph*  $G^\times$  of an embedded graph  $G$  is another embedded graph whose vertices correspond to the edges of  $G$ , and two vertices of  $G^\times$  are connected by an edge if the corresponding edges in  $G$  are consecutive in cyclic order around some vertex, or equivalently, around some face in  $G$ . Every vertex in every medial graph has degree 4; thus, every medial graph is the image of a multicurve. Conversely, when the surface is a sphere, the image of every non-simple multicurve is the medial graph of some plane graph. We call an embedded graph  $G$  *unicursal* if its medial graph  $G^\times$  is the image of a single closed curve.

*Smoothing* a multicurve  $\gamma$  at a vertex  $x$  replaces the intersection of  $\gamma$  with a small neighborhood of  $x$  with two disjoint simple paths, so that the result is another 4-regular embedded graph. There are two possible smoothings at each vertex. More generally, a *smoothing* of  $\gamma$  is any multicurve obtained by smoothing a subset of its vertices.



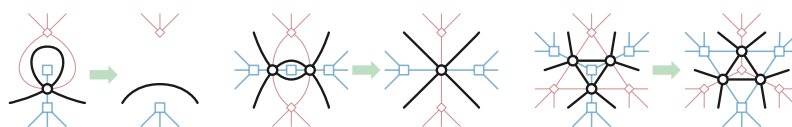
■ **Figure 3** Two possible smoothings of a vertex.

### 2.3 Local moves

A *homotopy* between two curves  $\gamma$  and  $\gamma'$  on the same surface  $\Sigma$  is a continuous deformation from one curve to the other, formally defined as a continuous function  $H: S^1 \times [0, 1] \rightarrow \Sigma$  such that  $H(\cdot, 0) = \gamma$  and  $H(\cdot, 1) = \gamma'$ . The definition of homotopy extends naturally to multicurves. Classical topological arguments imply that two multicurves are homotopic if and only if one can be transformed into the other by a finite sequence of *homotopy moves* (shown in Figure 4). Notice that a 1→0 move is applied to an empty *loop*, and a 2→0 move is applied on an empty *bigon*. A multicurve is *homotopically tight* (or *h-tight* for short) if no sequence of homotopy moves leads to a multicurve with fewer vertices.



■ **Figure 4** Homotopy moves 1→0, 2→0, and 3→3.



■ **Figure 5** Electrical moves 1→0, 2→1, and 3→3.

Facial electrical transformations in any embedded graph  $G$  correspond to local operations in the medial graph  $G^\times$  that closely resemble homotopy moves. We call these 1→0, 2→1, and 3→3 moves, where the numbers before and after each arrow indicate the number of local vertices before and after the move. We collectively refer to these operations and their inverses as *electrical moves*. A multicurve is *electrically tight* (or *e-tight* for short) if no sequence of electrical moves leads to another multicurve with fewer vertices. For multicurves on surfaces with boundary, both homotopy moves and electrical moves performed on boundary faces are forbidden. The fact that we use same name *tight* for both homotopy moves and electrical moves is not a coincidence; we will justify its usage in Section 3.2.

### 3 Connection between electrical and homotopy moves

For any connected multicurve (or 4-regular embedded graph)  $\gamma$  on surface  $\Sigma$ ,

- let  $X(\gamma)$  denote the minimum number of electrical moves required to tighten  $\gamma$ ,
- let  $H^\downarrow(\gamma)$  denote the minimum number of homotopy moves required to tighten  $\gamma$ , without ever increase the number of vertices; that is, no 0→1 and 0→2 moves are allowed.
- let  $H(\gamma)$  denote the minimum number of homotopy moves required to tighten  $\gamma$ .

It is not immediately obvious whether a multicurve  $\gamma$  that is tight under monotonic homotopy moves could be further tightened by allowing 0→1 and 0→2 moves or not. Hass and Scott [22] and de Graaf and Schrijver [13] independently proved that any multicurve  $\gamma$  can be tightened using monotonic homotopy moves, which implies that  $H^\downarrow(\gamma) = 0$  if and only if  $H(\gamma) = 0$ . In other words, (standard) homotopy moves and monotonic homotopy moves share the same set of tight multicurves. Now  $H^\downarrow(\gamma) \geq H(\gamma)$  follows for any multicurve  $\gamma$ .

#### 3.1 Smoothing lemma

We would like to compare  $X(\gamma)$  with  $H^\downarrow(\gamma)$  and  $H(\gamma)$ . The following key lemma follows from close reading of proofs by Truemper [40, Lemma 4] and several others [20, 25, 2, 27] that every minor of a  $\Delta Y$ -reducible graph is also  $\Delta Y$ -reducible.

► **Lemma 1** (Chang and Erickson [7, Lemma 3.1]). *Let  $\gamma$  be any connected multicurve on surface  $\Sigma$ , and let  $\check{\gamma}$  be a connected smoothing of  $\gamma$ . Applying any sequence of  $N$  electrical moves to  $\gamma$  to obtain  $\gamma'$ . Then one can apply a similar sequence of electrical moves of length at most  $N$  to  $\check{\gamma}$  to obtain a (possibly trivial) connected smoothing  $\check{\gamma}'$  of  $\gamma'$ .*

As a remark, using similar argument one can recover a result by Newmann-Coto [26]: any homotopy from multicurve  $\gamma$  to another multicurve  $\gamma'$  that never removes vertices can be turned into a homotopy from a smoothing of  $\gamma$  to a smoothing of  $\gamma'$ .

Using Lemma 1 one can show that  $X(\gamma) \geq H^\downarrow(\gamma)$  for every planar curve  $\gamma$ , a result implicit in the work of Noble and Welsh [27] and formally proved by Chang and Erickson [7].

► **Lemma 2** (Smoothing Lemma [7]).  *$X(\check{\gamma}) \leq X(\gamma)$  for every connected smoothing  $\check{\gamma}$  of every connected multicurve  $\gamma$  in the plane.*

► **Lemma 3** (Monotonicity Lemma [7]). *For every connected multicurve  $\gamma$ , there is a minimum-length sequence of electrical moves that simplifies  $\gamma$  to a simple closed curve that does not contain 0→1 or 1→2 moves.*

► **Lemma 4** (Electrical-Homotopy Ineq. [7]).  *$X(\gamma) \geq H^\downarrow(\gamma)$  for every planar curve  $\gamma$ .*

#### 3.2 Equivalence of tightness

One of the main obstacles to generalize Lemmas 2, 3, and 4 to curves on arbitrary surface is that again we do not know *a priori* whether the set of tight multicurves under electrical moves is the same as those under homotopy moves. Such problem did not exist in the planar setting as all planar multicurves can be tightened to simple curves using either electrical or homotopy moves. We first show that every electrically tight multicurve is also homotopically tight.

► **Lemma 5.** *Let  $\gamma$  be a connected multicurve on an arbitrary surface  $\Sigma$ . If  $\gamma$  is electrically tight, then  $\gamma$  is homotopically tight.*



**Proof.** Let  $\gamma$  be a connected multicurve in some arbitrary surface, and suppose  $\gamma$  is not homotopically tight. Results of Hass and Scott [22] and de Graaf and Schrijver [13] imply that  $\gamma$  can be tightened by a finite sequence of homotopy moves that never increases the number of vertices. In particular, applying some finite sequence of 3→3 moves to  $\gamma$  creates either an empty loop, which can be removed by a 1→0 move, or an empty bigon, which can be removed by either a 2→0 move or a 2→1 move. Thus,  $\gamma$  is not electrically tight. ◀

However, for the reverse direction, we don't have a similar monotonicity result for electrical moves on arbitrary surfaces. A careful reading of the sequence of work by de Graaf and Schrijver [29, 30, 32, 31, 11, 12, 13] leads to a five-way equivalence that shows the two versions of tightness coincide when the given curve is *primitive*. Unfortunately their results do not generalize as some of the equivalences break down with the presence of non-primitive counterexamples. We defer details to the full version of the paper.

**Routing set.** Inspired by the routing problem studied by de Graaf and Schrijver [12], we introduce the notion of *routing set*. Despite its naïve look, the routing set satisfies a crucial property that encapsulates the whole difficulty of the problem, which allows us to bypass the heavy machinery developed for the primitive case. We then use the established equivalence of tightness to derive the monotonicity lemma for electrical moves on arbitrary multicurves.

For any multicurve  $\gamma$ , the *routing set* of  $\gamma$  is the following collection of homotopy classes:

$$route(\gamma) := \{ [\check{\gamma}] \mid \check{\gamma} \text{ is a smoothing of } \gamma \}.$$

Each homotopy class in  $route(\gamma)$  is referred as a *route* of  $\gamma$ .

► **Lemma 6.** *Routing set of  $\gamma$  is invariant under electrical moves for any multicurve  $\gamma$ .*

**Proof.** Let  $\gamma'$  be the multicurve obtained from performing one electrical move to  $\gamma$ . Because electrical moves are closed under inverses, we only need to prove that  $route(\gamma) \subseteq route(\gamma')$ .

Let  $\check{\gamma}$  be an arbitrary smoothing of  $\gamma$ ;  $[\check{\gamma}]$  is in  $route(\gamma)$  by definition. By Lemma 1, one can obtain a connected smoothing  $\check{\gamma}'$  of  $\gamma'$  that is at most one electrical move away from  $\check{\gamma}$ . In particular,  $[\check{\gamma}']$  is in  $route(\gamma')$ . If  $\check{\gamma}'$  is equal to  $\check{\gamma}$  or is obtained from  $\check{\gamma}$  using a 1→0, 0→1, or 3→3 move, then immediately we have  $[\check{\gamma}] = [\check{\gamma}']$  to be a route in  $route(\gamma')$ . If  $\check{\gamma}'$  is obtained from  $\check{\gamma}$  using a 2→1 move, consider the multicurve  $\check{\gamma}^\circ$  obtained from  $\check{\gamma}$  by performing a 2→0 move (on the same empty bigon) instead.  $\check{\gamma}^\circ$  is a smoothing of  $\check{\gamma}'$ , which in turn is a smoothing of  $\gamma'$ . Because 2→0 is a homotopy move,  $[\check{\gamma}] = [\check{\gamma}^\circ]$  is a route in  $route(\gamma')$ . Similarly when  $\check{\gamma}'$  is obtained from  $\check{\gamma}$  using a 1→2 move, we consider  $\check{\gamma}$  as a smoothing of  $\check{\gamma}'$  thus  $[\check{\gamma}]$  is a route in  $route(\gamma')$ . This concludes the proof. ◀

The *intersection number* of a homotopy class  $[\gamma]$  is defined to be the minimum number of vertices among all curves homotopic to  $\gamma$ . The *main routes* of  $\gamma$  are those routes of  $\gamma$  that achieve the maximum intersection number.

► **Lemma 7.** *Any homotopically tight multicurve is also electrically tight.*

**Proof.** Assume for contradiction that there is an h-tight multicurve  $\gamma$  that is not e-tight. Tighten  $\gamma$  using electrical moves to an e-tight multicurve  $\gamma'$  with less number of vertices than  $\gamma$ . Now by Lemma 6 the routing set of  $\gamma$  and  $\gamma'$  is the same; in particular,  $[\gamma']$  is a main route of both  $\gamma$  and  $\gamma'$ . However since both  $\gamma$  and  $\gamma'$  are h-tight, the intersection number of  $[\gamma]$  is strictly greater than the intersection number of  $[\gamma']$  and thus  $[\gamma']$  cannot be a main route of  $\gamma$ , a contradiction. ◀

### 3.3 Monotonicity of electrical moves

As a corollary of Lemma 7, we are ready to generalize the monotonicity lemma (Lemma 3) to multicurves on general surfaces.

► **Lemma 8.** *Let  $\gamma$  be any connected multicurve  $\gamma$  on surface  $\Sigma$ , and let  $\check{\gamma}$  be a connected smoothing of  $\gamma$ , satisfying  $\text{route}(\gamma) = \text{route}(\check{\gamma})$ . Then  $X(\check{\gamma}) \leq X(\gamma)$  holds.*

**Proof.** Let  $\gamma$  be a connected multicurve with  $n(\gamma)$  vertices, and let  $\check{\gamma}$  be a connected smoothing of  $\gamma$ . If  $X(\gamma)$  equals to zero, then  $\gamma$  is both e-tight and h-tight by Lemma 5. The fact that  $\text{route}(\gamma) = \text{route}(\check{\gamma})$  implies that  $[\gamma]$  is a route of  $\check{\gamma}$  and its intersection number is equal to  $n(\gamma)$ . If  $\check{\gamma}$  is a proper smoothing of  $\gamma$ , then the intersection number of any route of  $\check{\gamma}$  is strictly less than  $n(\gamma)$ , a contradiction. As a result, the only smoothing of  $\gamma$  satisfying the condition is  $\gamma$  itself, and therefore the inequality trivially holds.

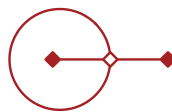
Otherwise, applying a minimum-length sequence of electrical moves that tightens  $\gamma$ . By Lemma 1 there is another sequence of electrical moves of length at most  $X(\gamma)$  that tightens  $\check{\gamma}$ . We immediately have  $X(\check{\gamma}) \leq X(\gamma)$  and the lemma is proved. ◀

► **Lemma 9.** *For any connected multicurve  $\gamma$ , there is a minimum-length sequence of electrical moves that tightens  $\gamma$  that does not contain  $0 \rightarrow 1$  or  $1 \rightarrow 2$  moves.*

The proof follows almost verbatim from Lemma 3 after substituting Lemma 8 for Lemma 2 and applying Lemma 6. See the full version for a proof.

## 4 Two-terminal plane graphs

Most applications of electrical reductions, starting with Kennelly's computation of effective resistance [24], designate two vertices of the input graph as *terminals* and require a reduction to a single edge between those terminals. In this context, electrical transformations that delete either of the terminals are forbidden; specifically: leaf contractions when the leaf is a terminal, series reductions when the degree-2 vertex is a terminal, and  $Y \rightarrow \Delta$  transformations when the degree-3 vertex is a terminal. An important subtlety here is that not every 2-terminal planar graph can be reduced to a single edge using only *facial* electrical transformations. The simplest bad example is the three-vertex graph shown in Figure 6.



■ **Figure 6** A facially irreducible 2-terminal plane graph; solid vertices are the terminals.

In this section, we show that in the worst case,  $\Omega(n^2)$  facial electrical transformations are required to reduce a 2-terminal plane graph with  $n$  vertices *as much as possible*. The medial graph  $G^\times$  of any 2-terminal plane graph  $G$  is properly considered as a multicurve embedded in the annulus; the faces of  $G^\times$  that correspond to the terminals are removed from the surface. The main strategy is to lower bound  $X(G^\times)$  by some function of  $H(G^\times)$ , then defer to the quadratic lower bound for untangling annular curve using homotopy moves [9]. To this end, we generalize Lemma 4 to annular curves; such result is obtained by the understanding of tight multicurves on the annulus.

First, we prove in Section 4.1 that any annular curve can be tightened to a unique family of curves. Next in Section 4.2, we generalize the results by Chang and Erickson [7], in particular the electrical-homotopy inequality (Lemma 4), to the annular case. We prove

our quadratic lower bound in Section 4.3. Existing algorithms for reducing an arbitrary 2-terminal plane graphs to a single edge rely on an additional operation which we call a *terminal-leaf contraction*, in addition to facial electrical transformations. We discuss this subtlety in more detail in the full version.

### 4.1 Tight annular curves

The *winding number* of a directed closed curve  $\gamma$  in the annulus is the number of times any generic path  $\pi$  from one (fixed) boundary component to the other crosses  $\gamma$  from left to right, minus the number of times  $\pi$  crosses  $\gamma$  from right to left. Two directed closed curves in the annulus are homotopic if and only if their winding numbers are equal.

The *depth* of any multicurve  $\gamma$  in the annulus is the minimum number of times a path from one boundary to the other crosses  $\gamma$ ; thus, depth is essentially an unsigned version of winding number. Just as the winding number around the boundaries is a complete homotopy invariant for curves in the annulus, the depth turns out to be a complete invariant for electrical moves on the annular multicurves.

► **Lemma 10.** *Electrical moves do not change the depth of any annular multicurve.*

For any integer  $d > 0$ , let  $\alpha_d$  denote the unique closed curve in the annulus with  $d - 1$  vertices and winding number  $d$ . Up to isotopy, this curve can be parametrized in the plane as

$$\alpha_d(\theta) := ((\cos(\theta) + 2) \cos(d\theta), (\cos(\theta) + 2) \sin(d\theta)).$$

In the notation of our other papers [7, 8],  $\alpha_d$  is the *flat torus knot*  $T(d, 1)$ .

The following lemmas are direct consequences of Lemma 7.

► **Lemma 11.** *For any integer  $d > 0$ , the curve  $\alpha_d$  is both  $h$ -tight and  $e$ -tight.*

► **Corollary 12.** *A connected multicurve  $\gamma$  in the annulus is  $e$ -tight if and only if  $\gamma = \alpha_{\text{depth}(\gamma)}$ ; therefore, any annular multicurve  $\gamma$  is  $e$ -tight if and only if  $\gamma$  is  $h$ -tight.*

### 4.2 Smoothing lemma in the annulus

Equipped with the understanding of tight annular curves, we are ready to extend the results in Section 3.1 to the annulus.

► **Lemma 13.** *For any connected smoothing  $\check{\gamma}$  of any connected multicurve  $\gamma$  in the annulus, we have  $X(\check{\gamma}) + \frac{1}{2} \text{depth}(\check{\gamma}) \leq X(\gamma) + \frac{1}{2} \text{depth}(\gamma)$ .*

**Proof.** Let  $\gamma$  be an arbitrary connected multicurve in the annulus, and let  $\check{\gamma}$  be an arbitrary connected smoothing of  $\gamma$ . Without loss of generality, we can assume that  $\gamma$  is non-simple, since otherwise the lemma is vacuous.

If  $\gamma$  is already  $e$ -tight, then  $\gamma = \alpha_d$  for some integer  $d \geq 2$  by Corollary 12. (The curves  $\alpha_0$  and  $\alpha_1$  are simple.) First, suppose  $\check{\gamma}$  is a connected smoothing of  $\gamma$  obtained by smoothing a single vertex  $x$ . The smoothed curve  $\check{\gamma}$  contains a single empty loop if  $x$  is the innermost or outermost vertex of  $\gamma$ , or a single empty bigon otherwise. Applying one  $1 \rightarrow 0$  or  $2 \rightarrow 0$  move transforms  $\check{\gamma}$  into the curve  $\alpha_{d-2}$ , which is  $e$ -tight by Lemma 11. Thus we have  $X(\check{\gamma}) = 1$  and  $\text{depth}(\check{\gamma}) = d - 2$ , which implies  $X(\check{\gamma}) + \frac{1}{2} \text{depth}(\check{\gamma}) = X(\gamma) + \frac{1}{2} \text{depth}(\gamma)$ . As for the general case when  $\check{\gamma}$  is obtained from  $\gamma$  by smoothing more than one vertices, the statement follows from the previous case by induction on the number of smoothed vertices.

If  $\gamma$  is not  $e$ -tight, applying a minimum-length sequence of electrical moves that tightens  $\gamma$  into some curve  $\gamma'$ . By Lemma 1 there is another sequence of electrical moves of length at most  $X(\gamma)$  that tightens  $\check{\gamma}$  to some connected smoothing  $\check{\gamma}'$  of  $\gamma'$ , which can be further

tightened electrically to an e-tight curve using arguments in the previous paragraph because  $\gamma'$  is e-tight. This implies that  $X(\check{\gamma}) \leq X(\gamma) + \frac{1}{2}(\text{depth}(\gamma') - \text{depth}(\check{\gamma}'))$ . By Lemma 10,  $\gamma$  and  $\gamma'$  have the same depth, and  $\check{\gamma}$  and  $\check{\gamma}'$  have the same depth. Therefore  $X(\check{\gamma}) + \frac{1}{2} \text{depth}(\check{\gamma}) \leq X(\gamma) + \frac{1}{2} \text{depth}(\gamma)$  and the lemma is proved.  $\blacktriangleleft$

► **Lemma 14.** *For every connected multicurve  $\gamma$  in the annulus, there is a minimum-length sequence of electrical moves that tightens  $\gamma$  to  $\alpha_{\text{depth}(\gamma)}$  without 0→1 or 1→2 moves.*

The proof follows almost verbatim from Lemma 3 and 9; see the full version.

► **Lemma 15.**  $X(\gamma) + \frac{1}{2} \text{depth}(\gamma) \geq H^\downarrow(\gamma) \geq H(\gamma)$  for every closed curve  $\gamma$  in the annulus.

**Proof.** Let  $\gamma$  be a closed curve in the annulus. If  $\gamma$  is already e-tight, then  $X(\gamma) = H^\downarrow(\gamma) = 0$  by Lemma 5, so the lemma is trivial. Otherwise, consider a minimum-length sequence of electrical moves that tightens  $\gamma$ . By Lemma 14, we can assume that the first move in the sequence is neither 0→1 nor 1→2. If the first move is 1→0 or 3→3, the theorem immediately follows by induction on  $X(\gamma)$ , since by Lemma 10 neither of these moves changes the depth of the curve.

The only interesting first move is 2→1. Let  $\gamma'$  be the result of this 2→1 move, and let  $\gamma^\circ$  be the result if we perform the 2→0 move on the same empty bigon instead. The minimality of the sequence implies  $X(\gamma) = X(\gamma') + 1$ , and we trivially have  $H^\downarrow(\gamma) \leq H^\downarrow(\gamma^\circ) + 1$ . Because  $\gamma$  is a single curve,  $\gamma^\circ$  is also a single curve and therefore a connected proper smoothing of  $\gamma'$ . Thus, Lemma 10, Lemma 13, and induction on the number of vertices imply

$$\begin{aligned} X(\gamma) + \frac{1}{2} \text{depth}(\gamma) &= X(\gamma') + \frac{1}{2} \text{depth}(\gamma') + 1 \\ &\geq X(\gamma^\circ) + \frac{1}{2} \text{depth}(\gamma^\circ) + 1 \\ &\geq H^\downarrow(\gamma^\circ) + 1 \\ &\geq H^\downarrow(\gamma), \end{aligned}$$

which completes the proof.  $\blacktriangleleft$

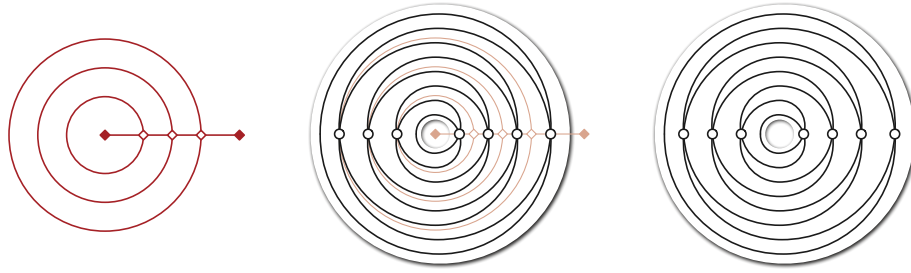
### 4.3 Quadratic lower bound

**Bullseyes.** For any  $k > 0$ , let  $B_k$  denote the 2-terminal plane graph that consists of a path of length  $k$  between the terminals, with a loop attached to each of the  $k - 1$  interior vertices, embedded so that collectively they form concentric circles that separate the terminals. We call each graph  $B_k$  a *bullseye*. For example,  $B_1$  is just a single edge;  $B_2$  is shown in Figure 6; and  $B_4$  is shown on the left in Figure 7. The medial graph  $B_k^\times$  of the  $k$ th bullseye is the curve  $\alpha_{2k}$ . Because different bullseyes have different medial depths, Lemma 10 implies that no bullseye can be transformed into any other bullseye by facial electrical transformations.

The following corollary is now immediate from the electrical-homotopy inequality for annular curves (Lemma 15).

► **Theorem 16.** *Let  $G$  be a 2-terminal plane graph, and let  $\gamma$  be any unicursal smoothing of  $G^\times$ . Reducing  $G$  to a bullseye requires at least  $H(\gamma) - \frac{1}{2} \text{depth}(\gamma)$  facial electrical transformations.*

Chang et al. [9] presented an infinite family of contractible curves in the annulus parametrized by their number of vertices  $n$  that require  $\Omega(n^2)$  homotopy moves to tighten. Every contractible curve is the medial graph of some 2-terminal plane graph (because they have



■ **Figure 7** The bullseye graph  $B_4$  and its medial graph  $\alpha_8$ .

even depth and thus the faces can be two-colored [37]). Euler’s formula implies that every  $n$ -vertex curve in the annulus has exactly  $n + 2$  faces (including the boundary faces) and therefore has depth at most  $n + 1$ .

► **Corollary 17.** *Reducing a 2-terminal plane graph to a bullseye requires  $\Omega(n^2)$  facial electrical transformations in the worst case.*

## 5 Planar electrical transformations

Finally, we extend our earlier  $\Omega(n^{3/2})$  lower bound for reducing plane graphs and our  $\Omega(n^2)$  lower bound for reducing graphs on surface – *without* terminals using only facial electrical transformations – to the larger class of *planar* electrical transformations. Recall that a plane graph  $G$  is *unicursal* if its medial graph  $G^\times$  is the image of a single closed curve. As in our earlier work [7], we analyze electrical transformations in an unicursal plane graph  $G$  in terms of a certain invariant of the medial graph of  $G$  called *defect*, first introduced by Aicardi [1] and Arnold [4, 3]. Our extension to non-facial electrical transformations is based on the following surprising observation: Although the medial graph of  $G$  depends on its embedding, the *defect* of the medial graph of  $G$  does not.

► **Theorem 18.** *Let  $G$  and  $H$  be planar embeddings of the same abstract planar graph. If  $G$  is unicursal, then  $H$  is unicursal and  $\text{defect}(G^\times) = \text{defect}(H^\times)$ .*

The goal of the section is to prove Theorem 18.

### 5.1 Defect

Let  $\gamma$  be an arbitrary closed curve on the sphere. Choose an arbitrary basepoint  $\gamma(0)$  and an arbitrary orientation for  $\gamma$ . For any vertex  $x$  of  $\gamma$ , we define  $\text{sgn}(x) = +1$  if the first traversal through  $x$  crosses the second traversal from right to left, and  $\text{sgn}(x) = -1$  otherwise. Two vertices  $x$  and  $y$  are *interleaved*, denoted  $x \bowtie y$ , if they alternate in cyclic order –  $x, y, x, y$  – along  $\gamma$ . Finally, following Polyak [28], we can define

$$\text{defect}(\gamma) := -2 \sum_{x \bowtie y} \text{sgn}(x) \cdot \text{sgn}(y),$$

where the sum is taken over all interleaved pairs of vertices of  $\gamma$ .

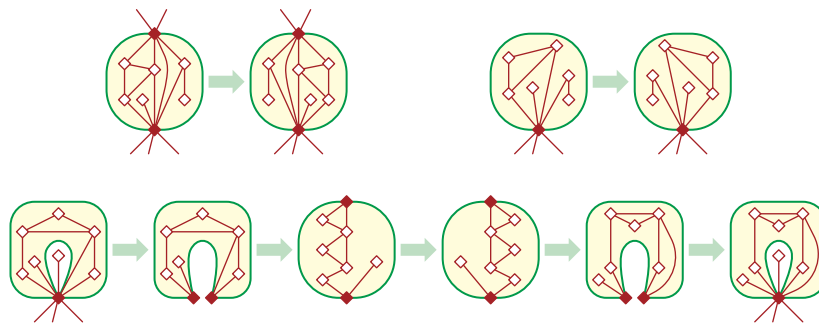
Trivially, every simple closed curve has defect zero. Straightforward case analysis [28] implies that the defect of a curve does not depend on the choice of basepoint or orientation. Moreover, any homotopy move changes the defect of a curve by at most 2; see the paper by Chang and Erickson [7, Section 2.1] for an explicit case breakdown. Defect is also preserved by any homeomorphism from the sphere to itself, including reflection.

### 5.2 Navigating between planar embeddings

Whitney [43, 39] showed that any planar embedding of a 2-connected planar graph  $G$  can be transformed into any other embedding by a finite sequence of *split reflections*, defined as follows. A *split curve* is a simple closed curve  $\sigma$  whose intersection with the embedding of  $G$  consists of two vertices  $x$  and  $y$ ; without loss of generality,  $\sigma$  is a circle with  $x$  and  $y$  at opposite points. A split reflection modifies the embedding of  $G$  by reflecting the subgraph inside  $\sigma$  across the line through  $x$  and  $y$ .

► **Lemma 19.** *Let  $G$  be an arbitrary 2-connected planar graph. Any two planar embeddings of  $G$  can be transformed into one other by a finite sequence of split reflections.*

To navigate among the planar embeddings of *arbitrary* connected planar graphs, we need two additional operations. First, we allow split curves that intersect  $G$  at only a single cut vertex; a *cut reflection* modifies the embedding of  $G$  by reflects the subgraph inside such a curve. More interestingly, we also allow degenerate split curves that pass through a cut vertex  $x$  of  $G$  *twice*, but are otherwise simple and disjoint from  $G$ . The interior of a degenerate split curve  $\sigma$  is an open topological disk. A *cut eversion* is a degenerate split reflection that everts the embedding of the subgraph of  $G$  inside such a curve, intuitively by mapping the interior of  $\sigma$  to an open circular disk (with two copies of  $x$  on its boundary), reflecting the interior subgraph, and then mapping the resulting embedding back to the interior of  $\sigma$ . Structural results of Stallman [33, 34] and Di Battista and Tamassia [15, Section 7] imply the following.



■ **Figure 8** Top row: A regular split reflection and a cut reflection. Bottom row: a cut eversion.

► **Lemma 20.** *Let  $G$  be an arbitrary connected planar graph. Any planar embedding of  $G$  can be transformed into any other planar embedding of  $G$  by a finite sequence of split reflections, cut reflections, and cut eversions.*

### 5.3 Tangle flips

Now consider the effect of the operations stated in Lemma 20 on the medial graph  $G^\times$ . By assumption,  $G$  is unicursal so that  $G^\times$  is a single closed curve. Let  $\sigma$  be any (possibly degenerate) split curve for  $G$ . Embed  $G^\times$  so that every medial vertex lies on the corresponding edge in  $G$ , and every medial edge intersects  $\sigma$  at most once. By the Jordan curve theorem, we can assume without loss of generality that  $\sigma$  is a circle, and that the intersection points  $\gamma \cap \sigma$  are evenly spaced around  $\sigma$ . A *tangle* of  $\gamma$  is the intersection of  $\gamma$  with either disk bounded by  $\sigma$ ; each tangle consists of one or more subpaths of  $\gamma$  called *strands*. We arbitrarily refer to the two tangles defined by  $\sigma$  as the *interior* and *exterior* tangles of  $\sigma$ . Split curve  $\sigma$  intersects at most four edges of  $G^\times$ , so the tangle of  $G^\times$  inside  $\sigma$  has at most two strands.

## 25:12 Lower Bounds for Electrical Reduction on Surfaces

Moreover, reflecting (or everting) the subgraph of  $G$  inside  $\sigma$  induces a *flip* of this tangle of  $G^\times$ . Any tangle can be *flipped* by reflecting the disk containing it, so that each strand endpoint maps to a different strand endpoint; see Figure 9. Straightforward case analysis implies that flipping any tangle of  $G^\times$  with at most two strands transforms  $G^\times$  into another closed curve; see Figure 10.



■ **Figure 9** Flipping tangles with one and two strands.

► **Lemma 21.** *Let  $\gamma$  be an arbitrary closed curve on the sphere. Flipping any tangle of  $\gamma$  with one strand yields another closed curve  $\gamma'$  with  $\text{defect}(\gamma') = \text{defect}(\gamma)$ .*

**Proof.** Let  $\sigma$  be a simple closed curve that crosses  $\gamma$  at exactly two points. These points decompose  $\sigma$  into two subpaths  $\alpha \cdot \beta$ , where  $\alpha$  is the unique strand of the interior tangle and  $\beta$  is the unique strand of the exterior tangle. Let  $\Sigma$  denote the interior disk of  $\sigma$ , and let  $\phi: \Sigma \rightarrow \Sigma$  denote the homeomorphism that flips the interior tangle. Flipping the interior tangle yields the closed curve  $\gamma' := \text{rev}(\phi(\alpha)) \cdot \beta$ , where  $\text{rev}$  denotes path reversal.

No vertex of  $\alpha$  is interleaved with a vertex of  $\beta$ ; thus, two vertices in  $\gamma'$  are interleaved if and only if the corresponding vertices in  $\gamma$  are interleaved. Every vertex of  $\text{rev}(\phi(\alpha))$  has the same sign as the corresponding vertex of  $\alpha$ , since both the orientation of the vertex and the order of traversals through the vertex changed. Thus, every vertex of  $\gamma'$  has the same sign as the corresponding vertex of  $\gamma$ . We conclude that  $\text{defect}(\gamma') = \text{defect}(\gamma)$ . ◀

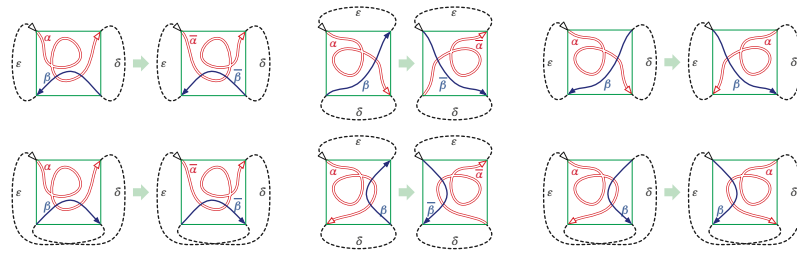
A tangle is *tight* if each strand is simple and each pair of strands crosses at most once. Any tangle can be *tightened* – that is, transformed into a tight tangle – by continuously deforming the strands without crossing  $\sigma$  or moving their endpoints, and therefore by a finite sequence of homotopy moves. Let  $\gamma \pitchfork \sigma$  and  $\gamma \cup \sigma$  denote the closed curves that result from tightening the interior and exterior tangles of  $\sigma$ , respectively.<sup>1</sup> The following lemma that flipping any 2-strand tangle does not change its defect follows from our inclusion-exclusion formula for defect [6, Lemma 5.4]; we give a simpler proof here to keep the paper self-contained.

► **Lemma 22.** *Let  $\gamma$  be an arbitrary closed curve on the sphere. Flipping any tangle of  $\gamma$  with two strands yields another closed curve  $\gamma'$  with  $\text{defect}(\gamma') = \text{defect}(\gamma)$ .*

**Proof.** Let  $\sigma$  be a simple closed curve that crosses  $\gamma$  at exactly four points. These four points naturally decompose  $\gamma$  into four subpaths  $\alpha \cdot \delta \cdot \beta \cdot \varepsilon$ , where  $\alpha$  and  $\beta$  are the strands of the interior tangle of  $\sigma$ , and  $\delta$  and  $\varepsilon$  are the strands of the exterior tangle. Flipping the interior tangle either exchanges  $\alpha$  and  $\beta$ , reverses  $\alpha$  and  $\beta$ , or both; see Figure 10. In every case, the result is a single closed curve  $\gamma'$ . We classify each vertex of  $\gamma$  as *interior* if it lies on  $\alpha$  and/or  $\beta$ , and *exterior* otherwise. Similarly, we classify pairs of interleaved vertices are either interior, exterior, or mixed.

<sup>1</sup> We recommend pronouncing  $\pitchfork$  as “tightened inside” and  $\cup$  as “tightened outside”; note that the symbols  $\pitchfork$  and  $\cup$  resemble the second letters of “inside” and “outside”.





■ **Figure 10** Flipping all six types of 2-strand tangle.

An interior vertex  $x$  and an exterior vertex  $y$  are interleaved if and only if  $x$  is an intersection point of  $\alpha$  and  $\beta$  and  $y$  is an intersection point of  $\delta$  and  $\epsilon$ . Thus, the total contribution of mixed vertex pairs to Polyak’s formula  $defect(\gamma) = -2 \sum_{x \bowtie y} \text{sgn}(x) \cdot \text{sgn}(y)$  is

$$-2 \sum_{x \in \alpha \cap \beta} \sum_{y \in \delta \cap \epsilon} \text{sgn}(x) \cdot \text{sgn}(y) = -2 \left( \sum_{x \in \alpha \cap \beta} \text{sgn}(x) \right) \left( \sum_{y \in \delta \cap \epsilon} \text{sgn}(y) \right).$$

Consider any sequence of homotopy moves that tightens the interior tangle with strands  $\alpha$  and  $\beta$ . Any  $2 \rightarrow 0$  move involving both  $\alpha$  and  $\beta$  removes one positive and one negative vertex; no other homotopy move changes the number of vertices in  $\alpha \cap \beta$  or the signs of those vertices. Thus, tightening  $\alpha$  and  $\beta$  leaves the sum  $\sum_{x \in \alpha \cap \beta} \text{sgn}(x)$  unchanged. Similarly, tightening the exterior tangle  $\delta \cup \epsilon$  leaves the sum  $\sum_{y \in \delta \cap \epsilon} \text{sgn}(y)$  unchanged. But after tightening both tangles, either  $\alpha$  and  $\beta$  are disjoint, or  $\delta$  and  $\epsilon$  are disjoint, or both, as  $\gamma$  is a single closed curve. Thus, at least one of the sums  $\sum_{x \in \alpha \cap \beta} \text{sgn}(x)$  and  $\sum_{y \in \delta \cap \epsilon} \text{sgn}(y)$  is equal to zero. We conclude that mixed vertex pairs do not contribute to the defect.

The curve  $\gamma \pitchfork \sigma$  obtained by tightening  $\alpha$  and  $\beta$  has at most one interior vertex (and therefore no interior vertex pairs); the exterior vertices of  $\gamma \pitchfork \sigma$  are precisely the exterior vertices of  $\gamma$ . Similarly, the curve  $\gamma \cup \sigma$  obtained by tightening both  $\delta$  and  $\epsilon$  has at most one exterior vertex; the interior vertices of  $\gamma \cup \sigma$  are precisely the interior vertices of  $\gamma$ . It follows that  $defect(\gamma) = defect(\gamma \cup \sigma) + defect(\gamma \pitchfork \sigma)$ .

Finally, let  $\gamma'$  be the result of flipping the interior tangle. The curve  $\gamma' \cup \sigma$  is just a reflection of  $\gamma \cup \sigma$ , which implies that  $defect(\gamma' \cup \sigma) = defect(\gamma \cup \sigma)$ , and straightforward case analysis implies  $\gamma' \pitchfork \sigma = \gamma \pitchfork \sigma$ . We conclude that  $defect(\gamma') = defect(\gamma' \pitchfork \sigma) + defect(\gamma' \cup \sigma) = defect(\gamma \pitchfork \sigma) + defect(\gamma \cup \sigma) = defect(\gamma)$ . ◀

Lemmas 20, 21, and 22 now immediately imply Theorem 18.

### 5.4 Back to planar electrical moves

Each planar electrical transformation in a plane graph  $G$  induces the same change in the medial graph  $G^\times$  as a finite sequence of 1- and 2-strand tangle flips (hereafter simply called “tangle flips”) followed by a single electrical move. For an arbitrary connected multicurve  $\gamma$ , let  $\bar{X}(\gamma)$  denote the minimum number of electrical moves in a mixed sequence of electrical moves and tangle flips that tightens  $\gamma$ . Similarly, let  $\bar{H}(\gamma)$  denote the minimum number of homotopy moves in a mixed sequence of homotopy moves and tangle flips that tightens  $\gamma$ . We emphasize that tangle flips are “free” and do not contribute to either  $\bar{X}(\gamma)$  or  $\bar{H}(\gamma)$ .

Our lower bound on planar electrical moves follows our earlier lower bound proof for facial electrical moves almost verbatim; the only subtlety is that the embedding of the graph can effectively change at every step of the reduction. A complete proof can be found in the full version of the paper.



► **Lemma 23.**  $\bar{X}(\gamma) \geq \bar{H}(\gamma) \geq |\text{defect}(\gamma)|/2$  for every closed curve  $\gamma$  on the sphere.

► **Theorem 24.** Let  $G$  be an arbitrary planar graph, and let  $\gamma$  be any unicursal smoothing of  $G^\times$  (defined with respect to any planar embedding of  $G$ ). Reducing  $G$  to a single vertex requires at least  $|\text{defect}(\gamma)|/2$  planar electrical transformations.

Finally, Hayashi et al. [23] and Even-Zohar et al. [17] describe infinite families of planar closed curves with defect  $\Omega(n^{3/2})$ ; see also [7, Section 2.2].

► **Corollary 25.** Reducing any  $n$ -vertex planar graph to a single vertex requires  $\Omega(n^{3/2})$  planar electrical transformations in the worst case.

## 6 Open problems

Our results suggest several open problems. Perhaps the most compelling, and the primary motivation for our work, is to find either a subquadratic upper bound or a quadratic lower bound on the number of (unrestricted) electrical transformations required to reduce any planar graph without terminals to a single vertex. Like Gitler [20], Feo and Provan [19], and Archdeacon et al. [2], we conjecture that  $O(n^{3/2})$  facial electrical transformations suffice. However, proving the conjecture appears to be challenging.

Another direction is to prove a quadratic lower bound for graphs on surfaces with positive genus under *crossing-free* electrical transformations. To generalize Theorem 18 to surface-embedded graphs, we need an extension of Lemma 20 to navigate through all the possible embeddings. Using the theory of *large-edgewidth (LEW) embeddings*, a result by Thomassen [38, Theorem 6.1] shows that any embedding of a surface-embedded graph can be obtained from the LEW-embedding (if there's one) by a finite sequence of split reflections. From here it is not hard to construct a toroidal curve that admits an LEW-embedding and has quadratic defect. The main difficulty is that we don't have a similar electrical-homotopy inequality for arbitrary surfaces.

Finally, none of our lower bound techniques imply anything about non-planar electrical transformations or about electrical reduction of non-planar graphs. Indeed, the only lower bound known in the most general setting, for *any* family of electrically reducible graphs, is the trivial  $\Omega(n)$ . It seems unlikely that planar graphs can be reduced more quickly by using non-planar electrical transformations, but we can't prove anything. Any non-trivial lower bound for this problem would be interesting.

---

## References

- 1 Francesca Aicardi. Tree-like curves. In Vladimir I. Arnold, editor, *Singularities and Bifurcations*, volume 21 of *Advances in Soviet Mathematics*, pages 1–31. Amer. Math. Soc., 1994.
- 2 Dan Archdeacon, Charles J. Colbourn, Isidoro Gitler, and J. Scott Provan. Four-terminal reducibility and projective-planar wye-delta-wye-reducible graphs. *J. Graph Theory*, 33(2):83–93, 2000.
- 3 Vladimir I. Arnold. Plane curves, their invariants, perestroikas and classifications. In Vladimir I. Arnold, editor, *Singularities and Bifurcations*, volume 21 of *Adv. Soviet Math.*, pages 33–91. Amer. Math. Soc., 1994.
- 4 Vladimir I. Arnold. *Topological Invariants of Plane Curves and Caustics*, volume 5 of *University Lecture Series*. Amer. Math. Soc., 1994.
- 5 Hsien-Chih Chang. *Tightening curves and graphs on surfaces*. Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2018.
- 6 Hsien-Chih Chang and Jeff Erickson. Electrical reduction, homotopy moves, and defect. Preprint, October 2015. [arXiv:1510.00571](https://arxiv.org/abs/1510.00571).

- 7 Hsien-Chih Chang and Jeff Erickson. Untangling Planar Curves. In *Proc. 32nd Int. Symp. Comput. Geom.*, volume 51 of *Leibniz International Proceedings in Informatics*, pages 29:1–29:15, 2016. URL: <http://drops.dagstuhl.de/opus/volltexte/2016/5921>.
- 8 Hsien-Chih Chang and Jeff Erickson. Unwinding Annular Curves and Electrically Reducing Planar Networks. Accepted to *Computational Geometry: Young Researchers Forum, Proc. 33rd Int. Symp. Comput. Geom.*, 2017.
- 9 Hsien-Chih Chang, Jeff Erickson, David Letscher, Arnaud de Mesmay, Saul Schleimer, Eric Sedgwick, Dylan Thurston, and Stephan Tillmann. Tightening Curves on Surfaces via Local Moves. Submitted, 2017.
- 10 Yves Colin de Verdière, Isidoro Gitler, and Dirk Vertigan. Réseaux électriques planaires II. *Comment. Math. Helvetici*, 71:144–167, 1996.
- 11 Maurits de Graaf and Alexander Schrijver. Characterizing homotopy of systems of curves on a compact surface by crossing numbers. *Linear Alg. Appl.*, 226–228:519–528, 1995.
- 12 Maurits de Graaf and Alexander Schrijver. Decomposition of graphs on surfaces. *J. Comb. Theory Ser. B*, 70:157–165, 1997.
- 13 Maurits de Graaf and Alexander Schrijver. Making curves minimally crossing by Reidemeister moves. *J. Comb. Theory Ser. B*, 70(1):134–156, 1997.
- 14 Lino Demasi and Bojan Mohar. Four terminal planar Delta-Wye reducibility via rooted  $K_{2,4}$  minors. In *Proc. 26th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1728–1742, 2015.
- 15 Giuseppe Di Battista and Roberto Tamassia. On-line planarity testing. *SIAM J. Comput.*, 25(5):956–997, 1996. doi:10.1137/S0097539794280736.
- 16 G. V. Epifanov. Reduction of a plane graph to an edge by a star-triangle transformation. *Dokl. Akad. Nauk SSSR*, 166:19–22, 1966. In Russian. English translation in *Soviet Math. Dokl.* 7:13–17, 1966.
- 17 Chaim Even-Zohar, Joel Hass, Nati Linial, and Tahl Nowik. Invariants of Random Knots and Links. *Discrete & Computational Geometry*, 56(2):274–314, 2016. arXiv:1411.3308.
- 18 Thomas A. Feo. *I. A Lagrangian Relaxation Method for Testing The Infeasibility of Certain VLSI Routing Problems. II. Efficient Reduction of Planar Networks For Solving Certain Combinatorial Problems*. PhD thesis, Univ. California Berkeley, 1985. URL: <http://search.proquest.com/docview/303364161>.
- 19 Thomas A. Feo and J. Scott Provan. Delta-wye transformations and the efficient reduction of two-terminal planar graphs. *Oper. Res.*, 41(3):572–582, 1993.
- 20 Isidoro Gitler. *Delta-wye-delta Transformations: Algorithms and Applications*. PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, 1991.
- 21 Isidoro Gitler and Feliú Sagols. On terminal delta-wye reducibility of planar graphs. *Networks*, 57(2):174–186, 2011.
- 22 Joel Hass and Peter Scott. Shortening curves on surfaces. *Topology*, 33(1):25–43, 1994.
- 23 Chuichiro Hayashi, Miwa Hayashi, Minori Sawada, and Sayaka Yamada. Minimal unknotting sequences of Reidemeister moves containing unmatched RII moves. *J. Knot Theory Ramif.*, 21(10):1250099 (13 pages), 2012. arXiv:1011.3963.
- 24 Arthur Edwin Kennelly. Equivalence of triangles and three-pointed stars in conducting networks. *Electrical World and Engineer*, 34(12):413–414, 1899.
- 25 Hiroyuki Nakahara and Hiromitsu Takahashi. An Algorithm for the Solution of a Linear system by  $\Delta$ -Y Transformations. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E79-A(7):1079–1088, 1996. Special Section on Multi-dimensional Mobile Information Network.
- 26 Max Neumann-Coto. A characterization of shortest geodesics on surfaces. *Algebraic & Geometric Topology*, 1:349–368, 2001.
- 27 Steven D. Noble and Dominic J. A. Welsh. Knot Graphs. *J. Graph Theory*, 34(1):100–111, 2000.
- 28 Michael Polyak. Invariants of curves and fronts via Gauss diagrams. *Topology*, 37(5):989–1009, 1998.

## 25:16 Lower Bounds for Electrical Reduction on Surfaces

- 29 Alexander Schrijver. Homotopy and crossing of systems of curves on a surface. *Linear Alg. Appl.*, 114–115:157–167, 1989.
- 30 Alexander Schrijver. Decomposition of graphs on surfaces and a homotopic circulation theorem. *J. Comb. Theory Ser. B*, 51(2):161–210, 1991.
- 31 Alexander Schrijver. Circuits in graphs embedded on the torus. *Discrete Math.*, 106/107:415–433, 1992.
- 32 Alexander Schrijver. On the uniqueness of kernels. *J. Comb. Theory Ser. B*, 55:146–160, 1992.
- 33 Matthias F. M. Stallmann. Using PQ-trees for planar embedding problems. Technical Report NCSU-CSC TR-85-24, Dept. Comput. Sci., NC State Univ., December 1985. URL: [https://people.engr.ncsu.edu/mfms/Publications/1985-TR\\_NCSU\\_CSC-PQ\\_Trees.pdf](https://people.engr.ncsu.edu/mfms/Publications/1985-TR_NCSU_CSC-PQ_Trees.pdf).
- 34 Matthias F. M. Stallmann. On counting planar embeddings. *Discrete Math.*, 122:385–392, 1993.
- 35 Ernst Steinitz. Polyeder und Raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, III.AB(12):1–139, 1916.
- 36 Ernst Steinitz and Hans Rademacher. *Vorlesungen über die Theorie der Polyeder: unter Einschluss der Elemente der Topologie*, volume 41 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1934. Reprinted 1976.
- 37 Peter Guthrie Tait. On Knots I. *Proc. Royal Soc. Edinburgh*, 28(1):145–190, 1876–7.
- 38 Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Comb. Theory Ser. B*, 48(2):155–177, 1990. doi:10.1016/0095-8956(90)90115-G.
- 39 Klaus Truemper. On Whitney’s 2-isomorphism theorem for graphs. *Journal of Graph Theory*, 4(1):43–49, 1980.
- 40 Klaus Truemper. On the delta-wye reduction for planar graphs. *J. Graph Theory*, 13(2):141–148, 1989.
- 41 Klaus Truemper. *Matroid Decomposition*. Academic Press, 1992.
- 42 Donald Wagner. Delta-wye reduction of almost-planar graphs. *Discrete Appl. Math.*, 180:158–167, 2015.
- 43 Hassler Whitney. 2-isomorphic graphs. *American Journal of Mathematics*, 55(1):245–254, 1933.

# Maintaining the Union of Unit Discs Under Insertions with Near-Optimal Overhead

**Pankaj K. Agarwal**

Department of Computer Science, Duke University, Durham, NC 27708, USA  
pankaj@cs.duke.edu

**Ravid Cohen**

School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel  
ravidcohn@gmail.com

**Dan Halperin**

School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel  
danha@post.tau.ac.il

**Wolfgang Mulzer** 

Institut für Informatik, Freie Universität Berlin, 14195 Berlin, Germany  
mulzer@inf.fu-berlin.de

---

## Abstract

We present efficient data structures for problems on unit discs and arcs of their boundary in the plane. (i) We give an output-sensitive algorithm for the dynamic maintenance of the union of  $n$  unit discs under insertions in  $O(k \log^2 n)$  update time and  $O(n)$  space, where  $k$  is the combinatorial complexity of the structural change in the union due to the insertion of the new disc. (ii) As part of the solution of (i) we devise a fully dynamic data structure for the maintenance of lower envelopes of pseudo-lines, which we believe is of independent interest. The structure has  $O(\log^2 n)$  update time and  $O(\log n)$  vertical ray shooting query time. To achieve this performance, we devise a new algorithm for finding the intersection between two lower envelopes of pseudo-lines in  $O(\log n)$  time, using *tentative* binary search; the lower envelopes are special in that at  $x = -\infty$  any pseudo-line contributing to the first envelope lies below every pseudo-line contributing to the second envelope. (iii) We also present a dynamic range searching structure for a set of circular arcs of unit radius (not necessarily on the boundary of the union of the corresponding discs), where the ranges are unit discs, with  $O(n \log n)$  preprocessing time,  $O(n^{1/2+\varepsilon} + \ell)$  query time and  $O(\log^2 n)$  amortized update time, where  $\ell$  is the size of the output and for any  $\varepsilon > 0$ . The structure requires  $O(n)$  storage space.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** lower envelopes, pseudo-lines, unit discs, range search, dynamic algorithms, tentative binary search

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.26

**Related Version** A full version of the paper is available at [3], <http://arxiv.org/abs/1903.10943>.

**Funding** *Pankaj K. Agarwal*: Work on this paper is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by grant 2012/229 from the U.S.-Israel Binational Science Foundation.

*Ravid Cohen*: Work by D.H. and R.C. has been supported in part by the Israel Science Foundation (grant no. 825/15), by the Blavatnik Computer Science Research Fund, by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University, and by grants from Yandex and from Facebook.

*Wolfgang Mulzer*: Partially supported by ERC STG 757609 and GIF grant 1367/2016.

**Acknowledgements** We thank Haim Kaplan and Micha Sharir for helpful discussions.



© Pankaj K. Agarwal, Ravid Cohen, Dan Halperin, and Wolfgang Mulzer;  
licensed under Creative Commons License CC-BY

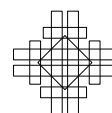
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 26; pp. 26:1–26:15

Leibniz International Proceedings in Informatics



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



## 1 Introduction

Let  $S$  be set of  $n$  points in  $\mathbb{R}^2$ , and let  $U$  be the union of the unit discs centered at the points of  $S$ . We would like to maintain the boundary  $\partial U$  of  $U$ , as new points are added to  $S$ . Even for discs of varying radii, the complexity of  $\partial U$  is  $O(n)$  [17], and it can be computed in  $O(n \log n)$  time using *power diagrams* [6]. An incremental algorithm [20] can maintain  $\partial U$  in total of  $O(n^2)$  time. This is worst-case optimal, as the overall complexity of the structural changes to  $\partial U$  under  $n$  insertions may be  $\Omega(n^2)$ ; see an example in [3]. Here, we describe in Section 3 an output-sensitive algorithm that uses  $O(n)$  space and updates  $\partial U$  in  $O(k \log^2 n)$  time per insertion of a disc, where  $k$  is the combinatorial complexity of the structural changes to  $\partial U$  due to the insertion. Some of our ideas resemble those of de Berg et al. [12], who present a semi-dynamic (insertion only) point-location data structure for  $U$ .

The efficient manipulation of collections of unit discs is a widely and frequently studied topic, for example in the context of sensor networks, where every disc represents the area covered by a sensor. Here, we are motivated by multi-agent coverage of a region in search of a target [11], where we investigate the pace of coverage and wish to estimate at each stage the portion of the overall area covered up to a certain point in time. Since the simulation is discretized (i.e., each agents is modeled by a unit disc whose motion is simulated by changing its location at fixed time steps), we can apply the structure above to update the area of the union within the same time bound. We give more details in Section 3.

A set of pseudo-lines in the plane is a set of infinite  $x$ -monotone curves each pair of which intersects at exactly one point. Arrangements of pseudo-lines have been intensively studied in discrete and computational geometry; see the recent survey on arrangements [14] for a review of combinatorial bounds and algorithms for arrangements of pseudo-lines. At the heart of our solution to the dynamic maintenance of  $U$  lies an efficient data structure for the following problem: Given  $n$  pseudo-lines in the plane, dynamically maintain their lower envelope such that one can efficiently answer vertical ray shooting queries from  $y = -\infty$ . Here, the dynamization allows insertions and deletions. For the case of lines (rather than pseudo-lines), there are several efficient data structures to choose from [8, 9, 18, 7, 16]; these are, however, not directly applicable for pseudo-lines. Also, there are powerful general structures based on shallow cuttings [5, 10, 15]. These structures can handle general families of algebraic curves of bounded description complexity and typically also work in  $\mathbb{R}^3$ . However, the additional flexibility comes at a cost: the algorithms are quite involved, the performance guarantees are in the expected and amortized sense, and the operations have (comparatively) large polylogarithmic running times. For pseudo-lines, Chan’s method [10], with improvements by Kaplan et al. [15], yields  $O(\log^3 n)$  amortized expected insertion time,  $O(\log^5 n)$  amortized expected deletion time, and  $O(\log^2 n)$  worst-case query time. The solution that we propose here is, however, considerably simpler and more efficient: We devise a fully dynamic data structure with  $O(\log^2 n)$  worst-case update-time,  $O(\log n)$  worst-case ray-shooting query-time, and  $O(n)$  space. Additionally, we describe how to find all pseudo-lines below a given query point in  $O(\log n + k \log^2 n)$  time, where  $k$  is the output size. The structure is an adaptation of the Overmars-van Leeuwen structure [18], matching the performance of the original structure for the case of lines. The key innovation is a new algorithm for finding the intersection between two lower envelopes of planar pseudo-lines in  $O(\log n)$  time, using *tentative* binary search (where each pseudo-line in one envelope is “smaller” than every pseudo-line in the other envelope, in a sense to be made precise below). To the best of our knowledge this is the most efficient data structure for the case of pseudo-lines to date.

For our solution to the union-maintenance problem, we need to answer intersection-searching queries of the form: Given the collection  $\mathcal{C}$  of unit-radius circular arcs that comprise  $\partial U$  and a query unit disc  $D$ , report the arcs in  $\mathcal{C}$  intersecting  $D$ . This problem is a special case of the intersection searching problem in which we wish to preprocess a set of geometric objects into a data structure so that the set of objects intersected by a query object can be reported efficiently. Intersection-searching queries are typically answered using multi-level partition trees; see the recent survey [1] for a comprehensive review. Our final result is a data structure for the intersection-searching problem in which the input objects are arbitrary unit-radius circular arcs rather than arcs forming the boundary of the union of the unit discs, and the query is a unit disc. We present a linear-size data structure with  $O(n \log n)$  preprocessing time,  $O(n^{1/2+\delta} + \ell)$  query time and  $O(\log^2 n)$  amortized update time, where  $\ell$  is the size of the output and  $\delta > 0$  is a small constant. Note that because of lack of space, many proofs are omitted from this version and can be found in [3].

## 2 Dynamic lower envelope for pseudo-lines

We describe a data structure to dynamically maintain the lower envelope of an arrangement of planar pseudo-lines under insertions and deletions. Even though we present our data structure for pseudo-lines, it holds for more general classes of planar curves; see below.

### 2.1 Preliminaries

Let  $E$  be a planar family of pseudo-lines, and let  $\ell$  be a vertical line strictly to the left of the first intersection point in  $E$ . The line  $\ell$  defines a total order  $\leq$  on the pseudo-lines in  $E$ , namely for  $e_1, e_2 \in E$ , we have  $e_1 \leq e_2$  if and only if  $e_1$  intersects  $\ell$  below  $e_2$ . Since each pair of pseudo-lines in  $E$  crosses exactly once, it follows that if we consider a vertical line  $\ell'$  strictly to the right of the last intersection point in  $E$ , the order of the intersection points between  $\ell'$  and  $E$ , from bottom to top, is exactly reversed.

The *lower envelope*  $\mathcal{L}(E)$  of  $E$  is the  $x$ -monotone curve obtained by taking the pointwise minimum of the pseudo-lines in  $E$ . Combinatorially, the lower envelope  $\mathcal{L}(E)$  is a sequence of connected segments of the pseudo-lines in  $E$ , where the first and last segment are unbounded. Two properties are crucial for our data structure: (A) every pseudo-line contributes at most one segment to  $\mathcal{L}(E)$ ; and (B) the order of these segments corresponds exactly to the order  $\leq$  on  $E$  defined above. In fact, our data structure works for every set of planar curves with properties (A) and (B) (with an appropriate order  $\leq$ ), even if they are not pseudo-lines in the strict sense; this fact will prove useful in Section 3 below.

We assume a computational model in which primitive operations on pseudo-lines, such as computing the intersection point of two pseudo-lines or determining the intersection point of a pseudo-line with a vertical line can be performed in constant time.

### 2.2 Data structure and operations

**The tree structure.** Our primary data structure is a balanced binary search tree  $\Xi$ . Such a tree data structure supports insert and delete, each in  $O(\log n)$  time. The leaves of  $\Xi$  contain the pseudo-lines, from left to right in the sorted order defined above. An internal node  $v \in \Xi$  represents the lower envelope of the pseudo-lines in its subtree. More precisely, every leaf  $v$  of  $\Xi$  stores a single pseudo-line  $e_v \in E$ . For an inner node  $v$  of  $\Xi$ , we write  $E(v)$  for the set of pseudo-lines in the subtree rooted at  $v$ . We denote the lower envelope of  $E(v)$  by  $\mathcal{L}(v)$ . The inner node  $v$  has the following variables:



## 26:4 Maintaining the Union of Unit Discs

- $f, \ell, r$ : a pointer to the parent, left child and right child of  $v$ , respectively;
- $\max$ : the *last* pseudo-line in  $E(V)$  (last in the ordering defined in Section 2.1)
- $\Lambda$ : a balanced binary search tree that stores the prefix or suffix of  $\mathcal{L}(v)$  that is not on the lower envelope  $\mathcal{L}(f)$  of the parent (in the root, we store the lower envelope of  $E$ ). The leaves of  $\Lambda$  store the pseudo-lines that support the segments on the lower envelope, with the endpoints of the segments, sorted from left to right. An inner node of  $\Lambda$  stores the common point of the last segment in the left subtree and the first segment in the right subtree. We will need split and join operations on the binary trees, which can be implemented in  $O(\log n)$  time.

**Queries.** We now describe the query operations available on our data structure. In a *vertical ray-shooting query*, we are given a value  $x_0 \in \mathbb{R}$ , and we would like to find the pseudo-line  $e \in E$  where the vertical line  $\ell : x = x_0$  intersects  $\mathcal{L}(E)$ . Since the root of  $\Xi$  explicitly stores  $\mathcal{L}(E)$  in a balanced binary search tree, this query can be answered easily in  $O(\log n)$  time.

► **Lemma 1.** *Let  $\ell : x = x_0$  be a vertical ray shooting query. We can find the pseudo-line(s) where  $\ell$  intersects  $\mathcal{L}(E)$  in  $O(\log n)$  time.*

► **Lemma 2.** *Let  $q \in \mathbb{R}^2$ . We can report all pseudo-lines in  $E$  that lie below  $q \in \mathbb{R}^2$  in total time  $O(\log n + k \log^2 n)$ , where  $k$  is the output size*

**Update.** To insert or delete a pseudo-line  $e$  in  $\Xi$ , we follow the method of Overmars and van Leeuwen [18]. We delete or insert a leaf for  $e$  in  $\Xi$  using standard binary search tree techniques (the  $v.\max$  pointers guide the search in  $\Xi$ ). As we go down, we construct the lower envelopes for the nodes hanging off the search path, using split and join operations on the  $v.\Lambda$  trees. Going back up, we recompute the information  $v.\Lambda$  and  $v.\max$ . To update the  $v.\Lambda$  trees, we need the following operation: given two lower envelopes  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$ , such that all pseudo-lines in  $\mathcal{L}_\ell$  are smaller than all pseudo-lines in  $\mathcal{L}_r$ , compute the intersection point  $q$  of  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$ . In the next section, we see how to do this in  $O(\log n)$  time, where  $n$  is the size of  $E$ . Since there are  $O(\log n)$  nodes in  $\Xi$  affected by an update, this procedure takes  $O(\log^2 n)$  time. More details can be found in [18, 19].

► **Lemma 3.** *It takes  $O(\log^2 n)$  to insert or delete a pseudo-line in  $\Xi$ .*

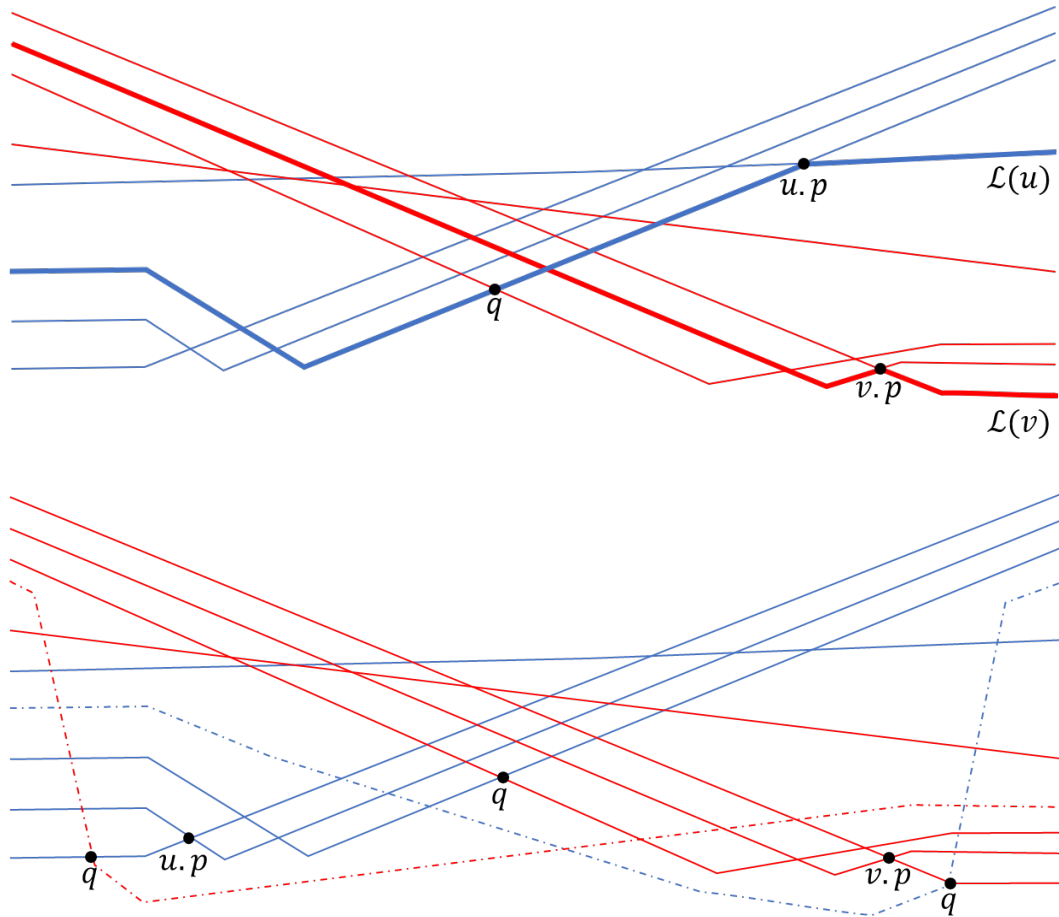
### 2.3 Finding the intersection point of two lower envelopes

Given two lower envelopes  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$  such that all pseudo-lines in  $\mathcal{L}_\ell$  are smaller than all pseudo-lines in  $\mathcal{L}_r$ , we would like to find the intersection point  $q$  between  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$  in  $O(\log n)$  time. We assume that  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$  are represented as balanced binary search trees. The leaves of  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$  store the pseudo-line segments on the lower envelopes, sorted from left to right. We assume that the pseudo-line segments in the leaves are half-open, containing their right, but not their left endpoint in  $\mathcal{L}_\ell$ ; and their left, but not their right endpoint in  $\mathcal{L}_r$ .<sup>1</sup> Thus, it is uniquely determined which leaves of  $\mathcal{L}_\ell$  and  $\mathcal{L}_r$  contain the intersection point  $q$ . A leaf  $v$  stores the pseudo-line  $\mathcal{L}(v)$  that supports the segment for  $v$ , as well as an endpoint  $v.p$  of the segment, namely the left endpoint if  $v$  is a leaf of  $\mathcal{L}_\ell$ , and the right

---

<sup>1</sup> We actually store both endpoints in the trees, but the intersection algorithm uses only one of them, depending on the role the tree plays in the algorithm.

endpoint if  $v$  is a leaf of  $\mathcal{L}_r$ .<sup>2</sup> An inner node  $v$  stores the intersection point  $v.p$  between the last pseudo-line in the left subtree  $v.l$  of  $v$  and the first pseudo-line in the right subtree  $v.r$  of  $v$ , together with the lower envelope  $\mathcal{L}(v)$  of these two pseudo-lines. These trees can be obtained by appropriate split and join operations from the  $\Lambda$  trees stored in  $\Xi$ .



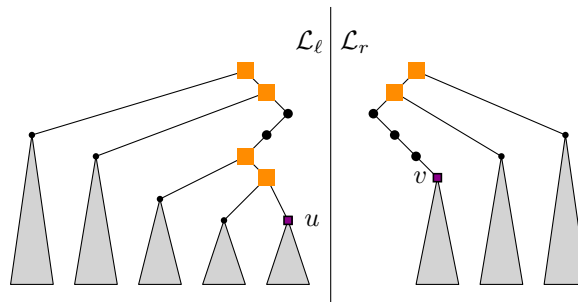
**Figure 1** (On the top). An example of case 1. Case 2 is symmetric. (On the bottom). An example of Case 3.  $\mathcal{L}_\ell$  is blue;  $\mathcal{L}_r$  is red. The solid pseudo-lines are fixed. The dashed pseudo-lines are optional, namely, either none of the dashed pseudo-lines exists or exactly one of them exists.  $u.p$  and  $v.p$  are the current points; and Case 3 applies. Irrespective of the local situation at  $u$  and  $v$ , the intersection point can be to the left of  $u.p$ , between  $u.p$  and  $v.p$ , or to the right of  $v.p$ , depending on which one of the dashed pseudo-lines exists.

Let  $u^* \in \mathcal{L}_\ell$  and  $v^* \in \mathcal{L}_r$  be the leaves whose segments contain  $q$ . Let  $\pi_\ell$  be the path in  $\mathcal{L}_\ell$  from the root to  $u^*$  and  $\pi_r$  the path in  $\mathcal{L}_r$  from the root to  $v^*$ . Our strategy is as follows: we simultaneously descend into  $\mathcal{L}_\ell$  and into  $\mathcal{L}_r$ . Let  $u$  be the current node in  $\mathcal{L}_\ell$  and  $v$  the current node in  $\mathcal{L}_r$ . In each step, we perform a local test on  $u$  and  $v$  to decide how to proceed. There are three possible outcomes:

<sup>2</sup> If the segment is unbounded, the endpoint might not exist. In this case, we use a symbolic endpoint at infinity that lies below every other pseudo-line.

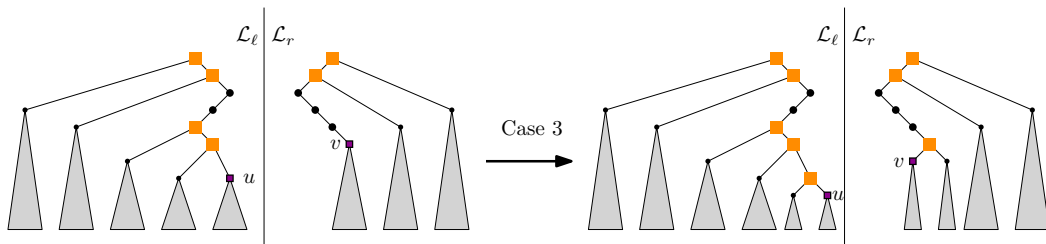


1.  $u.p$  is on or above  $\mathcal{L}(v)$ : the intersection point  $q$  is equal to or to the left of  $u.p$ . If  $u$  is an inner node, then  $u^*$  cannot lie in  $u.r$ ; if  $u$  is a leaf, then  $u^*$  lies strictly to the left of  $u$ ;
2.  $v.p$  lies on or above  $\mathcal{L}(u)$ : the intersection point  $q$  is equal to or to the right of  $v.p$ . If  $v$  is an inner node, then  $v^*$  cannot lie in  $v.l$ ; if  $v$  is a leaf, then  $v^*$  lies strictly to the right of  $v$ ;
3.  $u.p$  lies below  $\mathcal{L}(v)$  and  $v.p$  lies below  $\mathcal{L}(u)$ : then,  $u.p$  lies strictly to the left of  $v.p$  (since we are dealing with pseudo-lines). It must be the case that  $u.p$  is strictly to the left of  $q$  or  $v.p$  is strictly to the right of  $q$  (or both). In the former case, if  $u$  is an inner node,  $u^*$  lies in or to the right of  $u.r$  and if  $u$  is a leaf, then  $u^*$  is  $u$  or a leaf to the right of  $u$ . In the latter case, if  $v$  is an inner node,  $v^*$  lies in or to the left of  $v.l$  and if  $v$  is a leaf, then  $v^*$  is  $v$  or a leaf to the left of  $v$ ; see Figure 1.



■ **Figure 2** The invariant: the current search nodes are  $u$  and  $v$ .  $\mathbf{uStack}$  contains all nodes on the path from the root to  $u$  where the path goes to a right child (orange squares),  $\mathbf{vStack}$  contains all nodes from the root to  $v$  where the path goes to a left child (orange squares). The final leaves  $u^*$  and  $v^*$  are in one of the gray subtrees; and at least one of them is under  $u$  or under  $v$ .

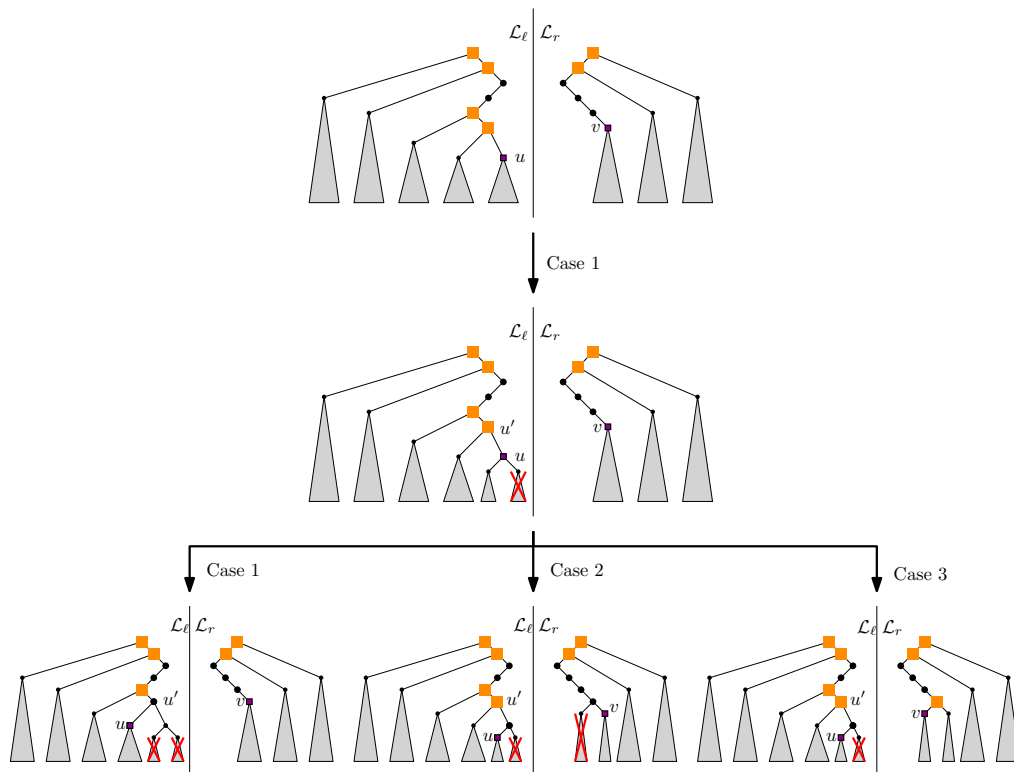
Although it is clear how to proceed in the first two cases, it is not immediately obvious how to proceed in the third case, because the correct step might be either to go to  $u.r$  or to  $v.l$ . In the case of lines, Overmars and van Leeuwen can solve this ambiguity by comparing the slopes of the relevant lines. For pseudo-lines, however, this does not seem to be possible. For an example, refer to Figure 1, where the local situation at  $u$  and  $v$  does not determine the position of the intersection point  $q$ . Therefore, we present an alternative strategy.



■ **Figure 3** Comparing  $u$  to  $v$ : in Case 3, we know that  $u^*$  is in  $u.r$  or  $v^*$  is in  $v.l$ ; we go to  $u.r$  and to  $v.l$ .

We will maintain the invariant that the subtree at  $u$  contains  $u^*$  or the subtree at  $v$  contains  $v^*$  (or both). In Case 3,  $u^*$  must be in  $u.r$ , or  $v^*$  must be in  $v.l$ ; see Figure 3. We move  $u$  to  $u.r$  and  $v$  to  $v.l$ . One of these moves must be correct, but the other move might be mistaken: we might have gone to  $u.r$  even though  $u^*$  is in  $u.l$  or to  $v.l$  even though  $v^*$  is in  $v.r$ . To correct this, we remember the current  $u$  in a stack  $\mathbf{uStack}$  and the current  $v$  in a stack  $\mathbf{vStack}$ , so that we can revisit  $u.l$  or  $v.r$  if it becomes necessary. This leads to the

general situation shown in Figure 2:  $u^*$  is below  $u$  or in a left subtree of a node on  $uStack$ , and  $v^*$  is below  $v$  or in a right subtree of a node on  $vStack$ , and at least one of  $u^*$  or  $v^*$  must be below  $u$  or  $v$ , respectively. Now, if Case 1 occurs when comparing  $u$  to  $v$ , we can exclude the possibility that  $u^*$  is in  $u.r$ . Thus,  $u^*$  might be in  $u.l$ , or in the left subtree of a node in  $uStack$ ; see Figure 4. To make progress, we now compare  $u'$ , the top of  $uStack$ , with  $v$ . Again, one of the three cases occurs. In Case 1, we can deduce that going to  $u'.r$  was mistaken, and we move  $u$  to  $u'.l$ , while  $v$  does not move. In the other cases, we cannot rule out that  $u^*$  is to the right of  $u'$ , and we move  $u$  to  $u.l$ , keeping the invariant that  $u^*$  is either below  $u$  or in the left subtree of a node on  $uStack$ . However, to ensure that the search progresses, we now must also move  $v$ . In Case 2, we can rule out  $v.l$ , and we move  $v$  to  $v.r$ . In Case 3, we move  $v$  to  $v.l$ . In this way, we keep the invariant and always make progress: in each step, we either discover a new node on the correct search paths, or we pop one erroneous move from one of the two stacks. Since the total length of the correct search paths is  $O(\log n)$ , and since we push an element onto the stack only when discovering a new correct node, the total search time is  $O(\log n)$ ; see an example run in [3]. For the full pseudocode and the formal proof see [3].



■ **Figure 4** Comparing  $u$  to  $v$ : in Case 1, we know that  $u^*$  cannot be in  $u.r$ . We compare  $u'$  and  $v$  to decide how to proceed: in Case 1, we know that  $u^*$  cannot be in  $u'.r$ ; we go to  $u'.l$ ; in Case 2, we know that  $u^*$  cannot be in  $u.r$  and that  $v^*$  cannot be in  $v.l$ ; we go to  $u.l$  and to  $v.r$ ; in Case 3, we know that  $u^*$  is in  $u'.r$  (and hence in  $u.l$ ) or in  $v.l$ ; we go to  $u.l$  and to  $v.l$ .

### 3 Maintaining the union of unit discs under insertions

To maintain the union of unit discs under insertions, we maintain dynamic data structures for representing the boundary of the union, for reporting the arcs of the boundary that intersect with the next disc to be inserted, and for updating the boundary representation due to the insertion of the new disc. This section is dedicated to these data structures.

**Overview of the algorithm.** We denote by  $D(x)$  the unit disc centered at  $x$ . Let  $U$  be the union of  $n$  unit discs and let  $D(x)$  be the new unit disc, which we wish to insert. In order to report the arcs of  $\partial U$  that intersect  $D(x)$ , we overlay the plane with an implicit grid, where only cells that intersect with  $U$  are stored, and where the size of the diagonal of a grid cell is 1. The arcs of  $\partial U$  are divided into the cells of the grid – each arc of  $\partial U$  is associated with the cell that contains it. Note that if an arc belongs to more than one cell then we split it into (sub)arcs at the boundaries of the cells that it crosses (see an illustration of the structure in [3]). We divide the arcs of a given cell into four sets: *top*, *right*, *bottom* and *left*, which we denote by  $E_t$ ,  $E_r$ ,  $E_b$  and  $E_l$  respectively (see Section 3.1). The algorithm consists of the following main steps: (1) Find the cells that  $D(x)$  intersects. (2) For each such cell find the arcs of each one of the sets  $E_t$ ,  $E_r$ ,  $E_b$  and  $E_l$  that  $D(x)$  intersects. Cells of the union that contain no boundary arcs are treated in a special way. (3) Update  $\partial U$  using the arcs we found in the previous step and with  $\partial D(x)$ .

Step 1 of the algorithm is implemented using a balanced binary tree  $\Omega$  on the *active cells*, namely cells that have non-empty intersection with the current union  $U$ . The key of each active cell is the pair of coordinates of its bottom left corner. The active cells are stored at the leaves of the tree in ascending lexicographic order. Finding the cells intersected by a new disc, inserting or deleting a cell, take  $O(\log n)$  time each. For details, see, e.g., [13]. As we will see below, the structure  $\Omega$  will also be used to decide whether a new disc is fully contained in the current union or lies completely outside the current union (Section 3.3).

Most of this section is dedicated to a description of Steps 2 and 3 of the algorithm for the set  $E_t$ . The sets  $E_r$ ,  $E_b$ , and  $E_l$  can be handled in a similar manner. The basic property that we use is that  $D(x)$  intersects an arc  $e$  if and only if  $x$  belongs to  $e \oplus D_1$ , namely the Minkowski sum of  $e$  with a unit disc.

We split the boundaries of the Minkowski sums of  $E_t$  into upper and lower curves at the  $x$ -extremal points; in what follows, we refer to them as upper and lower curves, and denote their respective sets by  $\Gamma^+$  and  $\Gamma^-$ . (For clarity, we will refer to portions of the boundary of the union as *arcs* and to portions of the boundary of the Minkowski sums as *curves*.) The disc  $D(x)$  intersects the arc  $e \in E_t$  if and only if  $x$  lies above the lower curve induced by  $e$  and below the upper curve induced by  $e$ . We will store the curves of  $\Gamma^+$  in a dynamic structure  $\Delta^+$  and the curves of  $\Gamma^-$  in a dynamic structure  $\Delta^-$  (both described in Section 3.2).

Another property that we use is the following (see Lemma 12 below): Let  $\ell$  be a vertical line that passes through  $x$ , the center of the new disc. Then the intersection points of curves in  $\Gamma^+$  with  $\ell$  are all above the intersection points of curves of  $\Gamma^-$  with  $\ell$ .

Assume for the sake of exposition that we are given the point  $\xi$  of intersection between  $\ell$  and the upper envelope of the curves in  $\Gamma^-$ . If the center  $x$  of our new disc is above  $\xi$  then, since  $x$  is above all the lower curves that cross  $\ell$  we only need to search the structure  $\Delta^+$  for the upper curves that lie above  $x$  – these will determine the arcs of  $E_t$  that are intersected by  $D(x)$ . If the point  $x$  coincides with or lies below  $\xi$  then we only need to search the structure  $\Delta^-$  for the lower curves that lie below  $x$  – now these will determine the arcs of  $E_t$  that are intersected by  $D(x)$ .

However, we cannot easily obtain the point  $\xi$ , and hence querying the data structures is a little more involved: We use  $\Delta^+$  to iterate over the upper curves that lie above  $x$ . For every upper curve we check in  $O(1)$  time whether its corresponding arc (of  $E_t$ ) intersects with  $D(x)$ . If it intersects then we add this arc to the output list and continue to the next upper curve. If all the upper curves above  $x$  turn out to be induced by arcs intersecting  $D(x)$  we output this list of arcs and stop.

If all the reported arcs from the query of  $\Delta^+$  indeed intersect  $D(x)$ , then we are guaranteed that  $x$  is above  $\xi$  and this is the complete answer. Due to Lemma 12, if we detect that the arc induced by a curve reported by  $\Delta^+$  to lie above  $x$  is not intersecting  $D(x)$ , then we are guaranteed that  $x$  is on or below  $\xi$  and we will obtain the full reply by querying  $\Delta^-$ .

We review the geometric foundations needed by our algorithms and data structures in Section 3.1, then describe the data structures in Section 3.2. Finally, in Section 3.3 we explain how we solve our motivating problem – dynamically reporting the area of the union.

### 3.1 Preliminaries

Let  $B$  be an axis-parallel square, which represents one grid cell with unit-length diagonal, and let  $\ell_1$  and  $\ell_2$  be lines that support the diagonals of  $B$ . These lines divide the plane into top, right, bottom and left quadrants, which we denote by  $Q_t$ ,  $Q_r$ ,  $Q_b$  and  $Q_l$ , respectively.

Let  $U$  be the union of  $n$  unit discs. We divide the arcs of  $\partial U$  that are contained in  $B$  into four sets according to the quadrant which contains their centers. In case that a center lies on one of the lines then it is added either to the top or to the bottom quadrant. Denote these four sets of arcs by  $E_t$ ,  $E_r$ ,  $E_b$  and  $E_l$ . The power of this subdivision into quadrants is that now the projections of the arcs in any one set onto a major axis (the  $x$ -axis for  $E_t$  or  $E_b$ , and the  $y$ -axis for  $E_l$  or  $E_r$ ), are pairwise interior disjoint. For example,  $E_t$  contains the arcs whose centers are located in  $Q_t$ , and the projections of the arcs in  $E_t$  onto the  $x$ -axis are pairwise interior disjoint, as we show below in Lemma 6.

► **Definition 4.** For two bounded  $x$ -monotone arcs  $e_i$  and  $e_j$  we write  $e_i \leq_x e_j$  if and only if the right endpoint of  $e_i$  is to the left of or coincides with the left endpoint of  $e_j$ .

► **Lemma 5.** Each arc in  $E_t$  is portion of a lower semicircle.

► **Lemma 6.** The  $x$ -projections of the (relative interiors of) arcs in  $E_t$  are pairwise disjoint.

Relying on Lemma 6, henceforth we assume that the arcs in  $E_t$  are ordered from left to right:  $e_1, \dots, e_m$ . We wish to find which arcs of the set  $E_t$  intersect with the new unit disc  $D(x)$  to be inserted. For this purpose, we compute the Minkowski sum of each arc  $e_i$  of  $E_t$  with a unit disc centred at the origin. Then, we divide the boundary of each Minkowski sum into upper and lower curves at the  $x$ -extremal points: denote the top curve by  $\gamma_i^+$  and the bottom curve by  $\gamma_i^-$ . We denote the set of the upper curves,  $\{\gamma_i^+ | e_i \in E_t\}$ , by  $\Gamma^+$  and the set of the lower curves,  $\{\gamma_i^- | e_i \in E_t\}$ , by  $\Gamma^-$ . In the rest of this section we prove some useful properties regarding the curves in  $\Gamma^+$  and  $\Gamma^-$ :

**P1** Every lower curve in  $\Gamma^-$  can appear at most once on the lower envelope of the curves in  $\Gamma^-$ . Furthermore, if  $\gamma_i^-$  and  $\gamma_j^-$  appear on the lower envelope then  $\gamma_i^-$  appears to the left of  $\gamma_j^-$  if and only if  $e_i <_x e_j$ .

**P2** Let  $e_i, e_{i+1}$  and  $e_{i+2}$  be an ordered sequence of arcs in  $E_t$  and  $q$  be a point. If  $q$  lies below  $\gamma_i^+$  and  $\gamma_{i+2}^+$  then  $q$  lies also below  $\gamma_{i+1}^+$ .

**P3** For every vertical line  $\ell$ . The intersection points of the lower curves with  $\ell$  are below the intersection points of the upper curves with  $\ell$ .

## 26:10 Maintaining the Union of Unit Discs

In order to prove Property **P1**, we first need to show that every pair of lower curves intersect at most once.

► **Lemma 7.** *Let  $e_i$  and  $e_j$  be arcs of  $E_t$ . Then  $\gamma_i^-$  and  $\gamma_j^-$  intersect in exactly one point.*

For two  $x$ -monotone curves  $\ell_1, \ell_2$  that intersect exactly once, we say that  $\ell_1 < \ell_2$  when  $\ell_1$  appears on their joint lower envelope immediately to the left of their intersection point and  $\ell_2 < \ell_1$  otherwise. The proof of Lemma 7 also implies,

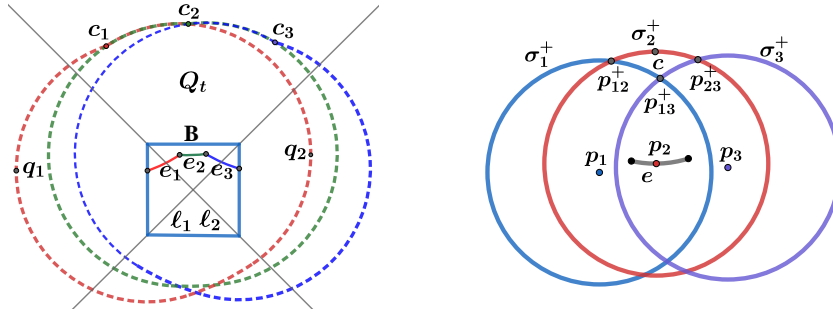
► **Corollary 8.** *For any pair of curves  $\gamma_i^-, \gamma_j^- \in \Gamma^-$ ,  $\gamma_i^- < \gamma_j^-$  if and only if  $e_i <_x e_j$ .*

We now turn to discuss the upper curves. To prove Property **P2** (Lemma 10), we first consider the structure of the upper envelope of the upper curves.

► **Observation 9.** *Let  $p$  and  $q$  be the endpoints of the arc  $e_i$  in  $E_t$ . The upper curve  $\gamma_i^+$  is the upper envelope of the upper boundaries (namely, semicircles) of the discs  $D(p)$  and  $D(q)$*

► **Lemma 10.** *Let  $e_i, e_{i+1}$  and  $e_{i+2}$  be an ordered sequence of arcs in  $E_t$  and  $q$  be a point. If  $q$  is below  $\gamma_i^+$  and  $\gamma_{i+2}^+$  then  $q$  is also below  $\gamma_{i+1}^+$ .*

**Proof.** Let  $p_1, p_2, p_3$  be points on arcs that belong to  $E_t$  with  $p_1 <_x p_2 <_x p_3$ . Let  $\sigma_1^+, \sigma_2^+$  and  $\sigma_3^+$  be the upper semicircles of  $\partial D(p_1), \partial D(p_2)$  and  $\partial D(p_3)$ , respectively. Let  $p_{12}^+$  and  $p_{23}^+$  be the intersection points of  $\sigma_1^+ \cap \sigma_2^+$  and  $\sigma_2^+ \cap \sigma_3^+$ , respectively. These intersection points exist, since the distance between every pair of points in  $B$  is at most one. By the assumption,  $p_1 <_x p_2$ , which means that  $\sigma_1^+$  appears to the left of  $\sigma_2^+$  on the upper envelope of  $\sigma_1^+$  and  $\sigma_2^+$ . Let  $c$  be the center of the arc  $e$  of  $E_t$  on which  $p_2$  lies. The point  $c$  is on  $\sigma_2^+$ , since  $p_2$  belongs to a lower semicircle of radius 1. In addition,  $c$  is not below  $\sigma_1^+$  since otherwise  $p_1 \in D(c)$  which would contradict that  $p_1$  is a point on an arc in  $E_t$ . This means that  $p_{12}^+ \leq_x c$ . The same argument implies that  $p_{23}^+ \geq_x c$  and therefore  $p_{12}^+ \leq_x p_{23}^+$ . This in turn implies that the intersection point,  $p_{13}^+$ , between  $\sigma_1^+$  and  $\sigma_3^+$  is below or on  $\sigma_2^+$  and therefore every point that lies below  $\sigma_1^+$  and  $\sigma_3^+$  lies below  $\sigma_2^+$ . The only condition on  $p_1, p_2$  and  $p_3$  is that they will be  $x$ -ordered.  $e_i \leq_x e_{i+1} \leq_x e_{i+2}$ , so the claim holds (see Figure 5). ◀



■ **Figure 5** (On the left). An example of  $\partial U \cap B$ .  $e_1, e_2$  and  $e_3$  are the arcs of  $E_t$  whose centers are  $c_1, c_2$  and  $c_3$ , respectively. The red, green and blue outer shapes are the boundary of the Minkowski sums of each of  $e_1, e_2$  and  $e_3$  with a disc of radius 1, respectively.  $\gamma_1^+$  and  $\gamma_1^-$  are denoted by the upper and lower red curves whose endpoints are  $q_1$  and  $q_2$ , respectively. (On the right) Illustration of the proof of Lemma 10.

For  $p$  an endpoint of  $e_i \in E_t$ , we call the upper semi-circle of the disc  $D(p)$  the upper curve of  $p$ . We denote the upper envelope of the curves in  $\Gamma^+$  by  $\mathcal{U}(\Gamma^+)$ . Note that some of the upper curves may appear on  $\mathcal{U}(\Gamma^+)$  as a single point, namely, they coincide with one of the breakpoints of  $\mathcal{U}(\Gamma^+)$ . The following corollary stems from the proof of Lemma 10.

► **Corollary 11.** (i) The upper curve of each endpoint of every arc of  $E_t$  appears on  $\mathcal{U}(\Gamma^+)$ .  
(ii) The  $x$ -order of the curves on  $\mathcal{U}(\Gamma^+)$  corresponds to the  $x$ -order of the endpoints of the arcs of  $E_t$ .

Next, we prove that for any pair of arcs  $e_i, e_j \in E_t$ ,  $\gamma_i^+$  and  $\gamma_j^-$  are disjoint. Furthermore, we show that  $\gamma_i^+$  is above  $\gamma_j^-$ , and by that prove Property P3.

► **Lemma 12.** Let  $e_i$  and  $e_j$  be two distinct arcs in  $E_t$  and let  $\ell$  be a vertical line. If  $\ell$  intersects with  $\gamma_i^+$  and  $\gamma_j^-$  at  $p$  and  $q$ , respectively, then  $p >_y q$ .

## 3.2 Data structures

In this section we describe two data structures. The data structure  $\Delta^+$  (resp.  $\Delta^-$ ), dynamically maintains the set  $\Gamma^+$  of the upper curves (resp.  $\Gamma^-$  of lower curves). The purpose of these structures is to efficiently answer the following queries: given a point  $x$ , report on the upper (resp. lower) curves which are above (resp. below)  $x$ . For the structure  $\Delta^+$  it is required that we get the answer gradually, one curve after the other, since we need to test each curve for being relevant (in addition to being above  $x$ ), and stop as soon as we detect the first irrelevant curve.

### 3.2.1 Dynamically maintaining the lower curves

For maintaining the lower curves  $\Gamma^-$  induced by the arcs in  $E_t$ , we implement  $\Delta^-$  using the data structure described in Section 2. Recall that the data structure dynamically maintains a set of curves fulfilling property P1 and supports the following **query**: given a point  $x$  report the curves in  $\Gamma^-$  that are below  $x$ .

**Update.** After we insert a new unit disc we may have to delete and insert many lower curves. If a lower curve  $\gamma_i^-$  is split into subcurves, then we delete  $\gamma_i^-$  and create two new subcurves instead. In order for Property P1 to hold at all times, we first delete the old lower curves from  $\Delta^-$  and then insert the new ones.

### 3.2.2 Dynamically maintaining the upper curves

**Description.** Let  $p_1, p_2, \dots, p_r$  be the endpoints of the arcs of  $E_t$  sorted in ascending  $x$ -order. Recall that  $\mathcal{U}(\Gamma^+)$  denotes the upper envelope of  $\Gamma^+$ . Let  $s_1, s_2, \dots, s_r$  be the arcs of  $\mathcal{U}(\Gamma^+)$  ordered from left to right. Note that each endpoint of  $E_t$  corresponds to an arc in  $\mathcal{U}(\Gamma^+)$ , i.e.,  $p_i$  corresponds to the curve  $s_i$ . The data structure  $\Delta^+$  is a balanced binary search tree. We store the points  $p_i$  in the leaves of the tree in their left-to-right order. We also store in each leaf pointers **rn** and **ln** to its right and left neighboring leaves respectively, if exist. Each internal node stores a pointer **lml** to the leftmost leaf of its right subtree. To keep the a structure simple, if two arcs of  $E_t$  meet at a single point, we keep only one of the endpoints incident to this point in the list  $\{p_i\}$ . However, we mark in the leaf of  $p_i$  which are the two arcs incident to it. Below, when we traverse the leaves of the tree and test the respective arcs of  $E_t$  for intersection with the new disc, in some nodes we may need to test two arcs.

**Query.** Let  $q$  be a query point. By following a path from the root, we first find the leaf  $v$  such that the vertical line through  $p$  intersects the edge  $s_v$ . The search down the tree is carried out as follows. Suppose we reached an internal node  $u$ . We use the pointer **lml**( $u$ ) to obtain the leaf  $w$ , and use **ln**( $w$ ) to find the point immediately to its left in the sequence

## 26:12 Maintaining the Union of Unit Discs

$\{p_i\}$ . These two points will determine the breakpoint of  $\mathcal{U}(\Gamma^+)$  that separates between the left and right portions of the envelope, which are represented by the subtrees rooted at the left and right children of  $u$ .

Recall that the structure  $\Delta^+$  plays the extra role of deciding whether the center  $x$  of the new disc lies above the point  $\xi$  or not (see the overview the algorithm in the beginning of Section 3). Therefore the query process is somewhat more involved than if we used the structure only to determine which curves of  $\Gamma^+$  pass above  $x$ .

Once we find the point  $p_i$  whose arc  $s_i$  of the envelope intersects the vertical line through the query point  $q$ , we will be traversing leaves of  $\Delta^+$  starting at  $v$  going both rightward and leftward. At each leaf  $u$  we test whether  $q$  lies below the curve  $s_j$  stored at  $u$  and if so, we check whether  $D(x)$  intersects the relevant arc of  $E_t$ . If the answer to both predicates is true then we continue the search in the same direction. If while we search rightwards the first predicate is false then we go leftwards starting from  $v$ . If the first predicate is false and we search leftwards then we stop the search and report on the arcs that we found. If the first predicate is true and second predicate is false then we continue with  $\Delta^-$ .

**Update.** After we insert a new disc, many arcs may be deleted from  $E_t$  and many new arcs may be inserted into  $E_t$ . We simply remove the endpoints of the deleted arcs and insert the endpoints of the new arcs into  $\Delta^+$ .

The correctness of the query procedure follows from Lemma 10. The performance of the structure is summarized in the following lemma whose proof is straightforward.

► **Lemma 13.** *The query time of the data structure is  $O(\log n + k)$ , where  $k$  is the number of reported arcs. The update requires  $O(\log n)$  time per operation.*

When querying the data structures  $\Delta^+$  and  $\Delta^-$  we obtain the set  $I$  of arcs of the existing union-boundary that need to be deleted or partially cut since they are covered by the new disc  $D(x)$  to be inserted. However, we also need to update the structures with the arcs that the boundary of the new disc contributes to the union boundary.

To find which portions of  $\partial D(x)$  appear on the boundary of the union  $U \cup D(x)$ , we construct the arrangement  $\mathcal{A}(I \cup \partial D(x))$  and look for faces of this arrangement that abut  $\partial D(x)$  and are not in the union  $U$ . One can show that in a face  $f$  of this type the arcs of  $\partial U$  appear on it as concave, meaning that any point inside this face is outside the disc bounded by the arcs. Denote the size of  $I$  by  $k$ . Assume first that  $k \geq 1$ . We can construct the arrangement in  $O(k \log k)$  time (recall that the arcs in  $I \cup \partial D(x)$  are pairwise interior disjoint). Finding the arcs of  $\partial D(x)$  that should be inserted takes another  $O(k)$  time.

If  $k = 0$ , there are two cases based on whether  $D(x) \cap U$  is (i)  $D(x)$  or (ii) the empty set. To distinguish between the cases we need to either (i) find a point that belongs to  $D(x)$  and  $U$ , or (ii) a point that belongs to  $D(x)$  but not to  $U$ . Recall that in order to find  $I$  we overlay the plane with a grid of cells of unit-length diagonal each. This implies that at least one of the cells, denoted by  $\omega$ , is fully contained in  $D(x)$ . If  $\omega$  is an *active cell*, i.e.,  $\omega \cap U \neq \emptyset$ , then  $\omega$  is fully contained in  $U$  ( $I$  is an empty set) and therefore  $D(x) \cap U = D(x)$ ; otherwise  $D_1(x) \cap U = \emptyset$ . To check whether  $\omega$  is active, we search for it in the structure  $\Omega$ . In case (i) we do nothing further, and in case (ii) we make all the grid cells covered by  $D(x)$  active, and we update the data structures of each grid cell crossed by  $\partial D(x)$  by the relevant portions of  $\partial D(x)$ . To conclude,

► **Theorem 14.** *The boundary arcs of the union of a set of  $n$  unit discs can be maintained under insertion in a data structure of  $O(n)$  size so that a new disc can be inserted in  $O(k \log^2 n)$  time, where  $k$  is the total number of changes on the boundary of the union.*



### 3.3 Maintaining the area of the union

We are now ready to solve our motivating problem, namely dynamically reporting the *area* of the union as we insert discs. At a high level our algorithm proceeds as follows:

1. Find the set  $I$  of the arcs on the boundary of the union  $U$  that intersect with the new disc  $D(x)$  to be inserted.
2. Compute the arrangement  $\mathcal{A}(I \cup \partial D(x))$ .
3. Calculate the extra area (over the area of the union before inserting  $D(x)$ ) that  $D(x)$  covers, using  $\mathcal{A}(I \cup \partial D(x))$ .

In order to find  $I$  we make use of the data structures described above and summarized in Theorem 14. Let  $k$  denote the number of arcs in  $I$  and assume that  $k \geq 1$ . We use a sweep-line algorithm to compute the arrangement  $\mathcal{A}(I \cup \partial D(x))$  in time  $O(k \log k)$ . To calculate the extra area that  $D(x)$  covers, we go over the faces of the arrangement and sum up the area of the faces that are contained in  $D(x) \setminus U$ . If  $k = 0$  then either the disc is fully contained in the current union (see above for how to determine this), in which case we do nothing, or it is disjoint from the union before the insertion of the disc, in which case we increase the area by  $\pi$ . To conclude,

► **Theorem 15.** *Given a sequence of  $n$  unit discs in  $\mathbb{R}^2$  to be inserted one after the other, reporting the area of the union of the discs after each insertion can be carried out in  $O(k \log^2 n)$  time and  $O(n)$  space, where  $k$  is the total number of structural changes to the boundary of the union incurred by the insertion of the new disc.*

## 4 Intersection-searching of unit arcs with unit disc

In this section we address the following intersection-searching problem: Preprocess a collection  $\mathcal{C}$  of circular arcs of unit radius into a data structure so that for a query unit disc  $D(x)$ , centered at the point  $x$ , the arcs in  $\mathcal{C}$  intersecting  $D(x)$  can be reported efficiently. We assume for simplicity that every arc in  $\mathcal{C}$  belong to the lower semicircle.

Let  $e \in \mathcal{C}$  be a unit-radius circular arc, and let  $p_1$  and  $p_2$  be its endpoints. A unit disc  $D(x)$  intersects  $e$  if and only if  $e \oplus D(0)$ , the Minkowski sum of  $e$  with a unit disc, contains the center  $x$ . Let  $z := D(p_1) \cup D(p_2)$ , and let  $D^+(c)$  be the disk of radius 2 centered at  $c$ ;  $z$  divides  $D^+(c)$  into three regions (see Fig. 6): (i)  $z^+$ , the portion of  $D^+(c) \setminus z$  above  $z$ , (ii)  $z$  itself, and (iii)  $z^-$ , the portion of  $D^+(c) \setminus z$  below  $z$ . It can be verified that  $e \oplus D(0) = z \cup z^-$ . We give an alternate characterization of  $z \cup z^-$ , which will help in developing the data structure.

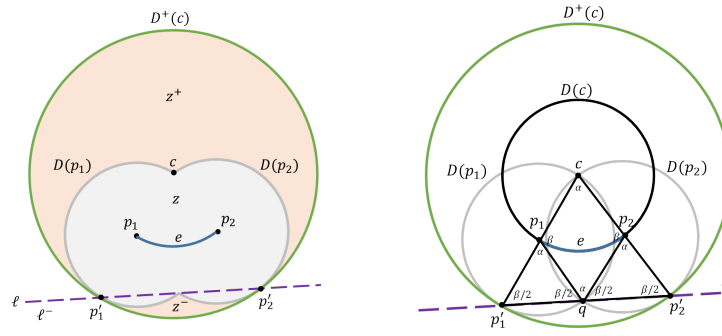
Let  $\ell$  be a line that passes through the tangents points,  $p'_1$  and  $p'_2$ , of  $D(p_1)$  and  $D(p_2)$  with  $D^+(c)$ , respectively, and let  $\ell^-$  be the halfplane lying below  $\ell$ . Set  $L(e) = D^+(c) \cap \ell^-$  (see Fig. 6).

► **Lemma 16.** *If  $\partial D(p_1)$  and  $\partial D(p_2)$  intersect at two points (one of which is always  $c$ ) then  $\ell$  passes through  $q := (\partial D(p_1) \cap \partial D(p_2)) \setminus \{c\}$ . Otherwise  $c \in \ell$ .*

**Proof.** Assume that  $q$  exists. The quadrilateral  $(c, p_1, q, p_2)$  is a rhombus since all its edges have length 1. Let  $\alpha$  be the angle  $\angle p_1 q p_2$  and  $\beta$  be the angle  $\angle c p_1 q$ . The angle  $\angle q p_1 p'_1$  is equal to  $\alpha$  since the segment  $(c, p'_1)$  is a diameter of  $D(p_1)$ . The angle  $\angle p_1 q p'_1$  is equal to  $\frac{\beta}{2}$  since  $\triangle p_1 q p'_1$  is an isosceles triangle. The same arguments apply to the angle  $\angle p_2 q p'_2$  implying that the angle  $\angle p'_1 q p'_2$  is equal to  $\pi$ .

Assume that  $q$  does not exist then the segment  $(p_1, p_2)$  is a diameter of  $D(c)$ . The segment  $(c, p'_1)$  is a diameter of  $D(p_1)$ . The segment  $(p_1, p_2)$  coincide with  $(c, p'_1)$  at the segment  $(c, p_1)$ . The same argument applies to the segment  $(c, p'_2)$ , implying that the angle  $\angle p'_1 q p'_2$  is equal to  $\pi$  (see Fig 6). ◀





■ **Figure 6** (On the left) Partition of  $D_2(c)$  into three regions:  $z^+$ ,  $z$  and  $z^-$ . (On the right) Illustration of Lemma 16.

The following corollary summarizes the criteria for the intersection of a unit circular arc with a unit disc.

► **Corollary 17.** *Let  $e$  be a circular arc in  $\mathcal{C}$  with endpoints  $p_1$  and  $p_2$  and center  $c$ . Then (i)  $z \cup z^- = z \cup L(e)$ . (ii)  $e$  intersects a unit disc  $D(x)$  if and only if at least one of the following conditions is satisfied: (a)  $x \in D(p_1)$  (or  $p_1 \in D(x)$ ), (b)  $x \in D(p_2)$  (or  $p_2 \in D(x)$ ), and (c)  $x \in L(e)$ .*

We thus construct three separate data structures. The first data structure preprocesses the left endpoints of arcs in  $\mathcal{C}$  for unit-disc range searching, the second data structure preprocesses the right endpoints of arcs in  $\mathcal{C}$  for unit-disc range searching, and the third data structure preprocesses  $\mathcal{L} = \{L(e) \mid e \in \mathcal{C}\}$  for inverse range searching, i.e., reporting all regions in  $\mathcal{L}$  that contain a query point. Using standard circle range searching data structures (see e.g. [4, 2]), we can build these three data structures so that each of them takes  $O(n)$  space and answers a query in  $O(n^{1/2+\epsilon} + k)$  time, where  $k$  is the output size. Furthermore, these data structures can handle insertions/deletions in  $O(\log^2 n)$  time. We omit all the details from here and conclude the following:

► **Theorem 18.** *Let  $\mathcal{C}$  be a set of  $n$  unit-circle arcs in  $\mathbb{R}^2$ .  $\mathcal{C}$  can be preprocessed into a data structure of linear size so that for a query unit disk  $D$ , all arcs of  $\mathcal{C}$  intersecting  $D$  can be reported in  $O(n^{1/2+\epsilon} + k)$  time, where  $\epsilon$  is an arbitrarily small constant and  $k$  is the output size. Furthermore the data structure can be updated under insertion/deletion of a unit-circle arc in  $O(\log^2 n)$  amortized time.*

---

**References**

- 1 Pankaj K. Agarwal. Range searching. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 40. Chapman & Hall/CRC, 3rd edition, 2017.
- 2 Pankaj K. Agarwal. *Simplex Range Searching and Its Variants: A Review*, pages 1–30. Springer International Publishing, Cham, 2017.
- 3 Pankaj K. Agarwal, Ravid Cohen, Dan Halperin, and Wolfgang Mulzer. Maintaining the Union of Unit Discs under Insertions with Near-Optimal Overhead, 2019. [arXiv:1903.10943v1](https://arxiv.org/abs/1903.10943v1).
- 4 Pankaj. K. Agarwal and Jiří Matoušek. On range searching with semialgebraic sets. *Discrete & Computational Geometry*, 11(4):393–418, 1994.
- 5 Pankaj K. Agarwal and Jiří Matoušek. Dynamic Half-Space Range Reporting and Its Applications. *Algorithmica*, 13(4):325–345, 1995.

- 6 F. Aurenhammer. Improved algorithms for discs and balls using power diagrams. *Journal of Algorithms*, 9(2):151–161, 1988. doi:10.1016/0196-6774(88)90035-1.
- 7 Gerth Stølting Brodal and Riko Jacob. Dynamic Planar Convex Hull with Optimal Query Time. In *Algorithm Theory - SWAT 2000, 7th Scandinavian Workshop on Algorithm Theory, Bergen, Norway, July 5-7, 2000, Proceedings*, pages 57–70, 2000.
- 8 Gerth Stølting Brodal and Riko Jacob. Dynamic Planar Convex Hull. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, pages 617–626, 2002.
- 9 Timothy M. Chan. Dynamic planar convex hull operations in near-logarithmic amortized time. *J. ACM*, 48(1):1–12, 2001.
- 10 Timothy M. Chan. A dynamic data structure for 3-D convex hulls and 2-D nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010.
- 11 Ravid Cohen, Dan Halperin, and Yossi Yovel. Sensory Regimes of Effective Distributed Searching without Leaders. Manuscript, 2018.
- 12 Mark de Berg, Kevin Buchin, Bart MP Jansen, and Gerhard Woeginger. Fine-grained complexity analysis of two classic TSP variants. *arXiv preprint*, 2016. arXiv:1607.02725.
- 13 Dan Halperin and Mark H. Overmars. Spheres, molecules, and hidden surface removal. *Comput. Geom.*, 11(2):83–102, 1998.
- 14 Dan Halperin and Micha Sharir. Arrangements. In Jacob E. Goodman, Joseph O’Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 28. Chapman & Hall/CRC, 3rd edition, 2017.
- 15 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic Planar Voronoi Diagrams for General Distance Functions and their Algorithmic Applications. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2495–2504, 2017.
- 16 Haim Kaplan, Robert Endre Tarjan, and Kostas Tsioutsoulis. Faster kinetic heaps and their use in broadcast scheduling. In *Proceedings of the 12th Annual Symposium on Discrete Algorithms*, pages 836–844, 2001.
- 17 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the Union of Jordan Regions and Collision-Free Translational Motion Amidst Polygonal Obstacles. *Discrete & Computational Geometry*, 1:59–70, 1986.
- 18 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981.
- 19 Franco P. Preparata and Michael Ian Shamos. *Computational Geometry. An Introduction*. Springer-Verlag, New York, 1985.
- 20 Paul G Spirakis. Very fast algorithms for the area of the union of many circles. *Report no. 98 – Dept. Computer Science, Courant Institute, New York University*, 1983.



# Almost Tight Lower Bounds for Hard Cutting Problems in Embedded Graphs

Vincent Cohen-Addad

Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France  
vcohenad@gmail.com

Éric Colin de Verdière

Université Paris-Est, LIGM, CNRS, ENPC, ESIEE Paris, UPEM, Marne-la-Vallée, France  
eric.colindeverdiere@u-pem.fr

Dániel Marx

Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI),  
Budapest, Hungary  
dmarx@cs.bme.hu

Arnaud de Mesmay

Univ. Grenoble Alpes, CNRS, Grenoble INP<sup>1</sup>, GIPSA-lab, 38000 Grenoble, France  
arnaud.de-mesmay@gipsa-lab.fr

---

## Abstract

We prove essentially tight lower bounds, conditionally to the Exponential Time Hypothesis, for two fundamental but seemingly very different cutting problems on surface-embedded graphs: the SHORTEST CUT GRAPH problem and the MULTIWAY CUT problem.

A cut graph of a graph  $G$  embedded on a surface  $S$  is a subgraph of  $G$  whose removal from  $S$  leaves a disk. We consider the problem of deciding whether an unweighted graph embedded on a surface of genus  $g$  has a cut graph of length at most a given value. We prove a time lower bound for this problem of  $n^{\Omega(g/\log g)}$  conditionally to ETH. In other words, the first  $n^{O(g)}$ -time algorithm by Erickson and Har-Peled [SoCG 2002, Discr. Comput. Geom. 2004] is essentially optimal. We also prove that the problem is W[1]-hard when parameterized by the genus, answering a 17-year old question of these authors.

A multiway cut of an undirected graph  $G$  with  $t$  distinguished vertices, called *terminals*, is a set of edges whose removal disconnects all pairs of terminals. We consider the problem of deciding whether an unweighted graph  $G$  has a multiway cut of weight at most a given value. We prove a time lower bound for this problem of  $n^{\Omega(\sqrt{gt+g^2}/\log(gt))}$ , conditionally to ETH, for any choice of the genus  $g \geq 0$  of the graph and the number of terminals  $t \geq 4$ . In other words, the algorithm by the second author [Algorithmica 2017] (for the more general multicut problem) is essentially optimal; this extends the lower bound by the third author [ICALP 2012] (for the planar case).

Reductions to planar problems usually involve a grid-like structure. The main novel idea for our results is to understand what structures instead of grids are needed if we want to exploit optimally a certain value  $g$  of the genus.

**2012 ACM Subject Classification** Mathematics of computing → Graphs and surfaces; Mathematics of computing → Graph algorithms

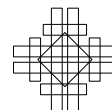
**Keywords and phrases** Cut graph, Multiway cut, Surface, Lower bound, Parameterized Complexity, Exponential Time Hypothesis

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.27

**Funding** *Vincent Cohen-Addad*: Ce projet a bénéficié d'une aide de l'État gérée par l'Agence Nationale de la Recherche au titre du Programme FOCAL portant la référence suivante : ANR-18-CE40-0004-01.

---

<sup>1</sup> Institute of Engineering Univ. Grenoble Alpes



*Éric Colin de Verdière*: Partially supported by the French ANR project ANR-17-CE40-0033 (SoS).  
*Dániel Marx*: Supported by ERC Consolidator Grant SYSTEMATICGRAPH (No. 725978).  
*Arnaud de Mesmay*: Partially supported by the French ANR projects ANR-16-CE40-0009-01 (GATO), ANR-18-CE40-0004-01 (FOCAL) and the CNRS PEPS project COMP3D.

**Acknowledgements** We are grateful to the anonymous referees for a careful reading of the paper and many helpful suggestions.

## 1 Introduction

During the past decade, there has been a flurry of works investigating the complexity of solving exactly optimization problems on planar graphs, leading to what was coined as the “square root phenomenon” by the third author [27]: many problems turn out to be easier on planar graphs, and the improvement compared to the general case is captured exactly by a square root. For instance, problems solvable in time  $2^{O(n)}$  in general graphs can be solved in time  $2^{O(\sqrt{n})}$  in planar graphs, and similarly, in a parameterized setting, FPT problems admitting  $2^{O(k)}n^{O(1)}$ -time algorithms or W[1]-hard problems admitting  $n^{O(k)}$ -time algorithms can often be sped up to  $2^{\tilde{O}(\sqrt{k})}n^{O(1)}$  and  $n^{\tilde{O}(\sqrt{k})}$ , respectively, when restricted to planar graphs. We have many examples where matching upper bounds (algorithms) and lower bounds (complexity reductions) show that indeed the best possible running time for the problems has this form. On the side of upper bounds, the improvement often stems from the fact that planar graphs have (recursive) planar separators of size  $O(\sqrt{n})$ , and the theory of bidimensionality provides an elegant framework for a similar speedup in the parameterized setting for some problems [12]. However, in many cases these algorithms rely on highly problem-specific arguments [6, 28, 21, 30, 2, 23, 14]. The lower bounds are conditional to the Exponential Time Hypothesis (ETH) of Impagliazzo, Paturi, and Zane [19] and follow from careful reductions from problems displaying this phenomenon, e.g., PLANAR 3-COLORING,  $k$ -CLIQUE, or GRID TILING. We refer to the recent book [9] for precise results along these lines.

While the theme of generalizing algorithms from planar graphs to surface-embedded graphs has attracted a lot of attention, and has flourished into an established field mixing algorithmic and topological techniques (see [7]), the same cannot be said at all of the lower bounds. Actually, up to our knowledge, there are very few works explicitly establishing algorithmic lower bounds based on the genus of the surfaces on which a graph is embedded, or even just hardness results when parameterized by the genus – the only ones we are aware of are the exhaustive treatise [29] of the third author and Pilipczuk on SUBGRAPH ISOMORPHISM, where some of the hardness results feature the genus of the graph, the lower bounds of Curticapean and the third author [8] on the problem of counting perfect matchings and the work of Chen et al. [1].

In this work, we address this surprising gap by providing lower bounds conditioned on ETH for two fundamental yet seemingly very different cutting problems on surface-embedded graphs: the SHORTEST CUT GRAPH problem and the MULTIWAY CUT problem. In both cases, our lower bounds match the best known algorithms up to a logarithmic factor in the exponent. We believe that the tools that we develop in this paper could pave the way towards establishing lower bounds for other problems on surface-embedded graphs.

**The shortest cut graph problem.** A *cut graph* of an edge-weighted graph  $G$  cellularly embedded on a surface  $S$  is a subgraph of  $G$  that has a unique face, which is a disk. Computing a shortest cut graph is a fundamental problem in algorithm design, as it is often

easier to work with a planar graph than with a graph embedded on a surface of positive genus, since the large toolbox that has been designed for planar graphs becomes available. Furthermore, making a graph planar is useful for various purposes in computer graphics and mesh processing, see, e.g., [34]. Be it for a practical or a theoretical goal, a natural measure of the distortion induced by the cutting step is the length of the topological decomposition.

Thus, the last decade has witnessed a lot of effort on how to obtain efficient algorithms for the problems of computing short topological decompositions, see for example the survey [7]. For the shortest cut graph problem, Erickson and Har-Peled [13] showed that the problem is NP-hard when the genus is considered part of the input and gave an exact algorithm running in time  $n^{O(g)}$ , where  $n$  is the size of the input graph and  $g$  the genus of the surface, together with an  $O(\log^2 g)$ -approximation running in time  $O(g^2 n \log n)$ . The first and fourth authors [5] gave a  $(1 + \varepsilon)$ -approximation algorithm running in time  $O(f(\varepsilon, g)n^3)$ , where  $f$  is some explicit computable function. Whether it is possible to improve upon the exact algorithm of Erickson and Har-Peled by designing an FPT algorithm for the problem, namely an exact algorithm running in time  $f(g)n^{O(1)}$ , has been raised by these authors [13, Conclusion] and has remained an open question over the last 17 years.

In this paper, we solve this question by proving that the result of Erickson and Har-Peled cannot be significantly improved. We indeed show a lower bound of  $n^{\Omega(g/\log g)}$  (for the associated decision problem, even in the unweighted case) assuming the Exponential Time Hypothesis (ETH) of Impagliazzo et al. [19] (see Definition 3), and also prove that the problem is W[1]-hard:

► **Theorem 1.** *Let us consider the SHORTEST CUT GRAPH problem: Given an unweighted graph  $G$  with  $n$  vertices cellularly embedded on an orientable surface of genus  $g$ , and an integer  $\lambda$ , decide whether  $G$  admits a cut graph of length at most  $\lambda$ .*

1. *This problem is W[1]-hard when parameterized by  $g$ .*
2. *Assuming ETH, there exists a universal constant  $\alpha_{CG}$  such that for any fixed integer  $g \geq 2$ , there is no algorithm solving all the SHORTEST CUT GRAPH instances of genus at most  $g$  in time  $O(n^{\alpha_{CG} \cdot g/\log g})$ .*

(In the second item, the constraint  $g \geq 2$  is just here to ensure that  $g/\log g$  is well-defined.)

**The multiway cut problem.** The second result of our paper concerns the MULTIWAY CUT problem (also known as the MULTITERMINAL CUT problem). Given an edge-weighted graph  $G$  together with a subset  $T$  of  $t$  vertices called terminals, a multiway cut is a set of edges whose removal disconnects all pairs of terminals. Computing a minimum-weight multicut is a classic problem that generalizes the minimum  $s - t$  cut problem and some closely related variants have been actively studied since as early as 1969 [18]. On general graphs, while the problem is polynomial-time solvable for  $t = 2$ , it becomes NP-hard for any fixed  $t \geq 3$ , see [10]. In the case of planar graphs, it remains NP-hard if  $t$  is arbitrarily large, but can be solved in time  $2^{O(t)} n^{O(\sqrt{t})}$ , where  $n$  is the number of vertices and edges of the graph [21], and a lower bound of  $n^{\Omega(\sqrt{t})}$  was proved (conditionally on ETH) by the third author [26]. A generalization to higher-genus graphs was recently obtained by the second author [6] who devised an algorithm running in time  $f(g, t) \cdot n^{O(\sqrt{gt+g^2})}$  in graphs of genus  $g$ , for some function  $f$  (actually, for the more general MULTICUT problem). If one allows some approximation, this can be significantly improved: three of the authors recently provided a  $(1 + \varepsilon)$ -approximation algorithm running in time  $f(\varepsilon, g, t) \cdot n \log n$  [3].

We prove a lower bound of  $n^{\Omega(\sqrt{gt+g^2}/\log(gt))}$  for the associated decision problem, even in the unweighted case, which almost matches the aforementioned best known upper bound, and generalizes the lower bound of the third author [26] for the planar case. Actually, we

prove a lower bound that holds for any value of the integers  $g$  and  $t$  as long as  $t \geq 4$ . The precise theorem is the following, where we use  $g^*$  to denote  $\max(2, g)$  so that the quantities are well-defined for  $g = 0$  and  $g = 1$ :

► **Theorem 2.** *Let us consider the MULTIWAY CUT problem: Given an unweighted graph  $G$ , a set  $T$  of vertices, and an integer  $\lambda$ , decide whether there exists a multiway cut of  $(G, T)$  of value at most  $\lambda$ .*

*Assuming ETH, there exists a universal constant  $\alpha_{\text{MC}}$  such that for any fixed choice of integers  $g \geq 0$  and  $t \geq 4$ , there is no algorithm that decides all the MULTIWAY CUT instances  $(G, T, \lambda)$  for which  $G$  is embeddable on the orientable surface of genus  $g$  and  $|T| \leq t$ , in time  $O(n^{\alpha_{\text{MC}} \sqrt{g^* t + g^{*2}} / \log(g^* t)})$ .*

Note that taking  $g = 0$  in this theorem yields lower bounds for the PLANAR MULTIWAY CUT problem, and recovers, up to a logarithmic factor, the lower bounds obtained by the third author [26] for that problem. In the opposite regime, we also prove W[1]-hardness with respect to the genus for instances with 4 terminals, see Proposition 11. We remark that  $t = 2$  corresponds to the minimum cut problem, which is polynomial-time solvable, so a lower bound on  $t$  is necessary. While the last remaining case, for  $t = 3$ , is known to be NP-hard [10], our techniques do not seem to encompass it, and we leave its parameterized complexity with respect to the genus as an open problem.

► **Remark.** Parameterized lower bounds in the literature often have the form “assuming ETH, there is no  $f(k)n^{o(h(k))}$  algorithm to solve problem X, for any function  $f$ ”, where  $h$  is some specific dependency on the parameter. The lower bounds that we prove in Theorems 1 and 2 are instead of the form “assuming ETH, there exists a universal constant  $\alpha$  such that for any fixed  $k$ , there is no  $O(n^{\alpha h(k)})$  algorithm to solve problem X”. The latter lower bounds imply the former: indeed,  $f(k)n^{o(h(k))} = O(n^{\alpha h(k)})$  for a fixed  $k$ . Our results are stronger, concerning instances for any fixed  $k$ . Moreover, lower bounds with two parameters are difficult to state with  $o()$  notation. The statement of Theorem 2 handles every combination of the two parameters in a completely formal way.

**Main ideas of the proof.** What is a good starting problem to prove hardness results for surface-embedded graphs? For planar graphs, the GRID TILING problem of the third author [24] has now emerged as a convenient, almost universal, tool to establish parameterized hardness results and precise lower bounds based on ETH. A similar approach, based on constraint satisfaction problems (CSPs) on  $d$ -dimensional grids, was used by the third author and Sidiropoulos [31] to obtain lower bounds for geometric problems on low-dimensional Euclidean inputs (see also [11] for a similar framework for geometric intersection graphs). However, these techniques do not apply directly for the problems that we consider. Indeed, the bounds implied by these approaches are governed by the treewidths of the underlying graphs and are of the type  $n^{\Omega(\sqrt{p})}$  or  $n^{\Omega(p^{1-1/d})}$  respectively, where  $p$  is the parameter of interest and  $d$  the dimension of the grid in the latter case. In contrast, here, we are looking for bounds of the form  $n^{\Omega^*(p)}$  (while this is not apparent from looking at Theorem 2, this also turns out to be the main regime of interest for the MULTIWAY CUT problem).

Our first contribution, in Section 3, is to introduce a new hard problem for embedded graphs, which is versatile enough to be used as a starting point to obtain lower bounds for both the SHORTEST CUT GRAPH and the MULTIWAY CUT problem (and hopefully others). It is a variant of the GRID TILING problem which we call 4-REGULAR GRAPH TILING; in a precise sense, it generalizes the GRID TILING problem to allow for embedded 4-regular graphs different from the planar grid to be used as the structure graph of the problem. We



show that a CSP instance with  $k$  binary constraints can be simulated by a 4-REGULAR GRAPH TILING instance with parameter  $k$ . A result of the third author [25] shows that, assuming the ETH, such CSP instances cannot be solved in time  $f(k)n^{\Omega(k/\log k)}$ , giving a similar lower bound for 4-REGULAR GRAPH TILING (Theorem 9).

We then establish in Sections 4 and 5 the lower bounds for the SHORTEST CUT GRAPH and “one half” of the lower bound for MULTIWAY CUT, namely, for the regime where the genus dominates the number of terminals. Both reductions proceed from 4-REGULAR GRAPH TILING and use as a building block an intricate set of *cross gadgets* originally designed by the third author [26] for his hardness proof of the PLANAR MULTIWAY CUT problem. While it does not come as a surprise that these gadgets are useful for more general non-planar MULTIWAY CUT instances, it turns out that via basic planar duality, they also provide exactly the needed technical tool for establishing the hardness of SHORTEST CUT GRAPH.

In order to establish the “second half” of the lower bound in Theorem 2, in the regime where the number of terminals dominates the genus, we use a similar strategy in Section 6 but bypass the use of the 4-REGULAR GRAPH TILING problem. Instead, we rely directly on the aforementioned theorem of the third author on the parameterized hardness of CSPs, which we apply not to a family of expanders, but to blow-ups of expanders, i.e., expanders where vertices are replaced by grids of a well-chosen size. This size is prescribed exactly by the tradeoff between the genus and the number of terminals, as described with the two integers  $g$  and  $t$  in Theorem 2. The key property of these blow-ups is that their treewidth is  $\text{tw} = \Theta(\sqrt{gt})$  and thus the  $n^{\Omega(\text{tw}/\log \text{tw})}$  lower bound on the complexity of CSPs with these blow-ups as primal graphs yields exactly the target lower bound. The reduction from CSPs to MULTIWAY CUT is carried out in Proposition 14 and also relies on cross gadgets.

There just remains to combine Propositions 11 and 14 to obtain Theorem 2. This is easy and we refer to the full version [4, Section 7] for the proof.

Note that while Theorem 2 does not use an embedded graph as an input, we can find an embedding of a graph on a surface with minimum possible genus in  $f(g)n$  time [20, 32]. Thus, the same hardness result holds in the embedded case and the question is not about whether we are given the embedding or not.

## 2 Preliminaries

**Graphs on surfaces.** For standard definitions for graphs embedded on surfaces, we refer to the classic textbook of Mohar and Thomassen [33] and the fullversion [4, Section 2]. In this article, all surfaces are compact, connected, and orientable.

**The Exponential Time Hypothesis.** Our lower bounds are conditioned on the Exponential Time Hypothesis (ETH), which was conjectured in [19].

► **Conjecture 3** (Exponential Time Hypothesis [19]). *There exists a positive real value  $s > 0$  such that 3-CNF-SAT, parameterized by  $n$ , has no  $2^{sn}(n+m)^{O(1)}$ -time algorithm (where  $n$  denotes the number of variables and  $m$  denotes the number of clauses).*

We refer to the survey [22] for background and discussion of this conjecture.

**Expanders and their treewidth.** We will rely on the following classical lemmas about expander graphs and their treewidth. A family  $\mathcal{G}$  of graphs is *dense* if for any  $n > 0$ , there exists a graph in  $\mathcal{G}$  with  $\Theta(n)$  vertices (where the  $\Theta()$  hides a universal constant). The other definitions and the proofs are included in the full version [4, Section 2]



► **Lemma 4.** *There exists a dense family  $\mathcal{H}$  of bipartite four-regular expanders.*

► **Lemma 5.** *Every  $d$ -regular graph  $G$  satisfies  $\text{tw}(G) \geq \lfloor |V(G)| \cdot (1 - \lambda(G)/d)/8 \rfloor$ .*

**Constraint satisfaction problems.** A *binary constraint satisfaction problem* is a triple  $(V, D, C)$  where

- $V$  is a set of variables,
- $D$  is a domain of values,
- $R$  is a set of constraints,  $\{c_1, \dots, c_q\}$ , which are all pairs  $\langle s_i, R_i \rangle$ , where  $s_i$  is a pair of variables called the *scope*, and  $R_i$  is a subset of  $D^2$  called the *relation*.

All the constraint satisfaction problems (CSPs) in this paper will be binary, and thus we will omit the adjective binary.

A solution to a constraint satisfaction problem instance is a function  $f : V \rightarrow D$  such that for each constraint  $c_i$  with  $s_i = (v_1, v_2)$ , the pair  $f(v_1, v_2)$  is a member of  $R_i$ . An algorithm decides a CSP instance  $I$  if it outputs true if and only if that instance admits a solution.

The *primal graph* of a CSP instance  $I = (V, D, C)$  is a graph with vertex set  $V$  such that distinct vertices  $u, v \in V$  are adjacent if and only if there is a constraint whose scope contains both  $u$  and  $v$ .

The starting points for the reductions in this paper are the following two theorems, which state in a precise sense that the treewidth of the primal graph of a binary CSP establishes a lower bound on the best algorithm to decide it.

► **Theorem 6** ([17, 16]). *Let  $\mathcal{G}$  be an arbitrary class of graphs with unbounded treewidth. Let us consider the problem of deciding the binary CSP instances whose primal graph,  $G$ , lies in  $\mathcal{G}$ . This problem is  $W[1]$ -hard parameterized by the treewidth.*

► **Theorem 7** ([25]). *Assuming ETH, there exists a universal constant  $\alpha_{\text{CSP}}$  such that for any fixed primal graph  $G$  with  $\text{tw}(G) \geq 2$ , there is no algorithm deciding the binary CSP instances whose primal graph is  $G$  in time  $O(|D|^{\alpha_{\text{CSP}} \cdot \text{tw}(G)} / \log \text{tw}(G))$ .*

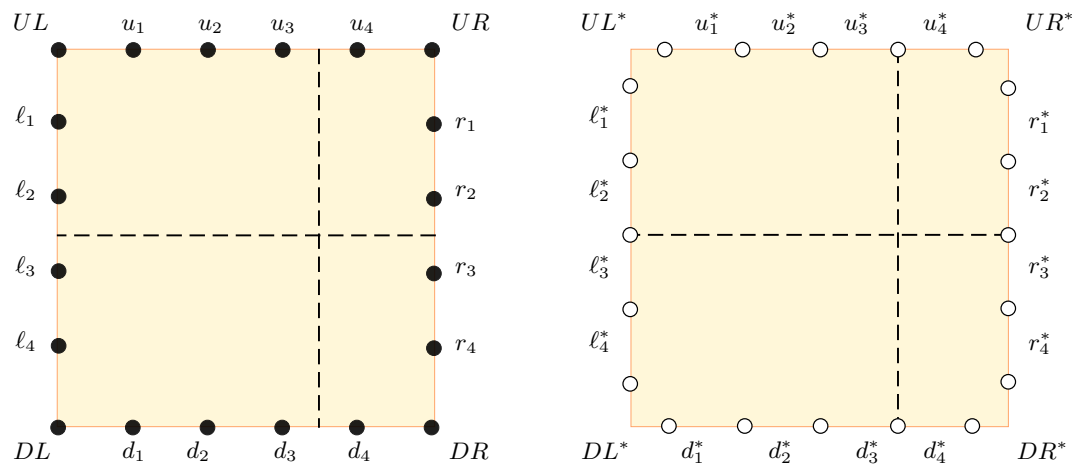
The first theorem is due to Grohe et al. [17] (see also Grohe [16]). The second one follows from the work of the third author [25]. Since this statement differs from the main theorem of [25], we explain in the full version [4, Section 2] how to obtain it.

**Cross gadgets.** We rely extensively on the following intricate family of gadgets introduced by the third author in his proof of hardness of PLANAR MULTIWAY CUT [26], which we call *cross gadgets*. Let  $\Delta$  be an integer. The gadgets always have the form of a planar graph  $G_S$  embedded on a disk, with  $4\Delta + 8$  distinguished vertices on its boundary, which are, in clockwise order, denoted by

$$UL, u_1, \dots, u_{\Delta+1}, UR, r_1, \dots, r_{\Delta+1}, DR, d_{\Delta+1}, \dots, d_1, DL, \ell_{\Delta+1}, \dots, \ell_1.$$

The embedding is chosen so that the boundary of the disk intersects the graph precisely in this set of distinguished vertices; the interior of the edges lie in the interior of the disk. We consider the vertices  $UL, UR, DR$ , and  $DL$  as terminals in that gadget, and thus a multiway cut  $M$  of the gadget is a subset of the edges of  $G_S$  such that  $G_S \setminus M$  has at least four components, and each of the terminals is in a distinct component. We say that a multiway cut  $M$  of the gadget represents the pair  $(i, j) \in [\Delta]^2$  (where, as usual,  $[\Delta]$  denotes the set  $\{1, \dots, \Delta\}$ ) if  $G_S \setminus M$  has exactly four components that partition the distinguished vertices into the following classes:

$$\begin{array}{ll} \{UL, u_1, \dots, u_j, \ell_1, \dots, \ell_i\} & \{UR, u_{j+1}, \dots, u_{\Delta+1}, r_1, \dots, r_i\} \\ \{DL, d_1, \dots, d_j, \ell_{i+1}, \dots, \ell_{\Delta+1}\} & \{DR, d_{j+1}, \dots, d_{\Delta+1}, r_{i+1}, \dots, r_{\Delta+1}\} \end{array}$$



■ **Figure 1** Left: a cross gadget  $G_S$  for  $\Delta = 3$ . The dashed line indicates a multiway cut that represents the pair  $(2, 3)$ . Right: a dual cross gadget  $G_S^*$  for  $\Delta = 3$ . The dashed lined is a dual multiway cut that represents the pair  $(2, 3)$ .

We remark that, as in the original article [26], the notation  $(i, j)$  is in matrix form. We will use the same convention throughout this paper, especially in Section 3.

The boundary of a cross gadget and a multiway cut representing a pair are pictured on Figure 1, left. The properties that we require are summarized in the following lemma:

► **Lemma 8** ([26, Lemma 2]). *Given a subset  $S \subseteq [\Delta]^2$ , we can construct in polynomial time a planar gadget  $G_S$  with  $\text{poly}(\Delta)$  unweighted edges and vertices, and an integer  $D_1$  such that the following properties hold:*

- i. *For every  $(i, j) \in S$ , the gadget  $G_S$  has a multiway cut of weight  $D_1$  representing  $(i, j)$ .*
- ii. *Every multiway cut of  $G_S$  has weight at least  $D_1$ .*
- iii. *If a multiway cut of  $G_S$  has weight  $D_1$ , then it represents some  $(i, j) \in S$ .*

Note that in [26], the third author uses weights to define the gadgets, but as he explains at the end of the introduction, the weights are polynomially large integers and thus can be emulated with parallel unweighted edges.

In the following, we will also use the dual of the graph  $G_S$  as one of our gadgets, yielding a *dual cross gadget*  $G_S^*$  (see Figure 1). Its properties mirror exactly the ones of cross gadgets in a dual setting, we refer to the full version [4, Section 2] for details.

### 3 The 4-regular graph tiling problem

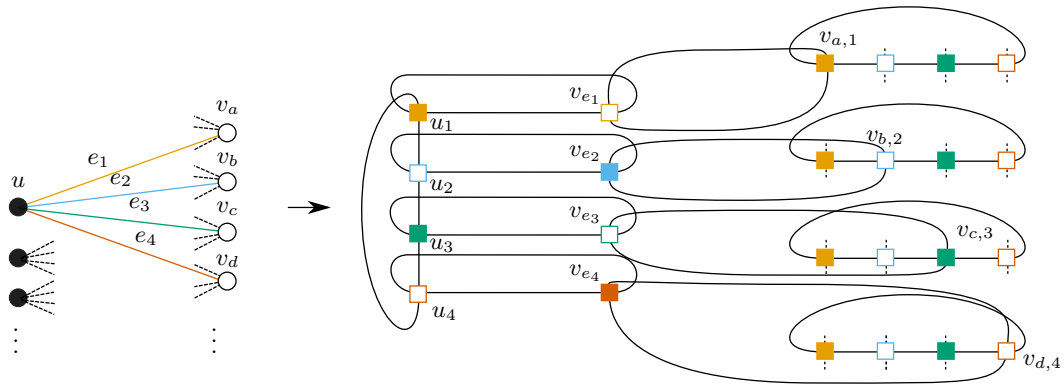
We introduce the problem 4-REGULAR GRAPH TILING which will be used as a basis to prove the reductions involved in Theorems 1 and 2.

4-REGULAR GRAPH TILING

**Input:** Integers  $k, n$ ; a four-regular graph  $\Gamma$  on  $k$  vertices where the edges are labeled by  $U, D, L, R$  in a way that each vertex is incident to exactly one of each label; for each vertex  $v$ , a non-empty set  $S_v \subseteq [n] \times [n]$ .

**Output:** For each vertex  $v$ , a value  $s_v \in S_v$  such that if  $s_v = (i, j)$ ,

1. the first coordinate of  $s_{L(v)}$  and  $s_{R(v)}$  is  $i$ , and
2. the second coordinate of  $s_{U(v)}$  and  $s_{D(v)}$  is  $j$ ,



■ **Figure 2** The reduction in the proof of Theorem 9. The bipartition on both sides are represented by hollow/solid vertices. The colors represent the 4-coloring of the edges, and the labels of the edges are suggested by their orientation, i.e., edges entering vertices vertically are labeled  $U$  or  $D$ , while edges entering vertices horizontally are labeled  $L$  or  $R$ .

where  $U(v)$ ,  $D(v)$ ,  $L(v)$ , and  $R(v)$  denote the vertex of the graph  $\Gamma$  connected to  $v$  via an edge labeled respectively by  $U$ ,  $D$ ,  $L$ , and  $R$ .

We call the two conditions above the *compatibility conditions* of the 4-REGULAR GRAPH TILING instance. The graph in the input is allowed to have parallel edges. It is easy to see that the GRID TILING problem [24] is a special case of 4-REGULAR GRAPH TILING.

In this section, we prove a larger lower bound for this more general problem: we prove an  $n^{\Omega(k/\log k)}$  lower bound, conditionally to ETH, for 4-REGULAR GRAPH TILING, even when the problem is restricted to bipartite instances and when fixing  $k$ . We also show that it is  $W[1]$ -hard when parameterized by the integer  $k$  (even for bipartite instances). Precisely:

► **Theorem 9.**

1. The 4-REGULAR GRAPH TILING problem restricted to instances whose underlying graph is bipartite, parameterized by the integer  $k$ , is  $W[1]$ -hard.
2. Assuming ETH, there exists a universal constant  $\alpha_{GT}$  such that for any fixed integer  $k \geq 2$ , there is no algorithm that decides all the 4-REGULAR GRAPH TILING instances whose underlying graph is bipartite and has at most  $k$  vertices, in time  $O(n^{\alpha_{GT} \cdot k / \log k})$ .

The analogous result for GRID TILING by the third author [24] embeds the  $k$ -CLIQUE problem in a  $k \times k$  grid. Here we start from a hardness result for 4-regular binary CSPs that follows from Theorem 7 and directly represent the problem as a 4-REGULAR GRAPH TILING instance by locally replacing each variable and each binary constraint in an appropriate way.

**Proof.** In the proof, we will use the well-known fact that a  $d$ -regular bipartite graph  $G$  can be properly edge-colored with  $d$  colors. This is proved by induction on  $d$ : The case  $d = 0$  is trivial; in general, take a perfect matching of  $G$ , which exists by Hall’s marriage theorem; color the edges with color  $d$ ; the subgraph of  $G$  made of the uncolored edges satisfies the induction hypothesis with  $d - 1$ , so it admits a proper edge-coloring with  $d - 1$  colors; thus  $G$  has a proper edge-coloring with  $d$  colors. This also implies that computing such a proper edge-coloring takes polynomial time.

The proof of the theorem proceeds by a reduction from the binary CSP instances involved in Theorems 6 and 7. Starting from a binary CSP instance  $I = (V, D, C)$  whose primal graph is  $P$ , a 4-regular bipartite graph, we define an instance of 4-REGULAR GRAPH TILING,  $(k, n, \{S_i\}, \Gamma)$ , in the following way.

1. We set  $n = |D|$  and  $k = 6|V|$ .
2. We find a proper edge coloring of  $P$  with 4 colors, as indicated above.
3. Denoting by  $V_1$  and  $V_2$  the two subsets of vertices of  $P$  corresponding to the bipartition of  $P$ , for each vertex  $u$  of  $V_1$ , we create four vertices  $u_1, u_2, u_3, u_4$  in  $\Gamma$  which we connect in a cycle in this order using two  $U$  and two  $D$  edges. Similarly, for each vertex  $v$  of  $V_2$ , we create four vertices  $v_1, v_2, v_3, v_4$  in  $\Gamma$  which we connect in a cycle in this order using two  $R$  and two  $L$  edges.
4. For each edge  $e = uv$  labeled with a color  $i$ , where  $u \in V_1$  and  $v \in V_2$ , we create one vertex  $v_e$  in  $\Gamma$ , which is connected to  $u_i$  via two edges, one labeled  $R$  and one labeled  $L$ , and to  $v_i$  via two edges, one labeled  $U$  and one labeled  $D$ .
5. For each vertex  $u_i$  or  $v_i$  of  $\Gamma$  coming from a vertex of  $P$ , the corresponding subset  $S_{u_i}$  or  $S_{v_i}$  is set to be  $\text{Diag}([n]) := \{(x, x) \mid x \in [n]\}$ .
6. For each vertex  $v_e$  of  $\Gamma$  coming from an edge  $e = uv$  of  $P$ , where  $u \in V_1$  and  $v \in V_2$ , the corresponding subset  $S_e$  is set to be the relation corresponding to  $e$ .

See Figure 2 for an illustration of this reduction. We claim that the graph  $\Gamma$  is bipartite: The bipartition is obtained by picking for one side the odd-numbered  $u$  and  $v$  vertices and the  $v_e$  vertices for  $e$  labeled by an even color, and for the other side the even-numbered  $u$  and  $v$  vertices and the  $v_e$  vertices for  $e$  labeled by an odd color. It follows from the construction that this is a bipartition.

We claim that this instance of 4-REGULAR GRAPH TILING is satisfiable if and only if  $I$  is satisfiable. Indeed, if  $I$  is satisfiable, the truth assignment  $f$  for  $I$  can be used to find the values for the  $s_i$  in the following way. For a vertex  $u_i$  or  $v_i$  of  $\Gamma$  coming from a vertex  $v$  of  $P$ , the value  $s_{u_i}$  or  $s_{v_i}$  can be chosen to be  $(f(v), f(v))$ . For a vertex  $v_e$  of  $\Gamma$  coming from an edge  $e = uv$  of  $P$  where  $u \in V_1$  and  $v \in V_2$ , the value of  $s_e$  can be chosen to be  $(f(u), f(v))$ . The compatibility conditions are trivially fulfilled. In the other direction, the values  $s_{v_i}$  for the four vertices of  $\Gamma$  coming from a vertex  $v$  of  $P$  are identical and of the form  $(x, x)$ . Choosing  $x$  as the truth assignment for  $v$  in  $I$  yields a solution to the CSP  $I$ .

We thus have a linear-time reduction from binary CSP, restricted to instances  $I$  whose primal graph has  $|V|$  vertices, is four-regular and is bipartite, to instances of 4-REGULAR GRAPH TILING on a bipartite graph with  $6|V|$  vertices. Combined with Theorem 6 applied to the infinite family  $\mathcal{H}$  of four-regular bipartite expanders output by Lemma 4 and Lemma 5 relating their treewidth to their number of vertices, this proves the first item of the theorem. For the second item, we fix an integer  $k$ ; by Lemma 4, there exists a constant  $c$  so that if  $k \geq c$ , there exists a four-regular bipartite expander  $G$  with expansion constant  $1 > c_{\text{exp}} > \lambda(G)/4$ , and with at least  $k/c$  and at most  $k/6$  vertices. We set  $\alpha_{\text{GT}}$  to be equal to  $\min(\log c/c, \alpha_{\text{CSP}} \cdot (1 - c_{\text{exp}})/16c)$ , where  $\alpha_{\text{CSP}}$  is the constant of Theorem 7. If  $k$  is smaller than  $c$ , such an expander may not exist in  $\mathcal{H}$ , but since  $\alpha_{\text{GT}} \cdot c/\log c < 1$ , the trivial linear lower bound for the 4-REGULAR GRAPH TILING problem, which holds for any  $k \geq 2$ , is enough to conclude. If  $k$  is at least  $c$ , observing that the polynomial-time reduction blows up the number of vertices by 6, we have that an algorithm deciding all the 4-REGULAR GRAPH TILING bipartite instances with at most  $k$  vertices in time  $O(n^{\alpha_{\text{GT}} \cdot k/\log k})$  would decide binary CSP instances whose primal graph is  $G$  in time

$$O(n^{\alpha_{\text{CSP}} \cdot \frac{(1-c_{\text{exp}})}{16c} \cdot \frac{c|V(G)|}{\log c|V(G)|}}) = O(n^{\alpha_{\text{CSP}} \cdot \frac{(1-c_{\text{exp}})}{16c} \cdot \frac{16c \cdot \text{tw}(G)}{(1-c_{\text{exp}}) \log \text{tw}(G)}}) = O(|D|^{\alpha_{\text{CSP}} \cdot \text{tw}(G)/\log \text{tw}(G)})$$

where the first equality uses Lemma 5. This would contradict Theorem 7. ◀

► Remark 10. It might seem more natural to use a definition of 4-REGULAR GRAPH TILING where *half-edges* are labeled by  $U, D, L$  and  $R$ , so that every edge contains either  $U$  and  $D$ , or  $L$  and  $R$  labels. This fits more the intuition that the top side of a vertex should be attached to the bottom side of the next vertex. It follows from roughly the same proof that the same hardness result also holds for that variant. However, it seems that both the bipartiteness and the unusual labeling are required for the reduction in Section 4.

#### 4 Multiway cut with four terminals

In this section, we prove the following proposition, which will yield Theorem 2 in the regime where the genus dominates the number of terminals.

► **Proposition 11.**

1. The MULTIWAY CUT problem when restricted to instances  $(G, T, \lambda)$  in which  $|T| = 4$  and  $G$  is embeddable on the surface of genus  $g$  is  $W[1]$ -hard parameterized by  $g$ .
2. Assuming ETH, there exists a universal constant  $\alpha_{\text{MC1}}$  such that for any fixed integer  $g \geq 2$ , there is no algorithm that decides all the MULTIWAY CUT instances  $(G, T, \lambda)$  for which  $G$  is embeddable on the surface of genus  $g$  and  $|T| = 4$ , in time  $O(n^{\alpha_{\text{MC1}} \cdot g / \log g})$ .

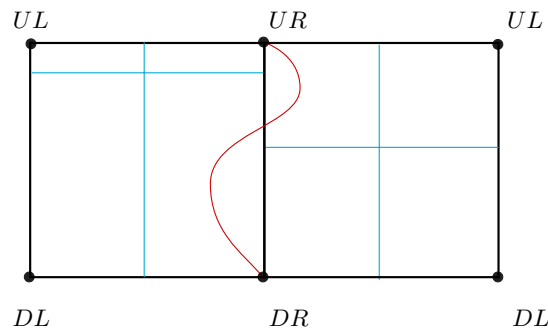
**Proof.** The idea is to reduce 4-REGULAR GRAPH TILING instances of Theorem 9 to the instances of MULTIWAY CUT specified by the proposition. Consider an instance of 4-REGULAR GRAPH TILING where the underlying graph  $\Gamma$  is bipartite and has at most  $k$  vertices ( $k$  being arbitrary for now). In polynomial time, we transform it into an equivalent instance  $(G, T, \lambda)$  of MULTIWAY CUT as follows.

1. To each vertex  $v$  of  $\Gamma$  corresponds a cross gadget  $G_S(v)$  where  $\Delta = n$  and the subset  $S$  is chosen to be  $S_v$ .
2. For each edge  $e = uv$  of  $\Gamma$  labeled  $U$ , we identify the vertices of the  $U$  side of the cross gadget  $G_S(v)$  to the corresponding vertices of the  $U$  side of the cross gadget  $G_S(u)$ . Similarly for the edges labeled  $D, R$ , and  $L$  for which the vertices on the  $D, R$ , and  $L$  sides, respectively, are identified. Note that only vertices, and not edges, are identified.
3. The four corner vertices  $UL, UR, DR$ , and  $DL$  of all the cross gadgets are identified in four vertices  $UL, UR, DR$ , and  $DL$ , where the four terminals are placed.

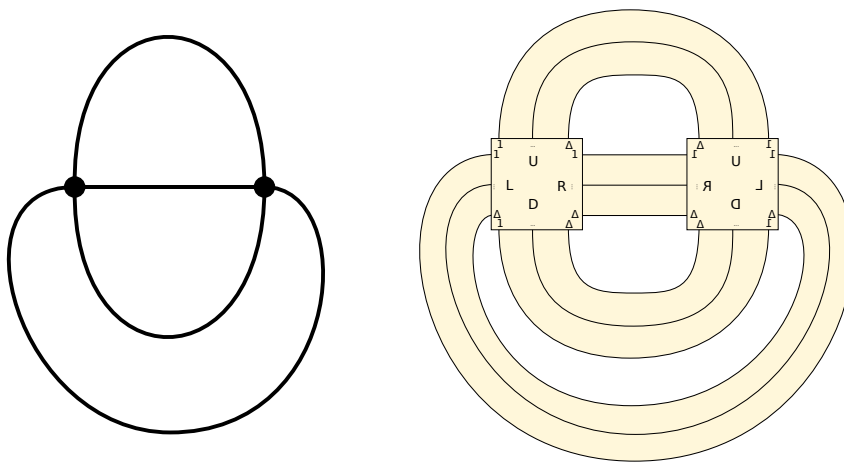
Note that since the sides are consistently matched in this last step, the four terminals remain distinct after this identification.

We claim that this instance admits a multiway cut of weight at most  $D_2 := kD_1$  (where  $D_1$  is the integer from Lemma 8) if and only if the 4-REGULAR GRAPH TILING instance is satisfiable. Assume first that the 4-REGULAR GRAPH TILING instance is satisfiable. For each vertex  $v$  of  $\Gamma$ , one can use the value  $s_v$  to choose, using Lemma 8(1), a multiway cut in  $G_S(v)$  representing  $s_v$ . We claim that the construction ensures that taking the union of all these sets of edges forms a multiway cut  $M$  separating the four terminals in  $G$ . Indeed, after removing the multiway cuts, the four terminals lie in four different components in each of the cross gadgets. This remains the case after identifying the four sides: consider two cross gadgets that have two sides identified; let  $w$  be a vertex on that common side; then, by the compatibility conditions in the definition of 4-REGULAR GRAPH TILING,  $w$  is connected, in the first gadget, to a terminal ( $UL, UR, DR$ , or  $DL$ ) if and only if it is connected to the corresponding terminal in the second gadget. The multiway cut  $M$  has weight at most  $kD_1$ , since it is the union of  $k$  edge sets of weight at most  $D_1$ .

For the other direction, we first observe that if the instance admits a multiway cut of weight at most  $kD_1$ , then each of the cross gadgets  $G_S$  must admit a multiway cut (otherwise the four terminals would not be disconnected). By Lemma 8(2), each of these  $k$  multiway cuts



■ **Figure 3** If the multiway cuts do not match (here represented by their duals), they do not separate the terminals.

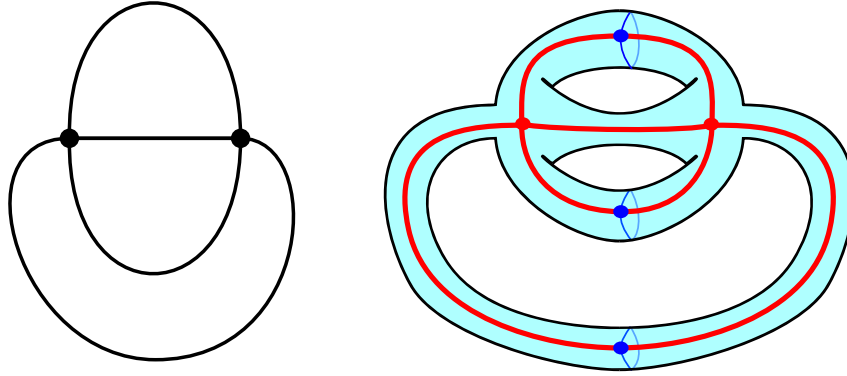


■ **Figure 4** Left: A bipartite four-valent graph  $\Gamma$  with two vertices. Right: The construction of the embedding of  $G$ . The orientation of each gadget of  $G$ , corresponding to a vertex  $v$ , is chosen according to the side of the bipartition vertex  $v$  lies in. This allows to connect pairs of vertices on the boundary of each gadget with the same indices.

has weight exactly  $D_1$ . Therefore, by Lemma 8(3), each of them represents some  $(i, j) \in S$ , which will be used as the value  $s_v$  for the 4-REGULAR GRAPH TILING instance. Furthermore, we claim that the multiway cuts need to match along identified sides, i.e., if a multiway cut represents the pair  $(i, j)$ , then a multiway cut in a cross gadget adjacent along an edge labeled  $U$  or  $D$  needs to represent a pair  $(k, j)$  for some  $k \in [n]$ , and similarly a multiway cut in a cross gadget adjacent along an edge labeled  $R$  or  $L$  needs to represent a pair  $(i, \ell)$  for some  $\ell \in [n]$ , for otherwise the four terminals are not separated. Indeed, if, say, a multiway cut representing the pair  $(i, j)$  is connected along an edge labeled  $R$  to a multiway cut representing the pair  $(i', \ell)$  for  $i' > i$ , there is a path connecting the terminals  $UR$  and  $DR$ , as pictured in Figure 3, contradicting the fact that we have a multiway cut. Therefore, the compatibility conditions of the 4-REGULAR GRAPH TILING instance are satisfied.

▷ **Claim 12.** The genus of the graph  $G$  is  $O(k)$ .

This claim is proved by providing an embedding of  $G$ , by connecting cross gadgets with at most  $k$  ribbons which will then be contracted. An important subtlety is that the naive way of doing so does not yield an orientable surface, and to fix this, the fact that  $G$  is bipartite turns out to be crucial: Our embedding switches the orientation of the gadgets based on the bipartition of the vertices (as pictured in Figure 4). The full proof of this claim is included in the full version [4, Section 4].



■ **Figure 5** Left: A bipartite four-valent graph  $\Gamma$  with two vertices. Right: The resulting graph  $\Gamma'$ . The graph  $\Gamma$  is in thick lines, and the edges forming the complement of a spanning tree are split with a new vertex, to which a loop (in thin lines) is attached.

To summarize: Given an instance of 4-REGULAR GRAPH TILING where the underlying graph  $\Gamma$  is bipartite and has at most  $k$  vertices, for an arbitrary  $k$ , we can transform it in polynomial time into an equivalent instance of MULTIWAY CUT with four terminals and whose graph has at most  $k \cdot \text{poly}(n)$  vertices and edges and is embeddable on a surface of genus at most  $ck$ , for some universal constant  $c \geq 1$ , where the polynomial is inherited from Lemma 8.

Combined with Theorem 9(1), this proves the first item. For the second one, for a given choice of  $g$ , we pick  $k = \lfloor g/c \rfloor$ . If  $k \geq 2$ , setting  $\alpha_{MC1} = \alpha_{GT}/cd$  where  $d$  is the degree of the polynomial and combining Theorem 9(2) with this reduction proves the second item of the theorem. Otherwise, as in the proof of Theorem 9, choosing  $\alpha_{MC1}$  to be smaller than  $2c/\log(2c)$  and using the trivial linear lower bound suffices to conclude. ◀

## 5 Shortest cut graph

In this section, we prove Theorem 1 on the hardness of the SHORTEST CUT GRAPH problem.

**Proof.** The idea is to reduce 4-REGULAR GRAPH TILING instances of Theorem 9 to instances of SHORTEST CUT GRAPH. Let  $\Gamma$  be a bipartite four-regular graph with  $k$  vertices.

From  $\Gamma$ , we build a surface  $S$  as follows (see Figure 5): We build one cylindrical tube for each edge of  $\Gamma$  and one sphere minus four disks for each vertex of  $\Gamma$ , attaching them in the natural way to obtain an orientable surface. By Euler’s formula, this surface has genus  $k + 1$ . Moreover, the graph  $\Gamma$  is naturally embedded in  $S$ , though not cellularly. In order to have a cellular embedding, and actually a cut graph, we transform  $\Gamma$  as follows. Let  $T$  be a spanning tree of  $\Gamma$ . Let  $\Gamma'$  be the graph obtained from  $\Gamma$  by subdividing each edge not in  $T$  into two edges, and adding a loop in the middle vertex. Now, embed  $\Gamma'$  into  $S$  in the natural way: Starting from the embedding of  $\Gamma$  into  $S$ , put each middle vertex on the corresponding cylindrical tube of  $S$ , and make the corresponding loop go around the tube. The resulting graph is a cut graph of  $S$  (indeed, it has a single face, because we only add loops in the middle of edges not in the spanning tree  $T$ ;  $\Gamma'$  has  $2k + 1$  vertices and  $4k + 2$  edges (being four-regular); so its unique face is a disk, by Euler’s formula).

Let  $V_1 \cup V_2$  be the bipartition of the vertices of  $\Gamma$ . We note that the above construction is possible while enforcing an arbitrary cyclic ordering of the edges incident to each vertex of  $\Gamma$ ; we do it in a way that the cyclic ordering of the edges around each vertex in  $V_1$  is the standard one ( $U, R, D, L$  in clockwise order), while the cyclic ordering around each vertex



in  $V_2$  is reversed ( $U, L, D, R$  in clockwise order). We now build a graph  $G$  embedded on the same surface  $S$  as follows, obtained by replacing each vertex of  $\Gamma'$  with a dual cross gadget and by (almost) identifying vertices on the corresponding sides of adjacent gadgets. In detail:

1. To each vertex  $v$  of  $\Gamma$  corresponds a dual cross gadget  $G_S^*(v)$  where  $\Delta = n$  and the subset  $S$  is chosen to be  $S_v$ . We embed that dual cross gadget with the same orientation as the corresponding vertex of  $\Gamma'$ .
2. For each edge  $e = uv$  of  $T$ , we identify the vertices (not the edges) on the side of  $G_S^*(u)$  corresponding to the label of  $e$  to the vertices on the same side of  $G_S^*(v)$ . By the choice of the rotation systems, and for the same reason as in Figure 4, this identifies the vertices in the gadget associated with  $u$  to the corresponding vertices in the gadget associated with  $v$ ; for example, if the label of edge  $e$  is  $R$ , the vertex  $r_i$  of the first gadget is associated to vertex  $r_i$  of the second gadget).
3. For an edge  $e = uv$  of  $\Gamma$  not in  $T$ , we use another dual cross gadget  $G_S^*(e)$ , for which we choose  $S$  to be the unconstrained relation  $S = [n]^2$ . We put that gadget on the vertex of  $\Gamma'$ . We identify the vertices on the side of  $G_S^*(u)$  corresponding to the label of  $e$  to the vertices of the same side of  $G_S^*(e)$ , and similarly the vertices on the side of  $G_S^*(v)$  corresponding to the label of  $e$  to the vertices on the opposite side of  $G_S^*(e)$ . The two opposite sides of  $G_S^*(e)$  which are not yet identified are identified to each other.

The following claim, whose proof is included in the full version [4, Section 5], shows that the reduction works as expected.

▷ **Claim 13.** The embedded graph  $G$  admits a cut graph  $C$  of weight at most  $(2k + 1)D_1$  if and only if the 4-REGULAR GRAPH TILING instance on  $\Gamma$  is satisfiable.

To summarize: Given an instance of 4-REGULAR GRAPH TILING where the underlying graph  $\Gamma$  is bipartite and has  $k$  vertices, for an arbitrary  $k$ , we can transform it in polynomial time into an equivalent instance of SHORTEST CUT GRAPH whose graph has  $k \cdot \text{poly}(n)$  vertices and edges, embedded on a surface of genus  $k + 1$ . Combined with Theorem 9(1), this proves the first item of the theorem. For the second item, for any choice of integer  $g \geq 3$ , we choose  $k = g - 1$ , and the above reduction, combined with Theorem 9(2), finishes the proof for  $\alpha_{CG} \leq \alpha_{GT}/d$  (where  $d$  is the degree of the polynomial of Lemma 8). For  $g = 2$ , we set  $\alpha_{CG} \leq 2$  and conclude using the trivial linear lower bound. ◀

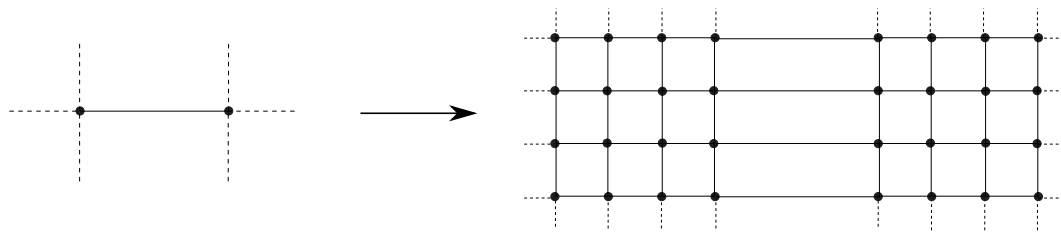
## 6 Multiway cut with a large number of terminals

The goal of this section is to prove the following proposition, which yields Theorem 2 when the number of terminals dominates the genus. Recall that  $g^*$  denotes  $\max(g, 2)$ .

▶ **Proposition 14.** *Assuming ETH, there exists a universal constant  $\alpha_{MC2}$  such that for any fixed choice of integers  $g \geq 0$  and  $t \geq 24g^*$ , there is no algorithm that decides all the MULTIWAY CUT instances  $(G, T, \lambda)$  for which  $G$  is embeddable on the surface of genus  $g$  and  $|T| \leq t$ , in time  $O(n^{\alpha_{MC2} \sqrt{g^* t} / \log(g^* t)})$ .*

**Sketch of proof.** We only sketch the proof and refer to the full version [4, Section 6] for details. Here are the key ideas: The reduction bypasses the use of 4-REGULAR GRID TILING and instead starts directly from a binary CSP instance on a four-regular graph  $P$ . In a way similar to the proofs of Proposition 11 and Theorem 1, this instance can be encoded in a MULTIWAY CUT instance by using cross gadgets to encode the constraints. In order to obtain the claimed lower bound, we will apply Theorem 7, and thus we need to choose for





■ **Figure 6** The construction of  $G_\delta$  for  $\delta = 4$ .

$P$  a graph having genus at most  $g^*$  and treewidth  $\Omega(g^*t)$ ; our construction uses a blow-up of an expander graph, i.e., an expander graph where each vertex has been replaced with a grid of an appropriate size (see Figure 6) – this is a construction reminiscent of one used in Gilbert, Hutchinson and Tarjan [15]. ◀

Finally, the proof of Theorem 2 is obtained by using Proposition 11 or Proposition 14, depending on the tradeoff between  $g$  and  $t$ . This is carried out in the full version [4, Section 7].

---

## References

- 1 Jianer Chen, Iyad A Kanj, Ljubomir Perković, Eric Sedgwick, and Ge Xia. Genus characterizes the complexity of certain graph problems: Some tight results. *Journal of Computer and System Sciences*, 73(6):892–907, 2007.
- 2 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight Bounds for Planar Strongly Connected Steiner Subgraph with Fixed Number of Terminals (and Extensions). In *25th ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.
- 3 Vincent Cohen-Addad, Éric Colin de Verdière, and Arnaud de Mesmay. A near-linear approximation scheme for multicuts of embedded graphs with a fixed number of terminals. In *29th ACM–SIAM Symposium on Discrete Algorithms (SODA)*, pages 1439–1458, 2018.
- 4 Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay. Almost Tight Lower Bounds for Hard Cutting Problems in Embedded Graphs. Full version of this article, 2019. arXiv:1903.08603.
- 5 Vincent Cohen-Addad and Arnaud de Mesmay. A fixed parameter tractable approximation scheme for the optimal cut graph of a surface. In *European Symposium on Algorithms (ESA)*, pages 386–398. Springer, 2015.
- 6 Éric Colin de Verdière. Multicuts in planar and bounded-genus graphs with bounded number of terminals. *Algorithmica*, 78(4):1206–1224, 2017.
- 7 Éric Colin de Verdière. Computational topology of graphs on surfaces. In Jacob E. Goodman, Joseph O’Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 23, pages 605–636. CRC Press LLC, third edition, 2018.
- 8 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *27th ACM–SIAM symposium on Discrete algorithms (SODA)*, pages 1650–1669. Society for Industrial and Applied Mathematics, 2016.
- 9 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 3. Springer, 2015.
- 10 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

- 11 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, Dániel Marx, and Tom C. van der Zanden. A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. In *50th ACM Symposium on Theory of Computing (STOC)*, pages 574–586, 2018.
- 12 Erik D Demaine, Fedor V Fomin, Mohammadtaghi Hajiaghayi, and Dimitrios M Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *Journal of the ACM*, 52(6):866–893, 2005.
- 13 Jeff Erickson and Sarel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004.
- 14 Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Subexponential Parameterized Algorithms for Planar and Apex-Minor-Free Graphs via Low Treewidth Pattern Covering. In *IEEE 57th Symposium on Foundations of Computer Science, FOCS*, pages 515–524, 2016. doi:10.1109/FOCS.2016.62.
- 15 John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984.
- 16 Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *Journal of the ACM*, 54(1):1, 2007.
- 17 Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *33rd ACM Symposium on Theory of Computing (STOC)*, pages 657–666, 2001.
- 18 Te C. Hu. Integer programming and network flows. Technical report, Wisconsin Univ Madison Dept. of Computer Sciences, 1969.
- 19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 653–662, 1998.
- 20 Ken-ichi Kawarabayashi, Bojan Mohar, and Bruce Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *49th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 771–780. IEEE, 2008.
- 21 Philip N. Klein and Dániel Marx. Solving PLANAR  $k$ -TERMINAL CUT in  $O(n^{c\sqrt{k}})$  Time. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 569–580. Springer, 2012.
- 22 Daniel Lokshtanov, Dániel Marx, Saket Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105), 2013.
- 23 Daniel Lokshtanov, Saket Saurabh, and Magnus Wahlström. Subexponential Parameterized Odd Cycle Transversal on Planar Graphs. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 424–434, 2012. doi:10.4230/LIPICs.FSTTCS.2012.424.
- 24 Dániel Marx. On the Optimality of Planar and Geometric Approximation Schemes. In *48th IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 338–348, 2007.
- 25 Dániel Marx. Can You Beat Treewidth? *Theory of Computing*, 6(1):85–112, 2010.
- 26 Dániel Marx. A Tight Lower Bound for Planar Multiway Cut with Fixed Number of Terminals. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 677–688, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- 27 Dániel Marx. The Square Root Phenomenon in Planar Graphs. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, volume 7924. Springer, 2013.
- 28 Dániel Marx, Marcin Pilipczuk, and Michał Pilipczuk. On subexponential parameterized algorithms for Steiner Tree and Directed Subset TSP on planar graphs. In *59th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 474–484, 2018.
- 29 Dániel Marx and Michał Pilipczuk. Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). In *31st International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 25, pages 542–553, 2014.

## 27:16 Almost Tight Lower Bounds for Hard Cutting Problems in Embedded Graphs

- 30 Dániel Marx and Michal Pilipczuk. Optimal Parameterized Algorithms for Planar Facility Location Problems Using Voronoi Diagrams. In *23rd European Symposium on Algorithms (ESA)*, pages 865–877, 2015.
- 31 Dániel Marx and Anastasios Sidiropoulos. The limited blessing of low dimensionality: when  $1 - 1/d$  is the best possible exponent for  $d$ -dimensional geometric problems. In *30th Symposium on Computational Geometry (SoCG)*, page 67. ACM, 2014.
- 32 Bojan Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM Journal on Discrete Mathematics*, 12(1):6–26, 1999.
- 33 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001.
- 34 Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics (TOG)*, 23(2):190–208, 2004.

# The VC Dimension of Metric Balls Under Fréchet and Hausdorff Distances

**Anne Driemel**

University of Bonn, Germany  
driemel@cs.uni-bonn.de

**Jeff M. Phillips**

University of Utah, Salt Lake City, USA  
jeffp@cs.utah.edu

**Ioannis Psarros**

National & Kapodistrian University of Athens, Greece  
ipsarros@di.uoa.gr

---

## Abstract

The Vapnik-Chervonenkis dimension provides a notion of complexity for systems of sets. If the VC dimension is small, then knowing this can drastically simplify fundamental computational tasks such as classification, range counting, and density estimation through the use of sampling bounds. We analyze set systems where the ground set  $X$  is a set of polygonal curves in  $\mathbb{R}^d$  and the sets  $\mathcal{R}$  are metric balls defined by curve similarity metrics, such as the Fréchet distance and the Hausdorff distance, as well as their discrete counterparts. We derive upper and lower bounds on the VC dimension that imply useful sampling bounds in the setting that the number of curves is large, but the complexity of the individual curves is small. Our upper bounds are either near-quadratic or near-linear in the complexity of the curves that define the ranges and they are logarithmic in the complexity of the curves that define the ground set.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Computational geometry

**Keywords and phrases** VC dimension, Fréchet distance, Hausdorff distance

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.28

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1903.03211>.

**Funding** *Anne Driemel*: Anne Driemel thanks the Hausdorff Center for Mathematics for their generous support and the Netherlands Organization for Scientific Research (NWO) for support under Veni Grant 10019853.

*Jeff M. Phillips*: Jeff Phillips thanks his support from NSF CCF-1350888, ACI-1443046, CNS-1514520, CNS-1564287, and IIS-1816149. Part of the work was completed while visiting the Simons Institute for Theory of Computing.

*Ioannis Psarros*: This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme « Human Resources Development, Education and Lifelong Learning » in the context of the project “Strengthening Human Resources Research Potential via Doctorate Research” (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

**Acknowledgements** We thank Peyman Afshani for useful discussions on the topic of this paper. We also thank the organizers of the 2016 NII Shonan Meeting “Theory and Applications of Geometric Optimization” where this research was initiated.



© Anne Driemel, Jeff M. Phillips, and Ioannis Psarros;  
licensed under Creative Commons License CC-BY

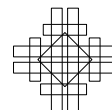
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 28; pp. 28:1–28:16

Leibniz International Proceedings in Informatics



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



## 1 Introduction

A *range space*  $(X, \mathcal{R})$  (also called *set system*) is defined by a ground set  $X$  and a set of ranges  $\mathcal{R}$ , where each  $r \in \mathcal{R}$  is a subset of  $X$ . A data structure for range searching answers queries for the subset of the input data that lies inside the query range. In range counting, we are interested only in the size of this subset. In our setting, a range is a metric ball defined by a curve and a radius. The ball contains all curves that lie within this radius from the center under a specific distance function (e.g., Fréchet or Hausdorff distance).

A crucial descriptor of any range space is its VC-dimension [33, 31, 30] and related shattering dimension, which we define formally below. These notions quantify how complex a range space is, and have played fundamental roles in machine learning [34, 6], data structures [12], and geometry [24, 10]. For instance, specific bounds on these complexity parameters are critical for tasks as diverse as neural networks [6, 27], art-gallery problems [32, 21, 28], and kernel density estimation [26]. The Fréchet distance is a popular distance measure for curves. The Fréchet distance is very similar to the Hausdorff distance for sets, which is defined as the minimal maximum distance of a pair of points, one from each set, under all possible mappings between the two sets. The difference between the two distance measures is that the Fréchet distance requires the mapping to adhere to the ordering of the points along the curve. Both distance measures allow flexible associations between parts of the input elements which sets them apart from classical  $\ell_p$  distances and makes them so suitable for trajectory data under varying speeds. In particular, the last five years have seen a surge of interest into data structures for trajectory processing under the Fréchet distance, manifested in a series of publications [14, 23, 15, 2, 35, 8, 19, 11, 18, 7, 20].

Our contribution in this paper is a comprehensive analysis of the Vapnik-Chervonenkis dimension of the corresponding range spaces. The resulting VC dimension bounds, while being interesting in their own right, have a plethora of applications through the implied sampling bounds. We detail a range of implications of our bounds in Section 10.

## 2 Definitions

In this section, we formally define the distances between curves as well as VC-dimension and range spaces, so we can state our main results. This basic set up will be enough to prove the main results for discrete distance. Then in Section 6 we provide more advanced geometric definitions and properties about VC dimension with our proofs for the continuous distances.

### 2.1 Distance measures

In the following, we define the Hausdorff distance, the discrete and the continuous Fréchet distance, and the Weak Fréchet distance. We denote by  $\|\cdot\|$  the Euclidean norm  $\|\cdot\|_2$ .

► **Definition 1** (Directed Hausdorff distance.). *Let  $X, Y$  be two subsets of some metric space  $(M, d)$ . The directed Hausdorff distance from  $X$  to  $Y$  is:*

$$d_{\vec{H}}(X, Y) = \sup_{u \in X} \inf_{v \in Y} d(u, v).$$

► **Definition 2** (Hausdorff distance.). *Let  $X, Y$  be two subsets of some metric space  $(M, d)$ . The Hausdorff distance between  $X$  and  $Y$  is:*

$$d_H(X, Y) = \max\{d_{\vec{H}}(X, Y), d_{\vec{H}}(Y, X)\}.$$

► **Definition 3.** Given polygonal curves  $V$  and  $U$  with vertices  $v_1, \dots, v_{m_1}$  and  $u_1, \dots, u_{m_2}$  respectively, a traversal  $T = (i_1, j_1), \dots, (i_t, j_t)$  is a sequence of pairs of indices referring to a pairing of vertices from the two curves such that:

1.  $i_1, j_1 = 1, i_t = m_1, j_t = m_2$ .
2.  $\forall (i_k, j_k) \in T : i_{k+1} - i_k \in \{0, 1\}$  and  $j_{k+1} - j_k \in \{0, 1\}$ .
3.  $\forall (i_k, j_k) \in T : (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$ .

► **Definition 4 (Discrete Fréchet distance).** Given polygonal curves  $V$  and  $U$  with vertices  $v_1, \dots, v_{m_1}$  and  $u_1, \dots, u_{m_2}$  respectively, we define the Discrete Fréchet Distance between  $V$  and  $U$  as the following function:

$$d_{dF}(V, U) = \min_{T \in \mathcal{T}} \max_{(i_k, j_k) \in T} \|v_{i_k} - u_{j_k}\|,$$

where  $\mathcal{T}$  denotes the set of all possible traversals for  $V$  and  $U$ .

Any polygonal curve  $V$  with vertices  $v_1, \dots, v_{m_1}$  and edges  $\overline{v_1 v_2}, \dots, \overline{v_{m_1-1} v_{m_1}}$  has a uniform parametrization that allows us to view it as a parametrized curve  $v : [0, 1] \mapsto \mathbb{R}^2$ .

► **Definition 5 (Fréchet distance).** Given two parametrized curves  $u, v : [0, 1] \mapsto \mathbb{R}^2$ , their Weak Fréchet distance is defined as follows:

$$d_F(u, v) = \min_{\substack{f: [0,1] \mapsto [0,1] \\ g: [0,1] \mapsto [0,1]}} \max_{\alpha \in [0,1]} \|v(f(\alpha)) - u(g(\alpha))\|,$$

where  $f$  ranges over all continuous and monotone bijections with  $f(0) = 0$  and  $f(1) = 1$ . The Weak Fréchet distance  $d_{wF}$  is defined as above, except that  $f$  and  $g$  range over all continuous functions (not exclusively bijections) with  $f(0) = 0$  and  $f(1) = 1$  and  $g(0) = 0$  and  $g(1) = 1$ .

## 2.2 Range spaces

Each range space can be defined as a pair of sets  $(X, \mathcal{R})$ , where  $X$  is the ground set and  $\mathcal{R}$  is the range set. Let  $(X, \mathcal{R})$  be a range space. For  $Y \subseteq X$ , we denote:

$$\mathcal{R}|_Y = \{R \cap Y \mid R \in \mathcal{R}\}.$$

If  $\mathcal{R}|_Y$  contains all subsets of  $Y$ , then  $Y$  is shattered by  $\mathcal{R}$ .

► **Definition 6 (Vapnik-Chernovenkis dimension).** The Vapnik-Chernovenkis dimension [30, 31, 33] (VC dimension) of  $(X, \mathcal{R})$  is the maximum cardinality of a shattered subset of  $X$ .

► **Definition 7 (Shattering dimension).** The shattering dimension of  $(X, \mathcal{R})$  is the smallest  $\delta$  such that, for all  $m$ ,

$$\max_{\substack{B \subseteq X \\ |B|=m}} |\mathcal{R}|_B| = O(m^\delta).$$

It is well-known [6, 24] that for a range space  $(X, \mathcal{R})$  with VC-dimension  $\nu$  and shattering dimension  $\delta$  that  $\nu \leq O(\delta \log \delta)$  and  $\delta = O(\nu)$ . So bounding the shattering dimension and bounding the VC-dimension are asymptotically equivalent within a log factor.

► **Definition 8 (Dual range space).** Given a range space  $(X, \mathcal{R})$ , for any  $p \in X$ , we define

$$\mathcal{R}_p = \{R \mid R \in \mathcal{R}, p \in R\}.$$

The dual range space of  $(X, \mathcal{R})$  is the range space  $(\mathcal{R}, \{\mathcal{R}_p \mid p \in X\})$ .

It is a well-known fact that if a range space has VC dimension  $\nu$ , then the dual range space has VC dimension  $\leq 2^{\nu+1}$  (see e.g. [24]).

Many ways are known to bound the VC dimension of geometric range spaces. For instance when the ground set is  $\mathbb{R}^d$  and the ranges are defined by inclusion in halfspaces, then the range space and its dual range space are isomorphic and both have VC-dimension and shattering dimension  $d$ . When the ranges are defined by inclusion in balls, then the VC-dimension and shattering dimension is  $d + 1$ , and the dual range spaces have bounds of  $d$  [24]. It is also for instance known [9] that the composition ranges formed as the  $k$ -fold union or intersection of ranges from a range space with bounded VC-dimension  $\nu$  induces a range space with VC-dimension  $O(\nu k \log k)$ , and this was recently shown that this is tight for even some simple range spaces such as those defined by halfspaces [13]. More such results are deferred to Section 6.

### 2.3 Range spaces induced by distance measures

Let  $(M, d)$  be a pseudometric space. We define the *ball* of radius  $r$  and center  $p$ , under the distance measure  $d$ , as the following set:

$$b_d(p, r) = \{x \in M \mid d(x, p) \leq r\},$$

where  $p \in M$ . The doubling dimension of a metric space  $(M, d)$ , denoted as  $\text{ddim}(M, d)$ , is the smallest integer  $t$  such that any ball can be covered by at most  $2^t$  balls of half the radius.

In this paper, we study the VC dimension of range spaces  $(X, \mathcal{R})$  induced by pseudometric spaces<sup>1</sup>  $(M, d)$  by setting  $X = M$  and

$$\mathcal{R} = \{b_d(p, r) \mid r \in \mathbb{R}, r > 0, p \in M\}.$$

It is a reasonable question to ask whether the doubling dimension of a metric space influences the VC dimension of the induced range space. In general, a bounded doubling dimension does not imply a bounded VC dimension of the induced range space and vice versa. Recently, Huang et al. [25] showed that if we allow a small  $(1 + \varepsilon)$ -distortion of the distance function  $d$ , the shattering dimension can be upper bounded by  $O(\varepsilon^{-O(\text{ddim}(M, d))})$ . It is conceivable that the doubling dimension of the metric space of the Discrete Fréchet distance and Hausdorff distance is bounded, as long as the underlying metric has bounded doubling dimension. However, for the continuous Fréchet distance, the doubling dimension is known to be unbounded [16]. Moreover, we will see that much better bounds can be obtained by a careful study of the specific distance measure.

Specifically, we study an *unbalanced* version of the above range space, in that we distinguish between the complexity of objects of the ground set and the complexity of objects defining the ranges. To this end, we define, for any integers  $d$  and  $m$ ,  $\mathbb{X}_m^d := (\mathbb{R}^d)^m$  and we treat the elements of this set as ordered sets of points in  $\mathbb{R}^d$  of size  $m$ . Formally, we study range spaces with ground set  $\mathbb{X}_m^d$  and range set defined as

$$\mathcal{R}_{d,k} = \{b_d(p, r) \cap \mathbb{X}_m^d \mid r \in \mathbb{R}, r > 0, p \in \mathbb{X}_k^d\}$$

under different variants of the Fréchet and the Hausdorff distance. We emphasize that the range space consists of ranges of all radii.

---

<sup>1</sup> While we may use the term *metric* or *pseudometric* to define the range, our methods do not assume any metric properties of the inducing distance measure.



### 3 Our Results

Table 1 shows an overview of our bounds. For metric balls defined on point sets (resp. point sequences) in  $\mathbb{R}^d$  we show that the VC dimension is at most near-linear in  $dk$ , the complexity of the ball centers that define the ranges, and at most logarithmic in  $dm$ , the complexity of point sets of the ground set. Our lower bounds show that these bounds are almost tight in all parameters  $k$ ,  $d$ , and  $m$ . For the Fréchet distance, where the ground set  $X$  are continuous polygonal curves in  $\mathbb{R}^d$  we show an upper bound that is quadratic in  $k$ , quadratic in  $d$ , and logarithmic in  $m$ . The same bounds in  $k$  and  $m$  hold for the Hausdorff distance, where the ground set are sets of line segments in  $\mathbb{R}^2$ . We obtain slightly better bounds in  $k$  for the Weak Fréchet distance. Our lower bounds extend to the continuous case, but are only tight in the dependence on  $m$  – the complexity of the ground set.

■ **Table 1** Our bounds on the VC dimension of range spaces of the form  $(\mathbb{X}_m^d, \mathcal{R}_{d,k})$ , for  $d$  being the distance measures in the table. In the first column we distinguish between  $\mathbb{X}_m^d$  consisting of *discrete* point sequences vs.  $\mathbb{X}_m^d$  consisting of *continuous* polygonal curves. The lower bounds hold for all distance measures in this table.

|            |              |                                           |                                                                  |                                                         |
|------------|--------------|-------------------------------------------|------------------------------------------------------------------|---------------------------------------------------------|
| discrete   | Hausdorff    | $O(dk \log(dkm))$ (Theorems 9,10)         | $\Omega(\max(dk \log k, \log dm))$<br>( $d \geq 4$ , Theorem 23) |                                                         |
|            | Fréchet      |                                           |                                                                  |                                                         |
| continuous | weak Fréchet | $O(d^2 k \log(dkm))$ (Theorem 16)         |                                                                  | $\Omega(\max(k, \log m))$<br>( $d \geq 2$ , Theorem 22) |
|            | Fréchet      | $O(d^2 k^2 \log(dkm))$ (Theorem 18)       |                                                                  |                                                         |
|            | Hausdorff    | $O(k^2 \log(km))$ ( $d = 2$ , Theorem 21) |                                                                  |                                                         |

While the VC dimension bounds for the discrete Hausdorff and Fréchet metric balls may seem like an easy implication of composition theorems for VC dimension [9, 13], we still find three things about these results remarkable:

1. First, for Fréchet variants, there are  $\Theta(2^k 2^m)$  valid alignment paths in the free space diagram. And one may expect that these may materialize in the size of the composition theorem. Yet by a simple analysis of the shattering dimension, we show that they do not.
2. Second, the VC dimension only has logarithmic dependence on the size  $m$  of the curves in the ground set, rather than a polynomial dependence one would hope to obtain by simple application of composition theorems. This difference has important implications in analyzing real data sets where we can query with simple curves (small  $k$ ), but may not have a small bound on the size of the curves in the data set (large  $m$ ).
3. Third, for the continuous variants, the range spaces can indeed be decomposed into problems with ground sets defined on line segments. However, we do not know of a general  $d$ -dimensional bound on the VC-dimension of range space with a ground set of segments, and ranges defined by segments within a radius  $r$  of another segment. We are able to circumvent this challenge with circuit-based methods to bound the VC-dimension and careful predicate design for the Fréchet distance, but for Hausdorff distance are only able to prove a bound in  $\mathbb{R}^2$ .

### 4 Our Approach

Our methods use the fact that both the Fréchet distance and the Hausdorff distance are determined by one of a discrete set of events, where each event involves a constant number of simple geometric objects. For example, it is well known that the Hausdorff distance between



two discrete sets of points is equal to the distance between two points from the two sets. The corresponding event happens as we consider a value  $\delta > 0$  increasing from 0 and we record which points of one set are contained in which balls of radius  $\delta$  centered at points from the other set. The same phenomenon is true for the discrete Fréchet distance between two point sequences. In particular, the so-called free-space matrix which can be used to decide whether the discrete Fréchet distance is smaller than a given value  $\delta$  encodes exactly the information about which pairs of points have distance at most  $\delta$ . The basic phenomenon remains true for the continuous versions of the two distance measures if we extend the set of simple geometric objects to include line segments and if we also consider triple intersections. Each type of event can be translated into a range space of which we can analyze the VC dimension. Together, the product of the range spaces encodes the information, which curves lie inside which metric balls, in the form of a set system. This representation allows us to prove bounds on the VC dimension of metric balls under these distance measures.

## 5 Basic Idea: Discrete Fréchet and Hausdorff

In this section we prove our upper bounds in the discrete setting. Let  $\mathbb{X}_m^d = (\mathbb{R}^d)^m$ ; we treat the elements of this set as ordered sets of points in  $\mathbb{R}^d$  of size  $m$ . The range spaces that we consider in this section are defined over the ground set  $\mathbb{X}_m^d$  and the range set of balls under either the Hausdorff or the Discrete Fréchet distance. The proofs in the proceeding sections all follow the basic idea of the proof in the discrete setting.

► **Theorem 9.** *Let  $(\mathbb{X}_m^d, \mathcal{R}_{dH,k})$  be the range space with  $\mathcal{R}_{dH,k}$  the set of all balls under the Hausdorff distance centered at point sets in  $\mathbb{X}_k^d$ . The VC dimension is  $O(dk \log(dkm))$ .*

**Proof.** Let  $\{S_1, \dots, S_t\} \subseteq \mathbb{X}_m^d$  and  $S = \bigcup_i S_i$ ; we define  $S$  so that it ignores the ordering with each  $S_i$  and is a single set of size  $tm$ . Any intersection of a Hausdorff ball with  $\{S_1, \dots, S_t\}$  is uniquely defined by a set  $\{B_1 \cap S, \dots, B_k \cap S\}$ , where  $B_1, \dots, B_k$  are balls in  $\mathbb{R}^d$ . To see that, notice that the discrete Hausdorff distance between two sets of points is uniquely defined by the distances between points of the two sets.

Consider the range space  $(\mathbb{R}^d, \mathcal{B})$ , where  $\mathcal{B}$  is the set of balls in  $\mathbb{R}^d$ . We know that the shattering dimension is  $d + 1$  [24]. Hence,

$$\max_{S \subseteq \mathbb{R}^d, |S|=tm} |\mathcal{B}|_S = O((tm)^{d+1}).$$

This implies that  $|\{\{B_1 \cap S, \dots, B_k \cap S\} \mid B_1, \dots, B_k \text{ are balls in } \mathbb{R}^d\}| \leq O((tm)^{(d+1)k})$ , and hence<sup>2</sup>,

$$2^t \leq O\left((tm)^{(d+1)k}\right) \implies t = O(dk \log(dkm)). \quad \blacktriangleleft$$

► **Theorem 10.** *Let  $(\mathbb{X}_m^d, \mathcal{R}_{dF,k})$  be the range space with  $\mathcal{R}_{dF,k}$  the set of all balls under the Discrete Fréchet distance centered at polygonal curves in  $\mathbb{X}_k^d$ . The VC dimension is  $O(dk \log(dkm))$ .*

**Proof.** Let  $\{S_1, \dots, S_t\} \subseteq \mathbb{X}_m^d$  and  $S = \bigcup_i S_i$ . Any intersection of a Discrete Fréchet ball with  $\{S_1, \dots, S_t\}$  is uniquely defined by a sequence  $B_1 \cap S, \dots, B_k \cap S$ , where  $B_1, \dots, B_k$  are balls in  $\mathbb{R}^d$ . The number of such sequences can be bounded by  $O((tm)^{(d+1)k})$  as in the

<sup>2</sup> for  $u > \sqrt{e}$  if  $x/\ln(x) \leq u$  then  $x \leq 2u \ln u$ . Hence, if  $tm/\log(tm) \leq dkm$ , then  $tm = O(dkm \log(dkm))$ .

proof of Theorem 9. Enforcing that a sequence contains a valid alignment path only reduces the number of possible distinct sets formed by  $t$  curves, and it can be determined using these intersections and the two orderings of  $B_1, \dots, B_k$  and of vertices within some  $S_j \in \mathbb{X}_m^d$ . ◀

## 6 Preliminaries

In this section, we provide a more advanced set of geometric primitives and other technical known results about VC-dimension. We also derive some simple corollaries. We also provide some basic results about the distances which will couple with the geometric primitives in our proofs for continuous distance measures.

We again consider a ground set  $\mathbb{X}_m^d = (\mathbb{R}^d)^m$  which we treat as a set of polygonal curves with points in  $\mathbb{R}^d$  of size  $m$ . Given such a curve  $s \in \mathbb{X}_m^d$ , let  $V(s)$  be its ordered set of vertices and  $E(s)$  its ordered set of edges.

### 6.1 A simple model of computation

We consider a model of computation that will be useful for modeling primitive geometric sets, and in turn bounding the VC-dimension of an associated range space. These will be useful in that they allow the invocation of powerful and general tools to describe range spaces defined by distances between curves. We allow the following operations, which we call *simple operations*:

- the arithmetic operations  $+$ ,  $-$ ,  $\times$ , and  $/$  on real numbers,
- jumps conditioned on  $>$ ,  $\geq$ ,  $<$ ,  $\leq$ ,  $=$ , and  $\neq$  comparisons of real numbers, and
- output 0 or 1.

We say a function requires  $t$  simple operations if it can be computed with a circuit of depth  $t$  composed only of these simple operations. Notably, the lack of a square-root operator creates some challenges when dealing with geometric objects.

### 6.2 Geometric primitives

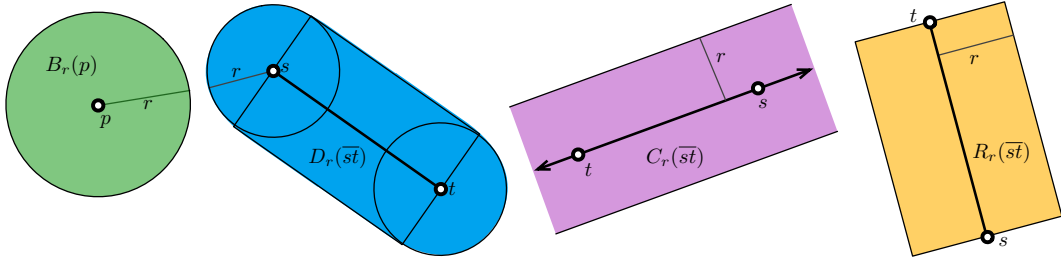
For any  $p \in \mathbb{R}^d$  we denote by  $B_r(p)$  the ball of radius  $r$ , centered at  $p$ . For any two points  $s, t \in \mathbb{R}^d$ , we denote by  $\overline{st}$  the line segment from  $s$  to  $t$ . Whenever we store such a line segment, for technicalities within the lemma below, we store the coordinates of its endpoints  $s$  and  $t$ . For any two points  $s, t \in \mathbb{R}^d$ , we define the stadium centered at  $\overline{st}$ ,  $D_r(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \overline{st} \ \|p - x\| \leq r\}$ . For any two points  $s, t \in \mathbb{R}^d$ , we define a cylinder  $C_r(\overline{st}) = \{x \in \mathbb{R}^d \mid \exists p \in \ell(\overline{st}) \ \|p - x\| \leq r\}$ , where  $\ell(\overline{st})$  denotes the line supporting the edge  $\overline{st}$ . Finally, for any two points  $s, t \in \mathbb{R}^d$ , we define the capped cylinder centered at  $\overline{st}$ :  $R_r(\overline{st}) = \{p + u \mid p \in \overline{st} \text{ and } u \in \mathbb{R}^d \text{ s.t. } \|u\| \leq r \text{ and } \langle t - s, u \rangle = 0\}$ .

For each of these geometric sets, we can determine if a point  $x \in \mathbb{R}^d$  is in the set with a constant number of operations under a simple model of computation.

► **Lemma 11.** *For a point  $x \in \mathbb{R}^d$ , and any set of the form  $B_r(p)$ ,  $D_r(\overline{st})$ ,  $C_r(\overline{st})$ , or  $R_r(\overline{st})$ , we can determine if  $x$  is in that set (returns 1, otherwise 0) using  $O(d)$  simple operations.*

**Proof.** For ball  $B_r(p)$  we can compute a distance  $\|x - p\|^2$  in  $O(d)$  time, and determine inclusion with a comparison to  $r^2$ . For cylinder  $C_r(\overline{st})$  we can compute the closest point to  $x$  on this line as

$$\pi_{\overline{st}}(x) = t + \frac{(s - t)\langle (s - t), x \rangle}{\|s - t\|^2}.$$



■ **Figure 1** Illustration of basic shapes in  $\mathbb{R}^2$ , from left to right: a ball  $B_r(p)$ , a stadium  $D_r(\overline{st})$ , a cylinder  $C_r(\overline{st})$ , and a capped cylinder  $R_r(\overline{st})$ .

Then we can determine inclusion by comparing  $\|\pi_{\overline{st}}(x) - x\|^2$  to  $r^2$ . For capped cylinder  $R_r(\overline{st})$  we also need to compare  $\|\pi_{\overline{st}}(x) - t\|^2$  and  $\|\pi_{\overline{st}}(x) - s\|^2$  to see if either of these terms is greater than  $\|s - t\|^2$ . For stadium  $D_r(\overline{st})$  we determine inclusion if any  $x$  is in any of  $R_r(\overline{st})$ ,  $B_r(s)$  or  $B_r(t)$ . ◀

### 6.3 Bounding the VC-Dimension

For range spaces defined on continuous curves, our proofs use a powerful theorem from Goldberg and Jerrum [22] as improved and restated by Anthony and Bartlett [6]. It allows one to easily bound the VC-dimension of geometric range spaces under our simple model of computation.

► **Theorem 12** (Theorem 8.4 [6]). *Suppose  $h$  is a function from  $\mathbb{R}^d \times \mathbb{R}^n$  to  $\{0, 1\}$  and let*

$$H = \{x \mapsto h(\alpha, x) : \alpha \in \mathbb{R}^d\}$$

*be the class determined by  $h$ . Suppose that  $h$  can be computed by an algorithm that takes as input the pair  $(\alpha, x) \in \mathbb{R}^d \times \mathbb{R}^n$  and returns  $h(\alpha, x)$  after no more than  $t$  simple operations. Then, the VC dimension of  $H$  is  $\leq 4d(t + 2)$ .*

An example implication can be seen for geometric sets via Lemma 11. Note that this implies any VC dimension upper bound proved in this approach applies to both the range space and its dual range space because the function  $h$  is unchanged and the ranges can still be described by  $O(d)$  real coordinates.

► **Corollary 13.** *For range spaces defined on  $\mathbb{R}^d$  with geometric sets  $B_r(p)$ ,  $D_r(\overline{st})$ ,  $C_r(\overline{st})$ , or  $R_r(\overline{st})$  as ranges, the VC dimension is  $O(d^2)$ . The same  $O(d^2)$  VC dimension bound holds for the corresponding dual range spaces, with ground sets as the geometric sets, and ranges defined by stabbing using points in  $\mathbb{R}^d$ .*

Note that these bounds are not always tight. Specifically, because the VC-dimension for ranges defined geometrically by balls  $B_r(p)$  is  $O(d)$  [24]. Moreover, the VC-dimension of range spaces defined by cylinders  $C_r(\overline{st})$  is known to be  $O(d)$  [4]. The ranges defined by capped cylinders  $R_r(\overline{st})$  are the intersection of a cylinder and two halfspaces, each with VC-dimension  $O(d)$  and hence by the composition theorem [9], this full range spaces also has VC-dimension  $O(d)$ . Finally, the stadium  $D_r(\overline{st})$  is defined by the union of a capped cylinder  $R_r(\overline{st})$  and two balls  $B_r(s)$  and  $B_r(t)$ ; hence again by the composition theorem [9], its VC-dimension is  $O(d)$ . However, it is not clear that these improved bounds hold for the dual range spaces, aside for the case of  $B_r$ . Moreover, when the ground set  $X$  of the

range space  $(X, \mathcal{R})$  is not  $\mathbb{R}^d$ , then we need to be cautious in using the  $k$ -fold composition theorem [9], which bounds the VC-dimension of complex range spaces derived as the logical intersection or union of simpler range spaces with bounded VC-dimension. In the case of a ground set  $X = \mathbb{R}^d$ , logical and geometric intersections are the same, but for other ground sets (like dual objects, or line segments  $\mathbb{X}_2^d$ ) this is not necessarily the case. For instance, a line segment  $e \in \mathbb{X}_2^d$  may intersect a ball  $B_r$  and also a halfspace  $H$  while not intersecting the intersection  $B_r \cap H$ .

## 6.4 Representation by predicates

In order to prove bounds on the VC dimension of range spaces defined on continuous curves, we establish sets of geometric predicates which are sufficient to determine if two curves have distance at most  $r$  to each other. Analyzing the range spaces associated with these predicates (over all possible radii  $r$ ) allows us to compose them further and to establish VC dimension bounds for the range space induced by the corresponding distance measure. For the Fréchet and Weak Fréchet distance, the predicates mirror those used in range searching data structures [2, 1]. And for the Hausdorff distance on continuous curves, the predicates are derived from the Voronoi diagram [5]. The technical challenges for each case are similar, but require different analyses.

## 7 The Fréchet distance

We consider the range spaces  $(\mathbb{X}_m^d, \mathcal{R}_{F_k})$  and  $(\mathbb{X}_m^d, \mathcal{R}_{wF_k})$ , where  $\mathcal{R}_{F_k}$  (resp.  $\mathcal{R}_{wF_k}$ ) denotes the set of all balls, centered at curves in  $\mathbb{X}_k^d$ , under the Fréchet distance (resp. weak Fréchet) distance.

### 7.1 Fréchet distance predicates

It is known that the Fréchet distance between two polygonal curves can be attained, either at a distance between their endpoints, at a distance between a vertex and a line supporting an edge, or at the common distance of two vertices with a line supporting an edge. The third type of event is sometimes called monotonicity event, since it happens when the Weak Fréchet distance is smaller than the Fréchet distance. In this sense, our representation of the ball of radius  $r$  under the Fréchet distance is based on the following predicates. Let  $s \in \mathbb{X}_m^d$  with vertices  $s_1, \dots, s_m$  and  $q \in \mathbb{X}_k^d$  with vertices  $q_1, \dots, q_k$ .

- $P_1$  (*Endpoints (start)*) This predicate returns true if and only if  $\|s_1 - q_1\| \leq r$ .
- $P_2$  (*Endpoints (end)*) This predicate returns true if and only if  $\|s_m - q_k\| \leq r$ .
- $P_3$  (*Vertex-edge (horizontal)*) Given an edge of  $s$ ,  $\overline{s_j s_{j+1}}$ , and a vertex  $q_i$  of  $q$ , this predicate returns true iff there exist a point  $p \in \overline{s_j s_{j+1}}$ , such that  $\|p - q_i\| \leq r$ .
- $P_4$  (*Vertex-edge (vertical)*) Given an edge of  $q$ ,  $\overline{q_i q_{i+1}}$ , and a vertex  $s_j$  of  $s$ , this predicate returns true iff there exist a point  $p \in \overline{q_i q_{i+1}}$ , such that  $\|p - s_j\| \leq r$ .
- $P_5$  (*Monotonicity (horizontal)*) Given two vertices of  $s$ ,  $s_j$  and  $s_t$  with  $j < t$  and an edge of  $q$ ,  $\overline{q_i q_{i+1}}$ , this predicate returns true if there exist two points  $p_1$  and  $p_2$  on the line supporting the directed edge, such that  $p_1$  appears before  $p_2$  on this line, and such that  $\|p_1 - s_j\| \leq r$  and  $\|p_2 - s_t\| \leq r$ .
- $P_6$  (*Monotonicity (vertical)*) Given two vertices of  $q$ ,  $q_i$  and  $q_t$  with  $i < t$  and an directed edge of  $s$ ,  $\overline{s_j s_{j+1}}$ , this predicate returns true if there exist two points  $p_1$  and  $p_2$  on the line supporting the directed edge, such that  $p_1$  appears before  $p_2$  on this line, and such that  $\|p_1 - q_i\| \leq r$  and  $\|p_2 - q_t\| \leq r$ .

► **Lemma 14** (Lemma 9, [1]). *Given the truth values of all predicates (P1) – (P6) of two curves  $s$  and  $q$  for a fixed value of  $r$ , one can determine if  $d_F(s, q) \leq r$ .*

Predicates  $P_1 - P_4$  are sufficient for representing metric balls under the weak Fréchet distance. The proof can be found in the full version of the paper [17].

► **Lemma 15.** *Given the truth values of all predicates (P1) – (P4) of two curves  $s$  and  $q$  for a fixed value of  $r$ , one can determine if  $d_{wF}(s, q) \leq r$ .*

## 7.2 Fréchet distance VC dimension bounds

We first consider the range space  $(\mathbb{X}_m^d, \mathcal{R}_{wF,k})$ , where  $\mathcal{R}_{wF,k}$  is the set of all balls under the Weak Fréchet distance centered at curves in  $\mathbb{X}_k^d$ . The main task is to translate the predicates  $P_1 - P_4$  into simple range spaces, and then bound their associated VC dimensions. Consider any two polygonal curves  $s \in \mathbb{X}_m^d$  and  $q \in \mathbb{X}_k^d$ . In order to encode the intersection of polygonal curves with metric balls, we will make use of the following sets:

- $P_1^r(q, s) = B_r(q_1) \cap V(s)$ ,
- $P_2^r(q, s) = B_r(q_k) \cap V(s)$ ,
- $P_3^r(q, s) = \{D_r(\overline{s_i s_{i+1}}) \cap V(q) \mid \overline{s_i s_{i+1}} \in E(s)\}$ ,
- $P_4^r(q, s) = \{D_r(\overline{q_i q_{i+1}}) \cap V(s) \mid \overline{q_i q_{i+1}} \in E(q)\}$ .

The proof of the following theorem can be found in the full version of the paper [17].

► **Theorem 16.** *Let  $\mathcal{R}_{wF,k}$  be the set of balls under the Weak Fréchet metric centered at polygonal curves in  $\mathbb{X}_k^d$ . The VC dimension of  $(\mathbb{X}_m^d, \mathcal{R}_{wF,k})$  is  $O(d^2 k \log(dkm))$ .*

We now consider the range space  $(\mathbb{X}_m^d, \mathcal{R}_{F,k})$ , where  $\mathcal{R}_{F,k}$  denotes the set of all balls, centered at curves in  $\mathbb{X}_k^d$ , under the Fréchet distance. The approach is the same as with the Weak Fréchet distance, except we also need to bound VC dimension of range spaces associated with predicates  $P_5$  and  $P_6$  to encode monotonicity. While there exists geometric set constructions that are used in the context of range searching [2, 1] we can simply appeal to Theorem 12. We need to define a set to represent predicates  $P_5$  and  $P_6$ . The appropriate ground set is over two points  $q_j, q_t \in \mathbb{R}^d$ , which for notational simplicity we reuse  $\mathbb{X}_2^d$ . Then the ranges  $\mathcal{M}$  are defined by sets  $M_r(\overline{st}) \in \mathcal{M}$ , defined with respect to radii  $r \geq 0$  and line segments  $\overline{st}$ . Specifically,  $M_r(\overline{st}) \subset \mathbb{X}_2^d$  so any  $\{q_1, q_2\} \in M_r(\overline{st})$  satisfies that

- $\|p_1 - q_1\| \leq r$  and  $\|p_2 - q_2\| \leq r$ ;
- $p_1, p_2 \in \ell$  where  $\overline{st}$  supports  $\ell$ ; and
- $p_1$  is less than  $p_2$  along the line as  $\langle p_1, t - s \rangle \leq \langle p_2, t - s \rangle$ .

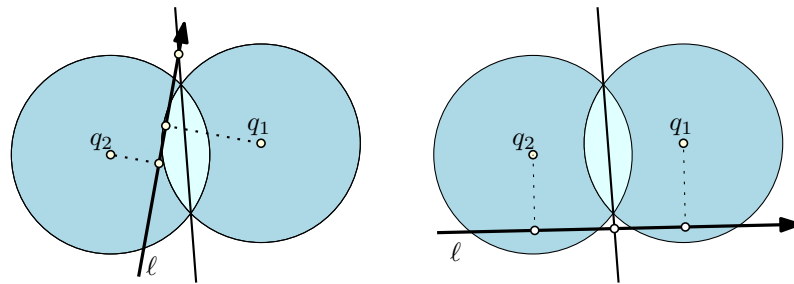
The predicate  $P_5$  is satisfied if  $s_j, s_t \in M_r(\overline{q_i q_{i+1}})$  and predicate  $P_6$  is satisfied if  $q_i, q_t \in M_r(\overline{s_j s_{j+1}})$ .

► **Lemma 17.** *The VC dimension of the range space  $(\mathbb{X}_2^d, \mathcal{M})$ , and of the associated dual range space, is  $O(d^2)$ .*

**Proof.** We may assume that  $q_1$  is the origin, since we can subtract  $q_1$  from all vectors  $s, t, q_2$  using  $O(d)$  simple calculations, without changing the outcome. As with bounding the VC dimension of range spaces on  $\mathbb{R}^d$  induced by sets  $C_r(\overline{st})$ , we can derive the closest points on  $\ell$  as  $\pi_{\overline{st}}(q_1)$  and  $\pi_{\overline{st}}(q_2)$  using  $O(d)$  simple operations.

If  $\ell$  is not perpendicular to  $q_2 - q_1$ , then it intersects the bisector of  $q_1$  and  $q_2$  and we can compute this intersection point as follows:

$$b_\ell(q_1, q_2) := s - \frac{\langle q_2, s - q_2/2 \rangle}{\langle t - s, q_2 \rangle} (t - s), \quad (1)$$



■ **Figure 2** Illustration of predicate  $P_5$  with line  $\ell$  and the two disks centered at  $q_1$  and  $q_2$ . In these examples, the projection of  $q_2$  onto  $\ell$  appears before the projection of  $q_1$  onto  $\ell$  along the direction of  $\ell$  and the intersection of  $\ell$  with the bisector lies outside of the lens formed by the two disks. On the left, the predicate is satisfied by setting  $p_1 = p_2 = \pi_{st}^{-1}(q_1)$ . On the right, the predicate evaluates to false.

This takes  $O(d)$  simple operations. Now, predicate  $P_5$  can be computed as follows (Predicate  $P_6$  can be computed in the same way):

- 1: **if**  $(\|\pi_{st}^{-1}(q_1) - q_1\|^2 > r^2)$  or  $(\|\pi_{st}^{-1}(q_2) - q_2\|^2 > r^2)$  **then**
- 2:     **return** 0
- 3: **else if**  $\langle \pi_{st}^{-1}(q_1), t - s \rangle \leq \langle \pi_{st}^{-1}(q_2), t - s \rangle$  **then**
- 4:     **return** 1
- 5: **else if**  $\|\pi_{st}^{-1}(q_1) - q_2\|^2 \leq r^2$  **then**
- 6:     **return** 1
- 7: **else if**  $\|\pi_{st}^{-1}(q_2) - q_1\|^2 \leq r^2$  **then**
- 8:     **return** 1
- 9: **else if**  $\langle q_2 - q_1, t - s \rangle \neq 0$  **then**
- 10:     compute  $b_\ell(q_1, q_2)$  using Eq. (1)
- 11:     **if**  $\|b_\ell(q_1, q_2) - q_1\|^2 \leq r^2$  and  $\|b_\ell(q_1, q_2) - q_2\|^2 \leq r^2$  **then**
- 12:         **return** 1
- 13:     **end if**
- 14: **else return** 0
- 15: **end if**

Lines 1-4 test if  $p_1 = \pi_{st}^{-1}(q_1)$  and  $p_2 = \pi_{st}^{-1}(q_2)$  satisfy the predicate. Lines 5-8 test if  $p_1 = p_2 = \pi_{st}^{-1}(q_2)$  or  $p_1 = p_2 = \pi_{st}^{-1}(q_1)$  satisfies the predicate. Then Line 9-12 tests if  $p_1 = p_2 = b_\ell(q_1, q_2)$  satisfies the predicate. Otherwise, we conclude that the predicate is not satisfied for any choice of  $p_1, p_2 \in \ell$ . To see why this is correct, assume the test in line 3 evaluates to false. In this case, the predicate is satisfied only if  $\ell$  intersects the lens formed by the intersection of the two balls centered at  $q_1, q_2$ . If the line intersects the bisector of  $q_1$  and  $q_2$  inside the lens, then we will find a satisfying assignment to  $p_1$  and  $p_2$ . If the line intersects the bisector outside of the lens, then by convexity the intersection of the line with either ball is completely contained in one of the two halfspaces bounded by the bisector. Therefore we would find a satisfying assignment to  $p_1$  and  $p_2$  among the closest points on the line to  $q_1$  or  $q_2$ , if there exists one. See Figure 2 for an example of the last case. ◀

We define sets to correspond with predicates  $P_5$  and  $P_6$ :

- $P_5^r(q, s) = \{\{s_j, s_t\} \in V(s) \times V(s) \mid (s_j, s_t) \in M_r(\overline{q_i q_{i+1}}) \text{ and } \overline{q_i q_{i+1}} \in E(q)\}$ .
- $P_6^r(q, s) = \{\{q_i, q_t\} \in V(q) \times V(q) \mid (s_i, s_t) \in M_r(\overline{s_j s_{j+1}}) \text{ and } \overline{s_j s_{j+1}} \in E(s)\}$ .

► **Theorem 18.** Let  $\mathcal{R}_{F,k}$  be the set of all balls, under the Fréchet distance, centered at polygonal curves in  $\mathbb{X}_k^d$ . The VC dimension of  $(\mathbb{X}_m^d, \mathcal{R}_{F,k})$  is  $O(d^2 k^2 \log(dkm))$ .

**Proof.** Due to Lemma 14, if  $S \subset \mathbb{X}_m^d$  is a set of  $t$  polygonal curves and  $q \in \mathbb{X}_k^d$ , the set  $\{s \in S \mid d_F(s, q) \leq r\}$  is uniquely defined by the sets

$$\bigcup_{s \in S} P_1^r(q, s), \bigcup_{s \in S} P_2^r(q, s), \bigcup_{s \in S} P_3^r(q, s), \bigcup_{s \in S} P_4^r(q, s), \bigcup_{s \in S} P_5^r(q, s), \bigcup_{s \in S} P_6^r(q, s).$$

As in the proof of Theorem 16, the number of all possible sets

$$\left( \bigcup_{r \geq 0} \bigcup_{s \in S} P_1(q, s), \bigcup_{r \geq 0} \bigcup_{s \in S} P_2(q, s), \bigcup_{r \geq 0} \bigcup_{s \in S} P_3(q, s), \bigcup_{r \geq 0} \bigcup_{s \in S} P_4(q, s) \right)$$

is bounded by  $(tm)^{O(d^2 k)}$ .

By Lemma 17 we are able to bound the number of all possible sets  $\bigcup_{r \geq 0} \bigcup_{s \in S} P_5^r(q, s)$  as  $(tm)^{O(d^2 k^2)}$ . The  $k^2$  term arises because we consider  $\Theta(k^2)$  pairs  $s_j, s_t$  for predicate  $P_5$ . And because this bound is proven using Theorem 12, then it applies to the dual range space, and we also bound the number of possible sets in  $\bigcup_{r \geq 0} \bigcup_{s \in S} P_6^r(s, q)$  as also  $(tm)^{O(d^2 k^2)}$ . So ultimately,

$$2^t \leq (tm)^{O(d^2 k^2)} \implies t = O(d^2 k^2 \log(dkm)). \quad \blacktriangleleft$$

## 8 Hausdorff distance

We consider the range space  $(\mathbb{X}_m^d, \mathcal{R}_{H_k}^r)$ , where  $\mathcal{R}_{H_k}^r$  denotes the set of all balls, of radius  $r$  centered at curves in  $\mathbb{X}_k^d$ , under the symmetric Hausdorff distance.<sup>3</sup> We also consider the same problems under both directed versions of the Hausdorff distance, and their induced range spaces  $(\mathbb{X}_m^d, \mathcal{R}_{H_k}^r)$  and  $(\mathbb{X}_m^d, \overleftarrow{\mathcal{R}}_{H_k}^r)$ . While some intermediate arguments hold in  $\mathbb{R}^d$ , we are only able to provide VC dimension bounds in  $\mathbb{R}^2$ . Proofs can be found in the full version of the paper [17].

► **Theorem 19.** Let  $\overrightarrow{\mathcal{R}}_{H,k}$  be the set of all balls, under the directed Hausdorff distance from polygonal curves in  $\mathbb{X}_k^2$ . The VC dimension of  $(\mathbb{X}_m^2, \overrightarrow{\mathcal{R}}_{H,k})$  is  $O(k^2 \log(km))$ .

► **Theorem 20.** Let  $\overleftarrow{\mathcal{R}}_{H,k}$  be the set of all balls, under the directed Hausdorff distance to polygonal curves in  $\mathbb{X}_k^2$ . The VC dimension of  $(\mathbb{X}_m^2, \overleftarrow{\mathcal{R}}_{H,k})$  is  $O(k \log(km))$ .

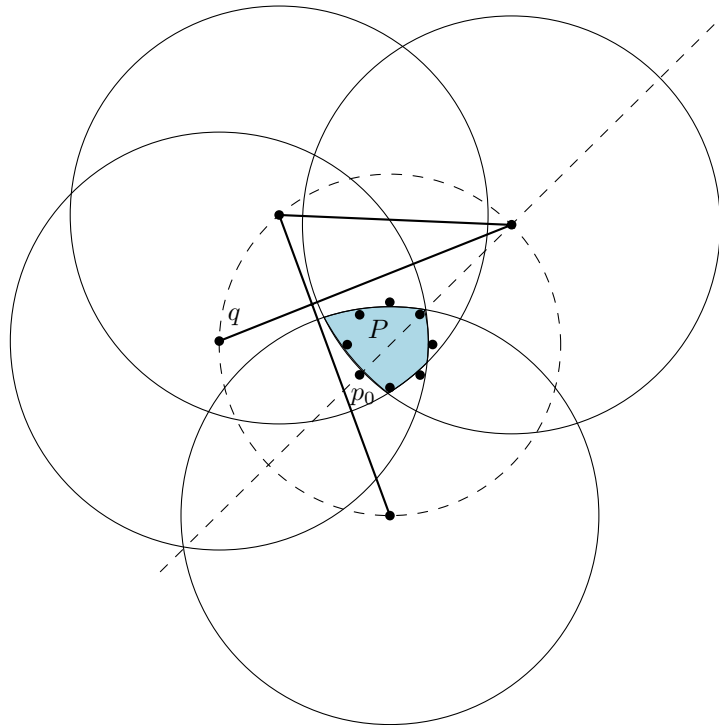
► **Theorem 21.** Let  $\mathcal{R}_{H,k}$  be the set of all balls, under the symmetric Hausdorff distance in  $\mathbb{X}_k^2$ . The VC dimension of  $(\mathbb{X}_m^2, \mathcal{R}_{H,k})$  is  $O(k^2 \log(km))$ .

## 9 Lower bounds

Our lower bounds are constructed in the simplified setting that either  $k = 1$  or  $m = 1$ , i.e., either the ground set or the curves defining the metric ball consist of one vertex only. In this case, all of our considered distance measures (except for one direction of the directed

<sup>3</sup> The proofs in this section are written for polygonal curves in  $\mathbb{X}_m^d$  (resp.  $\mathbb{X}_m^2$ ), but they readily extend to (not-necessarily connected) sets of line segments in  $\mathbb{R}^d$  (resp.  $\mathbb{R}^2$ ) of cardinality  $m' = \frac{m-1}{2}$ .





■ **Figure 3** A curve  $q$  with metric ball of radius  $R - \varepsilon$  containing a subset of  $P$ . The shaded area is the set of points that are contained inside the metric ball.

Hausdorff distance) are equal. The basic idea behind our lower bound construction is that, for  $m = 1$ , the ranges behave like convex polygons with  $k$  facets. In particular, the set of points contained inside the range centered at a curve  $q$ , is equal to the intersection of a set of equal-size Euclidean balls centered at the vertices of  $q$ . Figure 3 shows a sketch of the construction. For  $k = 1$  and  $m \geq 1$ , we use [24, Lemma 5.18], which bounds the VC dimension of the dual range space as a function of the VC dimension of the primal space. Proofs of the lower bounds can be found in the full version of the paper [17].

► **Theorem 22.** *The VC-dimension of the range spaces  $(\mathbb{X}_m^2, \mathcal{R}_{dF,k})$ ,  $(\mathbb{X}_m^2, \mathcal{R}_{dH,k})$ ,  $(\mathbb{X}_m^2, \mathcal{R}_{F,k})$ ,  $(\mathbb{X}_m^2, \mathcal{R}_{wF,k})$ , and  $(\mathbb{X}_m^2, \mathcal{R}_{H,k})$  is  $\Omega(\max(k, \log m))$ .*

► **Theorem 23.** *For  $d \geq 4$ , the VC-dimension of the range spaces  $(\mathbb{X}_m^d, \mathcal{R}_{dF,k})$ ,  $(\mathbb{X}_m^d, \mathcal{R}_{dH,k})$ ,  $(\mathbb{X}_m^d, \mathcal{R}_{wF,k})$ ,  $(\mathbb{X}_m^d, \mathcal{R}_{F,k})$ , and  $(\mathbb{X}_m^d, \mathcal{R}_{H,k})$  is  $\Omega(\max(dk \log k, \log dm))$ .*

## 10 Implications

In this section we demonstrate that bounds on the VC-dimension for the range space defined by metric balls on curves immediately implies various results about prediction and statistical generalization over the space of curves. In the following consider a range space  $(X, \mathcal{R})$  with a ground set of  $X$  of curves of size  $n = |X|$ , where  $\mathcal{R}$  are the ranges corresponding to metric balls for some distance measure we consider, and the VC-dimension is bounded by  $\nu$ . This section discusses accuracy bounds that depend directly on the size  $|X| = n$  and the VC-dimension  $\nu$ . They will assume that  $X$  is a random sample of some much larger set  $X_{\text{big}}$  or an unknown continuous generating distribution  $\mu$ . Under the randomness in this



assumed sampling procedure, there is a probability of failure  $\delta$  that often shows up in these bounds, but is minor since it shows up as  $\log(1/\delta)$ . These bounds take two closely-linked forms. First, given a limited set  $X$  from an unknown  $\mu$ , then how accurate is a query or a prediction made using only  $X$ . Second, given the ability to draw samples (at a cost) from an unknown distribution  $\mu$ , how many are required so the prediction on the samples set  $X$  has bounded prediction error. The theme of the following results, as implied by our above VC-dimension results, is that if these families of curves are only inspected with or queried with curves with a small number of segments ( $k$  is small), then the VC-dimension of the associated range space  $\nu = O(k \log km)$  or  $O(k^2 \log km)$  is small, and that such analyses generalize well. We show this in several concrete examples. More examples are detailed in the full version of the paper [17].

**Approximate range counting on curves.** Given a large set of curves  $X$  (of potentially very large complexity  $m$ ), and a query curve  $q$  (with smaller complexity  $k$ ) we would like to approximate the number of curves nearby  $q$ . For instance, we restrict  $X$  to historical queries at a certain time of day, and query with the planned route  $q$ , and would like to know the chance of finding a carpool. VC-dimension  $\nu$  of the metric balls shows up directly in two analyses. First, if we assume  $X \sim \mu$  where  $\mu$  is a much larger unknown distribution (but the real one), then we can estimate the accuracy of the fraction of all curves in this range within additive error  $O(\sqrt{(1/|X|)(\nu + \log(1/\delta))})$ . On the other hand, if  $X$  is too large to conveniently query, we can sample a subset  $S \subset X$  of size  $O((1/\varepsilon^2)(\nu + \log(1/\delta)))$  and know that the estimate for the fraction of curves from  $S$  in that range is within additive  $\varepsilon$  error of the fraction from  $X$ . Such sampling techniques have a long history in traditional databases [29], and have more recently become important when providing online estimates during a long query processing time as incrementally increasing size subsets are considered [3]. Ours provides the first formal analysis of these results for queries over curves.

**Density estimation of curves.** A related task in generalization to new curves is density estimation. Consider a large set of curves  $X$  which represent a larger unknown distribution  $\mu$  that models a distribution of curves; we want to understand how unusual a new curve  $q$  would be given we have not yet seen exactly the same curve before. One option is to use the distance to the ( $k$ th) nearest neighbor curve in  $X$ , or a bit more robust option is to choose a radius  $r$ , and count how many curves are within that radius (e.g., the approximate range counting results above). Alternatively, for  $X \subset \mathbb{M}$ , consider now a kernel density estimate  $\text{KDE}_X : \mathbb{M} \rightarrow \mathbb{R}$  defined by  $\text{KDE}_X(p) = \frac{1}{n} \sum_{p \in P} K(x, p)$  with kernel  $K(x, p) = \exp(-d(x, p)^2)$  (where  $d$  is some distance of choice among curves, e.g.,  $d_F$ ). The kernel is defined such that each superlevel set  $K_x^\tau = \{p \in \mathbb{M} \mid K(x, p) \geq \tau\}$  corresponds with some range  $R \in \mathcal{R}$  so that  $R \cap X = K_x^\tau \cap X$ . Then a random sample  $S \subset X$  of size  $O((1/\varepsilon^2)(\nu + \log \frac{1}{\delta}))$  satisfies that  $\|\text{KDE}_X - \text{KDE}_S\|_\infty \leq \varepsilon$  [26]. Thus, again the VC-dimension  $\nu$  of the metric balls directly influences this estimates accuracy, and for query curves with small complexity  $k$ , the bound is quite reasonable.

---

## References

- 1 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. *CoRR*, arXiv:1707.04789v1, 2017. [arXiv:1707.04789](https://arxiv.org/abs/1707.04789).
- 2 Peyman Afshani and Anne Driemel. On the complexity of range searching among curves. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*

- 2018, New Orleans, LA, USA, January 7-10, 2018, pages 898–917, 2018. doi:10.1137/1.9781611975031.58.
- 3 S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *EuroSys*, 1993.
  - 4 Yohji Akama, Kei Irie, Akitoshi Kawamura, and Yasutaka Uwano. VC Dimension of Principal Component Analysis. *Discrete & Computational Geometry*, 44:589–598, 2010.
  - 5 Helmut Alt, Bernd Behrends, and Johannes Blömer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, 13(3):251–265, September 1995. doi:10.1007/BF01530830.
  - 6 Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.
  - 7 Maria Astefanoaei, Paul Cesaretti, Panagiota Katsikouli, Mayank Goswami, and Rik Sarkar. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2018)*, volume 10, 2018.
  - 8 Julian Baldus and Karl Bringmann. A Fast Implementation of Near Neighbors Queries for Fréchet Distance (GIS Cup). In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'17, pages 99:1–99:4, 2017.
  - 9 Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM*, 36:929–965, 1989.
  - 10 Hervé Brönnimann and Michael T. Goodrich. Almost Optimal Set Covers in Finite VC-Dimension. *Discrete & Computational Geometry*, 1995.
  - 11 Kevin Buchin, Yago Diez, Tom van Diggelen, and Wouter Meulemans. Efficient trajectory queries under the Fréchet distance (GIS Cup). In *Proc. 25th Int. Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 101:1–101:4, 2017.
  - 12 Bernard Chazelle and Emo Welzl. Quasi-Optimal Range Searching in Spaces of Finite VC-Dimension. *Discrete and Computational Geometry*, 4:467–489, 1989.
  - 13 Monika Csikos, Andrey Kupavskii, and Nabil H. Mustafa. Optimal Bounds on the VC-dimension. *arXiv:1807.07924*, 2018. arXiv:1807.07924.
  - 14 Mark De Berg, Atlas F Cook, and Joachim Gudmundsson. Fast Fréchet queries. *Computational Geometry*, 46(6):747–755, 2013.
  - 15 Mark de Berg and Ali D. Mehrabi. Straight-Path Queries in Trajectory Data. In *WALCOM: Algorithms and Computation - 9th Int. Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings*, pages 101–112, 2015. doi:10.1007/978-3-319-15612-5\_10.
  - 16 Anne Driemel, Amer Krivošija, and Christian Sohler. Clustering time series under the Fréchet distance. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 766–785, 2016. doi:10.1137/1.9781611974331.ch55.
  - 17 Anne Driemel, Jeff M. Phillips, and Ioannis Psarros. The VC dimension of metric balls under Fréchet and Hausdorff distances. *CoRR*, arXiv:1903.03211, 2019. arXiv:1903.03211.
  - 18 Anne Driemel and Francesco Silvestri. Locally-sensitive hashing of curves. In *33rd International Symposium on Computational Geometry, SoCG 2017*, pages 37:1–37:16, 2017.
  - 19 Fabian Dütsch and Jan Vahrenhold. A Filter-and-Refinement- Algorithm for Range Queries Based on the Fréchet Distance (GIS Cup). In *Proc. 25th Int. Conference on Advances in Geographic Information Systems (SIGSPATIAL)*, pages 100:1–100:4, 2017.
  - 20 Ioannis Z. Emiris and Ioannis Psarros. Products of Euclidean Metrics and Applications to Proximity Questions among Curves. In *Proc. 34th Int. Symposium on Computational Geometry (SoCG)*, volume 99 of *LIPICs*, pages 37:1–37:13, 2018.
  - 21 Alexander Gilbers and Rolf Klein. A new upper bound for the VC-dimension of visibility regions. *Computational Geometry: Theory and Applications*, 74:61–74, 2014.

- 22 Paul W. Goldberg and Mark R. Jerrum. Bounding the Vapnik-Chervonenkis Dimension of Concept Classes Parameterized by Real Numbers. *Machine Learning*, 18:131–148, 1995.
- 23 Joachim Gudmundsson and Michiel Smid. Fast algorithms for approximate Fréchet matching queries in geometric trees. *Computational Geometry*, 48(6):479–494, 2015. doi:10.1016/j.comgeo.2015.02.003.
- 24 Sarel Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society, Boston, MA, USA, 2011.
- 25 Lingxiao Huang, Shaofeng Jiang, Jian Li, and Xuan Wu. Epsilon-Coresets for Clustering (with Outliers) in Doubling Metrics. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 814–825. IEEE, 2018.
- 26 Sarang Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian. Comparing Distributions and Shapes Using the Kernel Distance. In *ACM SoCG*, 2011.
- 27 Marek Karpinski and Angus Macintyre. Polynomial bounds for VC dimension of sigmoidal neural networks. In *STOC*, 1995.
- 28 Elmar Langetepe and Simone Lehmann. Exact VC-dimension for L1-visibility of points in simple polygons. *arXiv:1705.01723*, 2017. arXiv:1705.01723.
- 29 Frank Olken. *Random Sampling in Databases*. PhD thesis, University of California at Berkeley, 1993.
- 30 Norbert Sauer. On the Density of Families of Sets. *Journal of Combinatorial Theory Series A*, 13:145–147, 1972.
- 31 Saharon Shelah. A Combinatorial Problem; Stability and Order for Models and Theories in Infinitary Languages. *Pacific Journal of Mathematics*, 41(1), 1972.
- 32 Pavel Valtr. Guarding Galleries Where No Point Sees a Small Area. *Israel Journal of Mathematics*, 104:1–16, 1998.
- 33 Vladimir Vapnik and Alexey Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to their Probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.
- 34 Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.
- 35 Martin Werner and Dev Oliver. ACM SIGSPATIAL GIS Cup 2017: Range queries under Fréchet distance. *SIGSPATIAL Special*, 10(1):24–27, June 2018. doi:10.1145/3231541.3231549.

# Dual Circumference and Collinear Sets

Vida Dujmović

School of Computer Science and Electrical Engineering, University of Ottawa, Canada  
<https://cglab.ca/~vida/>  
vida.dujmovic@uottawa.ca

Pat Morin

School of Computer Science, Carleton University, Canada  
<https://cglab.ca/~morin/>  
morin@scs.carleton.ca

---

## Abstract

---

We show that, if an  $n$ -vertex triangulation  $T$  of maximum degree  $\Delta$  has a dual that contains a cycle of length  $\ell$ , then  $T$  has a non-crossing straight-line drawing in which some set, called a *collinear set*, of  $\Omega(\ell/\Delta^4)$  vertices lie on a line. Using the current lower bounds on the length of longest cycles in 3-regular 3-connected graphs, this implies that every  $n$ -vertex planar graph of maximum degree  $\Delta$  has a collinear set of size  $\Omega(n^{0.8}/\Delta^4)$ . Very recently, Dujmović et al. (SODA 2019) showed that, if  $S$  is a collinear set in a triangulation  $T$  then, for any point set  $X \subset \mathbb{R}^2$  with  $|X| = |S|$ ,  $T$  has a non-crossing straight-line drawing in which the vertices of  $S$  are drawn on the points in  $X$ . Because of this, collinear sets have numerous applications in graph drawing and related areas.

**2012 ACM Subject Classification** Mathematics of computing → Graph theory; Mathematics of computing → Extremal graph theory; Human-centered computing → Graph drawings

**Keywords and phrases** Planar graphs, collinear sets, untangling, column planarity, universal point subsets, partial simultaneous geometric drawings

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.29

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1811.03427>.

**Funding** This work was partly funded by NSERC and MRI.

**Acknowledgements** Much of this research took place during the Sixth Workshop on Order and Geometry held in Ciężka, Poland, September 19–22, 2018. The authors are grateful to the organizers, Stefan Felsner and Piotr Micek, and to the other participants for providing a stimulating research environment.

## 1 Introduction

Throughout this paper, all graphs are simple and finite and have at least 4 vertices. For a planar graph  $G$ , we say that a set  $S \subseteq V(G)$  is a *collinear set* if  $G$  has a non-crossing straight-line drawing in which the vertices of  $S$  are all collinear. A *plane graph* is a planar graph  $G$  along with a particular non-crossing drawing of  $G$ . The *dual*  $G^*$  of a plane graph  $G$  is the graph whose vertex set  $V(G^*)$  is the set of faces in  $G$  and in which  $fg \in E(G^*)$  if and only if the faces  $f$  and  $g$  of  $G$  have at least one edge in common. The *circumference*,  $c(G)$ , of a graph  $G$  is the length of the longest cycle in  $G$ . In Section 2, we prove the following:

► **Theorem 1.** *Let  $T$  be a triangulation of maximum degree  $\Delta$  whose dual  $T^*$  has circumference  $\ell$ . Then  $T$  has a collinear set of size  $\Omega(\ell/\Delta^4)$ .*

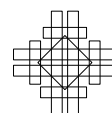
The dual of a triangulation is a 3-connected cubic planar graph. The study of the circumference of 3-connected cubic planar graphs has a long and rich history going back to at least 1884 when Tait [27] conjectured that every such graph is Hamiltonian. In 1946, Tait’s conjecture was disproved by Tutte who gave a non-Hamiltonian 46-vertex example [28].



© Vida Dujmović and Pat Morin;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 29; pp. 29:1–29:17



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



Repeatedly replacing vertices of Tutte’s graph with copies of itself gives a family of graphs,  $\langle G_i : i \in \mathbb{Z} \rangle$  in which  $G_i$  has  $46 \cdot 45^i$  vertices and circumference at most  $45 \cdot 44^i$ . Stated another way,  $n$ -vertex members of the family have circumference  $O(n^\alpha)$ , for  $\alpha = \log_{44}(45) < 0.9941$ . The current best upper bound of this type is due to Grünbaum and Walther [18] who construct a 24-vertex non-Hamiltonian cubic 3-connected planar graph, resulting in a family of graphs in which  $n$ -vertex members have circumference  $O(n^\alpha)$  for  $\alpha = \log_{23}(22) < 0.9859$ .

A series of results has steadily improved the lower bounds on the circumference of  $n$ -vertex (not necessarily planar) 3-connected cubic graphs. Barnette [5] showed that, for every  $n$ -vertex 3-connected cubic graph  $G$ ,  $c(G) = \Omega(\log n)$ . Bondy and Simonovits [8] improved this bound to  $e^{\Omega(\sqrt{\log n})}$  and conjectured that it can be improved to  $\Omega(n^\alpha)$  for some  $\alpha > 0$ . Jackson [19] confirmed this conjecture with  $\alpha = \log_2(1 + \sqrt{5}) - 1 > 0.6942$ . Billings et al. [6] improved this to the solution of  $4^{1/\alpha} - 3^{1/\alpha} = 2$ , which implies  $\alpha > 0.7532$ . The current record is held by Liu, Yu, and Zhang [22] who show that  $\alpha > 0.8$ .

It is known that any planar graph of maximum degree  $\Delta$  can be triangulated so that the resulting triangulation has maximum degree  $\lceil 3\Delta/2 \rceil + 11$  [21]. This fact, together with Theorem 1 and the result of Liu, Yu, and Zhang [22], implies the following corollary:

► **Corollary 2.** *Every  $n$ -vertex triangulation of maximum degree  $\Delta$  contains a collinear set of size  $\Omega(n^{0.8}/\Delta^4)$ .*

It is known that every planar graph  $G$  has a collinear set of size  $\Omega(\sqrt{n})$  [9, 13]. Corollary 2 therefore improves on this bound for bounded-degree planar graphs and, indeed for the family of  $n$ -vertex planar graphs of maximum degree  $\Delta \in O(n^\delta)$ , with  $\delta < 0.075$ . For example, the triangulations dual to Grünbaum and Walther’s construction have maximum degree  $\Delta \in O(\log n)$ . As discussed below, this implies that there exists  $n$ -vertex triangulations of maximum degree  $O(\log n)$  whose largest collinear set has size  $O(n^{0.9859})$ . Corollary 2 implies that every  $n$ -vertex planar graph of maximum degree  $O(\log n)$  has a collinear set of size  $\Omega(n^{0.8})$ .

Recently, Dujmović et al. [14] have shown that every collinear set is *free*. That is, for any planar graph  $G$ , any collinear set  $S \subseteq V(G)$ , and any set  $X \subset \mathbb{R}^2$  with  $|X| = |S|$ , there exists a non-crossing straight-line drawing of  $G$  in which the vertices of  $S$  are drawn on the points of  $X$ . Because of this, collinear sets have immediate applications in graph drawing and related areas. For applications of Corollary 2, including untangling [11, 23, 29, 17, 20, 9, 12, 13, 25], column planarity [3, 15, 12, 13], universal point subsets [16, 1, 12, 13], and partial simultaneous geometric drawings [15, 4, 2, 7, 13] the reader is referred to Dujmović [13] and Dujmović et al. [14, Section 1.1]. Corollary 2 gives improved bounds for all of these problems for planar graphs of maximum  $\Delta \in o(n^{0.075})$ .

For example, it is known that every  $n$ -vertex planar geometric graph can be untangled while keeping some set of  $\Omega(n^{0.25})$  vertices fixed [9] and that there are  $n$ -vertex planar geometric graphs that cannot be untangled while keeping any set of  $\Omega(n^{0.4948})$  vertices fixed [10]. Although asymptotically tight bounds are known for paths [11], trees [17], outerplanar graphs [17], planar graphs of treewidth two [25], and planar graphs of treewidth three [12], progress on the general case has been stuck for 10 years due to the fact that the exponent 0.25 comes from two applications of Dilworth’s Theorem. Thus, some substantially new idea appears to be needed. By relating collinear/free sets to dual circumference, the current paper presents an effective new idea. Indeed, Corollary 2 implies that every bounded-degree  $n$ -vertex planar geometric graph can be untangled while keeping  $\Omega(n^{0.4})$  vertices fixed. Note that, even for bounded-degree planar graphs,  $\Omega(n^{0.25})$  was the best previously-known lower bound.

Our work opens two avenues for further progress:

1. Lower bounds on the circumference of 3-regular 3-connected graphs is an active area of research. Indeed, the  $\Omega(n^{0.8})$  lower bound of Liu, Yu, and Zhang [22] is less than a year old. Any further progress on these lower bounds will translate immediately to an improved bound in Corollary 2 and all its applications.
2. It is possible that the dependence on  $\Delta$  can be removed from Theorem 1 and Corollary 2, thus making these results applicable to all planar graphs, regardless of maximum degree.

## 2 Proof of Theorem 1

Let  $G$  be a plane graph. We treat the vertices of  $G$  as points, the edges of  $G$  as closed curves, and the faces of  $G$  as closed sets (so that a face contains all the edges on its boundary and an edge contains both its endpoints). Whenever we consider subgraphs of  $G$  we treat them as having the same embedding as  $G$ . Similarly, if we consider a graph  $G'$  that is homeomorphic<sup>1</sup> to  $G$  then we assume that the edges of  $G'$  – each of which represents a path in  $G$  whose internal vertices all have degree 2 – inherit their embedding from the paths they represent in  $G$ .

Finally, if we consider the dual  $G^*$  of  $G$  then we treat it as a plane graph in which each vertex  $f$  is represented as a point in the interior of the face  $f$  of  $G$  that it represents. The edges of  $G^*$  are embedded so that an edge  $fg$  is contained in the union of the two faces  $f$  and  $g$  of  $G$ , it intersects the interior of exactly one edge of  $G$  that is common to  $f$  and  $g$ , and this intersection consists of a single point.

A *proper good curve*  $C$  for a plane graph  $G$  is a Jordan curve with the following properties:

1. *proper*: for any edge  $xy$  of  $G$ ,  $C$  either contains  $xy$ , intersects  $xy$  in a single point (possibly an endpoint), or is disjoint from  $xy$ ; and
2. *good*:  $C$  contains at least one point in the interior of some face of  $G$ .

Da Lozzo et al. [12] show that proper good curves define collinear sets:

► **Theorem 3.** *In a plane graph  $G$ , a set  $S \subseteq V(G)$  is a collinear set if and only if there is a proper good curve for  $G$  that contains  $S$ .*

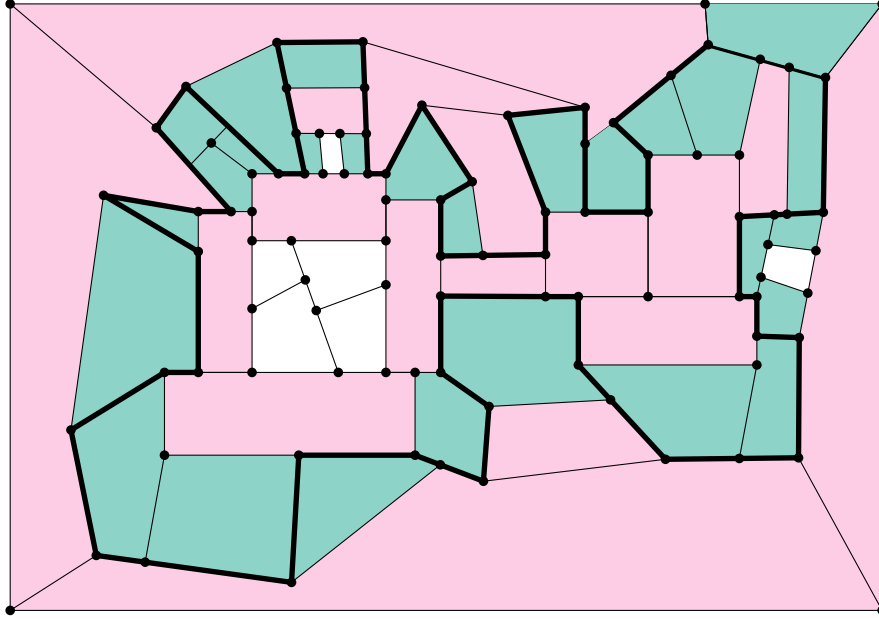
For a triangulation  $T$ , let  $v(T)$  denote the size of a largest collinear set in  $T$ . We will show that, for any triangulation  $T$  of maximum degree  $\Delta$  whose dual is  $T^*$ ,  $v(T) = \Theta(c(T^*)/\Delta^4)$  by relating proper good curves in  $T$  to cycles in  $T^*$ .

As shown by Ravsky and Verbitsky [25, 24], the inequality  $v(T) \leq c(T^*)$  is easy: If  $T$  is a triangulation that has a proper good curve  $C$  containing  $k$  vertices, then a slight deformation of  $C$  produces a proper good curve that contains no vertices. This curve intersects a cyclic sequence of faces  $f_0, \dots, f_{k'-1}$  of  $T$  with  $k' \geq k$ . In this sequence,  $f_i$  and  $f_{(i+1) \bmod k'}$  share an edge, for every  $i \in \{0, \dots, k' - 1\}$ , so this sequence is a closed walk in the dual  $T^*$  of  $T$ . The properness of the original curve and the fact that each face of  $T$  is a triangle ensures that  $f_i \neq f_j$  for any  $i \neq j$ , so this sequence is a cycle in  $T^*$  of length  $k' \geq k$ . Therefore,  $c(T^*) \geq v(T)$ . From the result of Grünbaum and Walther described above, this implies that there are  $n$ -vertex triangulations  $T$  such that  $v(T) = O(n^{0.9859})$ .

The other direction, lower-bounding  $v(T)$  in terms  $c(T^*)$  is more difficult. Not every cycle  $C$  of length  $\ell$  in  $T^*$  can be easily transformed into a proper good curve containing a similar number of vertices in  $C$ . In the next section, we describe three parameters  $\tau$ ,  $\rho$ , and  $\kappa$  of a cycle  $C$  in  $T^*$  and show that  $C$  can always be transformed into a proper good curve containing  $\Omega(\kappa)$  vertices of  $T$ .

<sup>1</sup> We say that a graph  $G'$  is homeomorphic to  $G$  if  $G'$  can be obtained from  $G$  by repeatedly contracting an edge of  $G$  that is incident to a degree-2 vertex.





■ **Figure 1** Faces of  $T^*$  that are pinched and caressed by  $C$ .  $C$  is bold, caressed faces are teal, pinched faces are pink, and untouched faces are unshaded.

## 2.1 Faces that are Touched, Pinched, and Caressed

Throughout the remainder of this section,  $T$  is a triangulation whose dual is  $T^*$  and  $C$  is a cycle in  $T^*$ . Refer to Figure 1 for the following definitions. We say that a face  $f$  of  $T^*$

1. is *touched* by  $C$  if  $f \cap C \neq \emptyset$ ;
2. is *pinched* by  $C$  if  $f \cap C$  is a cycle or has more than one connected component; and
3. is *caressed* by  $C$  if it is touched but not pinched by  $C$ .

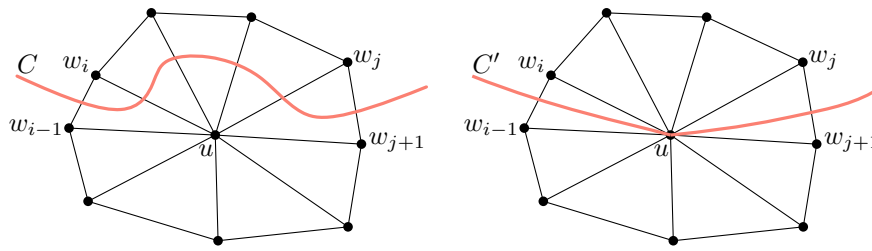
Since  $C$  is almost always the cycle of interest, we will usually say that a face  $f$  of  $T^*$  is touched, pinched, or caressed, without specifically mentioning  $C$ . We will frequently use the values  $\tau$ ,  $\rho$ , and  $\kappa$  to denote the number of faces of  $T^*$  in some region that are  $\tau$ ouched,  $\rho$ inched or  $\kappa$ aressed. Observe that, since every face that is touched is either pinched or caressed, we have the identity  $\tau = \rho + \kappa$ .

► **Lemma 4.** *If  $C$  caresses  $\kappa$  faces of  $T^*$  then  $T$  has a proper good curve that contains at least  $\kappa/4$  vertices so, by Theorem 3,  $v(T) \geq \kappa/4$ .*

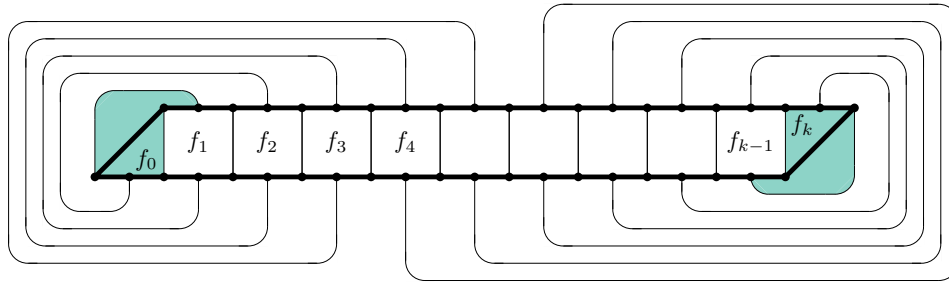
**Proof.** Let  $F$  be the set of faces in  $T^*$  that are caressed by  $C$ . Each element  $u \in F$  corresponds to a vertex of  $T$  so we will treat  $F$  as a set of vertices in  $T$ . Consider the subgraph  $T[F]$  of  $T$  induced by  $F$ . The graph  $T[F]$  is planar and has  $\kappa$  vertices. Therefore, by the 4-Colour Theorem [26],  $T[F]$  contains an independent set  $F' \subseteq F$  of size at least  $\kappa/4$ .

We claim that there is a proper good curve for  $T$  that contains all the vertices in  $F'$ . To see this, first observe that the cycle  $C$  in  $T^*$  already defines a proper good curve (that does not contain any vertices of  $T$ ) that we also call  $C$ . We perform local modifications on  $C$  so that it contains all the vertices in  $F'$ .

For any vertex  $u \in F'$ , let  $w_0, \dots, w_{d-1}$  denote the neighbours of  $u$  in cyclic order. The curve  $C$  intersects some contiguous subsequence  $uw_i, \dots, uw_j$  of the edges adjacent to  $u$ . Since  $u$  is caressed, this sequence does not contain all edges incident to  $u$ . Therefore, the curve  $C$  crosses the edge  $w_{i-1}w_i$ , then crosses  $uw_i, \dots, uw_j$ , and then crosses the edge  $w_jw_{j+1}$ . We



■ **Figure 2** Transforming the dual cycle  $C$  into a proper good curve  $C'$  containing  $u$ .



■ **Figure 3** A Hamiltonian cycle  $C$  in  $T^*$  that caresses only four faces.

modify  $C$  by removing the portion between the first and last of these crossings and replacing it with a curve that contains  $u$  and is contained in the two triangles  $w_{i-1}uw_i$  and  $w_juw_{j+1}$ . (See Figure 2.)

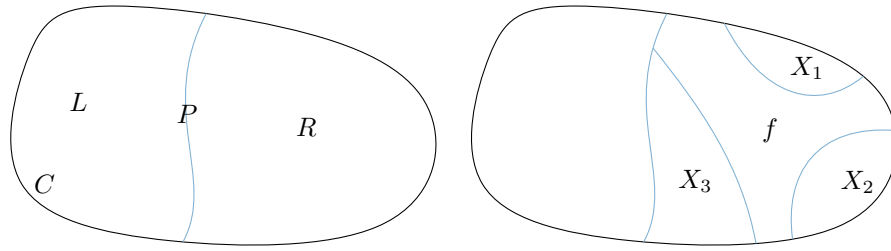
After performing this local modification for each  $u \in F'$  we have a curve  $C'$  that contains every vertex  $u \in F'$ . All that remains is verify that  $C'$  is good and proper for  $T$ . That  $C'$  is good for  $T$  is obvious. That  $C'$  is proper for  $T$  follows from the following two observations: (i)  $C'$  does not contain any two adjacent vertices (since  $F'$  is an independent set); and (ii) if  $C'$  contains a vertex  $u$ , then it does not intersect the interior of any edge incident to  $u$ . ◀

Lemma 4 reduces our problem to finding a cycle in  $T^*$  that caresses many faces. It is tempting to hope that any sufficiently long cycle in  $T^*$  caresses many faces, but this is not true; Figure 3 shows that even a Hamiltonian cycle  $C$  in  $T^*$  may caress only four faces, two inside  $C$  and two outside of  $C$ . In this example, there is an obvious sequence of faces  $f_0, \dots, f_k$ , all contained in the interior of  $C$  where  $f_i$  shares an edge with  $f_{i+1}$  for each  $i \in \{0, \dots, k-1\}$ . The only faces caressed by  $C$  are the endpoints  $f_0$  and  $f_k$  of this sequence.

Our strategy is to define a tree structure,  $T_0$  on groups of faces contained in the interior of  $C$  and a similar structure,  $T_1$  on groups of faces in the exterior of  $C$ . We will then show that every leaf of  $T_0$  or  $T_1$  contains a face caressed by  $C$ . In Figure 3, the tree  $T_0$  is the path  $f_0, \dots, f_k$  and, indeed, the leaves  $f_0$  and  $f_k$  of this tree are caressed by  $C$ . After a non-trivial amount of analysis of the trees  $T_0$  and  $T_1$ , we will eventually show that, if  $C$  does not caress many faces, then  $T_0$  and  $T_1$  have many nodes, but few leaves. Therefore  $T_0$  and  $T_1$  have many degree-2 nodes. This abundance of degree-2 nodes makes it possible to perform a surgery on  $C$  that increases the number of caressed faces. Performing this surgery repeatedly will then produce a curve  $C$  that caresses many faces.

A path  $P = v_1, \dots, v_r$  in  $T^*$  is a *chord path* (for  $C$ ) if  $v_1, v_r \in V(C)$  and  $v_2, \dots, v_{r-1} \notin V(C)$ . Note that this definition implies that the interior vertices  $v_2, \dots, v_{r-1}$  of  $P$  are either all contained in the interior of  $C$  or all contained in the exterior of  $C$ .





■ **Figure 4** The proof of Lemma 5.

► **Lemma 5.** *Let  $P$  be a chord path for  $C$  and let  $L$  and  $R$  be the two faces of  $P \cup C$  that each contain  $P$  in their boundary. Then  $R$  contains at least one face of  $T^*$  that is caressed by  $C$ .*

**Proof.** The proof is by induction on the number,  $t$ , of faces of  $T^*$  contained in  $R$ . If  $t = 1$ , then  $R$  is a face of  $T^*$  and it is caressed by  $C$ .

If  $t > 1$ , then consider the face  $f$  of  $T^*$  that is contained in  $R$  and has the first edge of  $P$  on its boundary. Refer to Figure 4. Since  $t > 1$ ,  $X = R \setminus f$  is non-empty. The set  $X$  may have several connected components  $X_1, \dots, X_k$ , but each  $X_i$  has a boundary that contains a chord path  $P_i$  for  $C$ . We can therefore apply induction on  $P_1$  (or any  $P_i$ ) using  $R = X_1$  in the inductive hypothesis. ◀

## 2.2 Auxilliary Graphs and Trees: $H$ , $\tilde{H}$ , $T_0$ , and $T_1$

Refer to Figure 5. Consider the auxilliary graph  $H$  with vertex set  $V(H) \subseteq V(T^*)$  and whose edge set consist of the edges of  $C$  plus those edges of  $T^*$  that belong to any face pinched by  $C$ . Let  $v_0, \dots, v_{r-1}$  be the clockwise cyclic sequence of vertices on some face  $f$  of  $T^*$  that is pinched by  $C$ . We identify three kinds of vertices that are *special* with respect to  $f$ :

1. A vertex  $v_i$  is special of *Type A* if  $v_{i-1}v_i$  is an edge of  $C$  and  $v_iv_{i+1}$  is not an edge of  $C$ .
2. A vertex  $v_i$  is special of *Type B* if  $v_{i-1}v_i$  is not an edge of  $C$  and  $v_iv_{i+1}$  is an edge of  $C$ .
3. A vertex  $v_i$  is special of *Type Y* if  $v_i$  not incident to any edge of  $C$  and  $v_i$  has degree 3 in  $H$ .

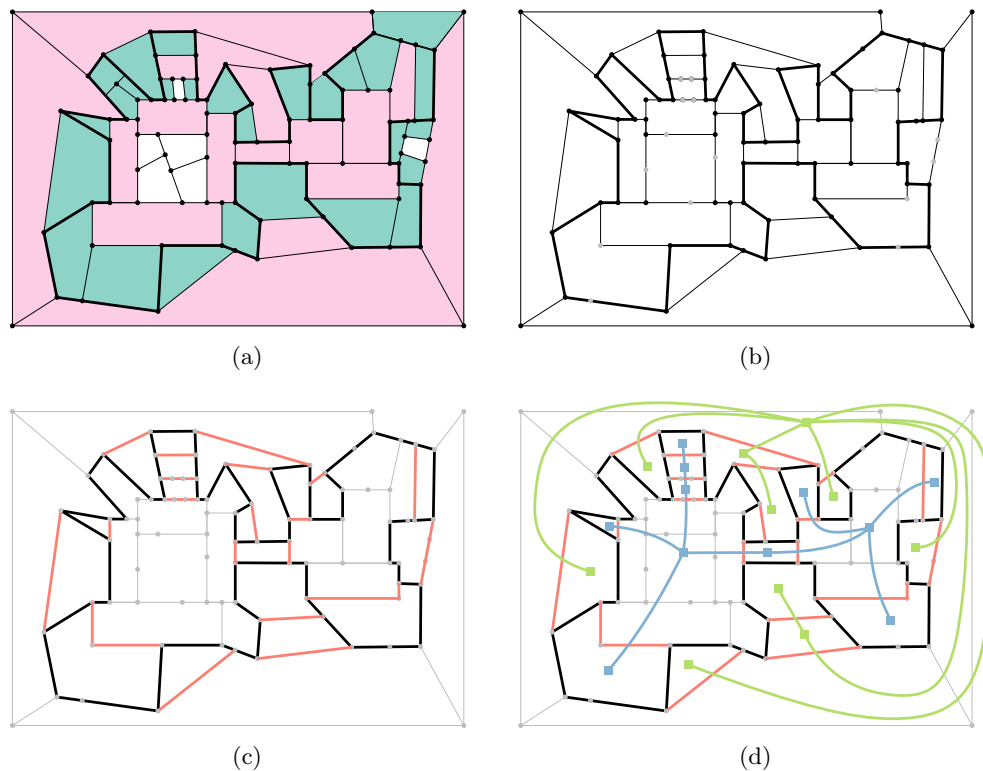
We say that a chord path  $v_i, \dots, v_j$  is a *keeper* with respect to  $f$  if  $v_i$  is special of Type A,  $v_j$  is special of Type B, and none of  $v_{i+1}, \dots, v_{j-1}$  are special. We let  $\tilde{H}$  denote the subgraph of  $H$  containing all the edges of  $C$  and all the edges of all paths that are keepers with respect to some pinched face  $f$  of  $T^*$ .

It is worth emphasizing at this point that, by definition, every keeper is entirely contained in the boundary of at least one face  $f$  of  $T^*$ . This property will be useful shortly.

Let  $\tilde{H}'$  denote the graph that is homeomorphic to  $\tilde{H}$  but does not contain any degree 2 vertices. That is,  $\tilde{H}'$  is the minor of  $\tilde{H}$  obtained by repeatedly contracting an edge incident a degree-2 vertex. The graph  $\tilde{H}'$  naturally inherits an embedding from the embedding of  $\tilde{H}$ . This embedding partitions the edges of  $\tilde{H}'$  into three sets:

1. The set  $B$  of edges that are contained in (the embedding of)  $C$ ;
2. The set  $E_0$  of edges whose interiors are contained in the interior of (the embedding of)  $C$ ;  
and
3. The set  $E_1$  of edges whose interiors are contained in the exterior of (the embedding of)  $C$ .

Observe that, for each  $i \in \{0, 1\}$ , the graph  $H_i$  whose edges are exactly those in  $B \cup E_i$  is outerplanar, since all vertices of  $H_i$  are on a single face, whose boundary is  $C$ . Let  $H_i^*$  be dual of  $H_i$  and let  $T_i$  be the subgraph of  $H_i^*$  whose edges are all those dual to the edges of  $E_i$ . From the outerplanarity of  $H_i$ , it follows that  $T_i$  is a tree.



■ **Figure 5** (a) the cycle  $C$  in  $T^*$  with faces classified as pinched or caressed; (b) the auxiliary graph  $H$ ; (c) the auxiliary graph  $\tilde{H}$  with keeper paths highlighted; (d) the trees  $T_0$  and  $T_1$ .

Each vertex of  $T_i$  corresponds to a face of  $\tilde{H}$ . From this point onwards, we will refer to the vertices of  $T_i$  as *nodes* to highlight this fact, so that a node  $u$  of  $T_i$  is synonymous with the subset of  $\mathbb{R}^2$  contained in the corresponding face of  $\tilde{H}$ . In the following, when we say that a node  $u$  of  $T_i$  contains a face  $f$  of  $T^*$  we mean that  $f$  is one of the faces of  $T^*$  whose union makes up  $u$ . The degree,  $\delta_u$  of any node  $u$  in  $T_i$  is exactly equal to the number of keeper paths on the boundary of  $u$ .

The following lemma allows us to direct our effort towards proving that one of  $T_0$  or  $T_1$  has many leaves.

► **Lemma 6.** *Each leaf  $u$  of  $T_i$  contains at least one face of  $T^*$  that is caressed by  $C$ .*

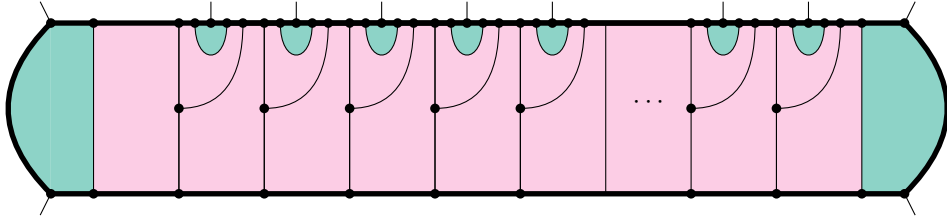
**Proof.** The edge of  $T_i$  incident to  $u$  corresponds to a chord path  $P$ . The graph  $P \cup C$  has two faces with  $P$  on its boundary, one of which is  $u$ . The lemma now follows immediately from Lemma 5, with  $R = u$ . ◀

We will make use of the following well-known property of 3-connected plane graphs.

► **Lemma 7.** *If  $T$  has  $n \geq 4$  vertices then any two faces of  $T^*$  share at most one edge.*

► **Lemma 8.** *Let  $u$  be a node of  $T_i$  and let  $\rho_u$ ,  $\kappa_u$ , and  $\delta_u$  denote the number of pinched faces of  $T^*$  in  $u$ , the number of caressed faces of  $T^*$  in  $u$ , and the degree of  $u$  in  $T_i$ , respectively. Then  $\rho_u \leq 2(\kappa_u + \delta_u)$ .*

Before proving Lemma 8, we point out that the leading constant 2 is tight. Figure 6 shows an example in which all  $\rho_u = 2k + 1$  pinched faces of  $T^*$  are contained in a single (pink) node  $u$  of  $T_0$  that contains  $\kappa_u = 0$  caressed faces and has degree  $\delta_u = k + 2$ .



■ **Figure 6** An example showing the tightness of Lemma 8.

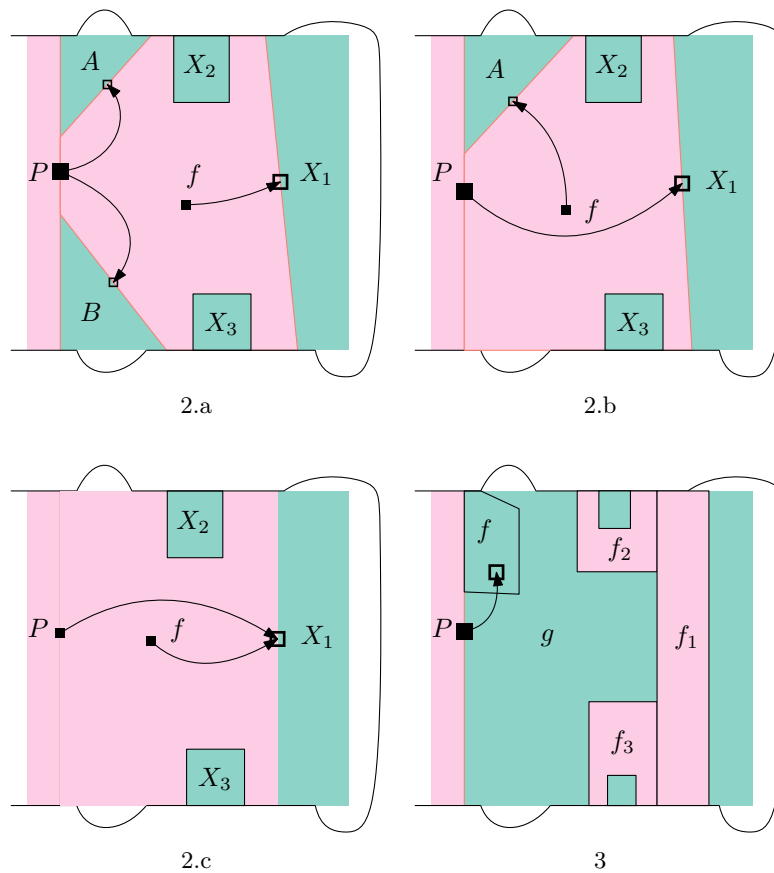
**Proof.** The proof is a discharging argument. We assign each pinched face in  $u$  a single unit of charge, so that the total charge is  $\rho_u$ . We then describe a discharging procedure that preserves the total charge. After executing this procedure, pinched faces in  $u$  have no charge, each caressed face in  $u$  has charge at most 2, and each keeper path in  $u$  has charge at most 2. Since there is a bijection between keeper paths in  $u$  and edges of  $T_i$  incident to  $u$ , this proves the result.

We now describe the discharging procedure, which is recursive and takes as input a chord path  $P$  that partitions  $u$  into two parts  $L$  and  $R$ . We require as a precondition that there are  $m \geq 1$  pinched faces of  $T^*$  in  $L$ , each of which contains at least one edge of  $P$  and such that every edge of  $P$  is contained in at least one of these faces. During a recursive call,  $P$  may have a charge  $c \in \{0, 1, 2\}$ . This charge will be at most 1 if  $m > 1$ , but can be 2 if  $m = 1$ .

To initialize the discharging procedure, we choose an arbitrary pinched face  $f$  contained in  $u$ . The face  $f$  begins with one unit of charge and has  $r \geq 2$  chord paths  $P_1, \dots, P_r$  on its boundary. We move the charge from  $f$  onto  $P_1$  and apply the recursive procedure to  $P_1$ , with a charge of 1 (with  $L$  being the component of  $u \setminus P_1$  that contains  $f$ ). We then recursively apply the discharging procedure on each of  $P_2, \dots, P_r$  with a charge of 0.

Next we describe each recursive step, during which we are given  $P$  with some charge  $c \in \{0, 1, 2\}$ . There are several cases to consider (see Figure 7):

1.  $R$  contains no face of  $T^*$  that is pinched by  $C$ . If  $R$  is empty, then  $P$  is a keeper path, in which case we leave a charge of  $c$  on it and we are done. Otherwise  $R$  is non-empty and Lemma 5 ensures that  $R$  contains at least one caressed face  $f$ . We move the charge from  $P$  onto  $f$  and we are done.
2.  $R$  contains a face  $f$  that is pinched by  $C$  and that shares at least one edge with  $P$ . We consider three subcases:
  - a.  $f$  contains neither endpoint of  $P$ . In this case,  $R \setminus f$  has at least three connected components,  $A$ ,  $B$ , and  $X_1, \dots, X_k$ , where  $A$  and  $B$  each contain an endpoint of  $P$  and each  $X_i$  has a chord path  $P_i$  in common with  $f$ . We recurse on each of these components so that each of these components takes the place of  $R$  in the recursion. When recursing on  $A$  we take one unit of charge from  $P$  (if needed) and place it on  $A$ 's chord path. When recursing on  $B$  we take the second unit of charge from  $P$  (if needed) and place it on  $B$ 's chord path. When recursing on  $X_1$  we move the unit of charge from  $f$  to  $P_1$ . When recursing on  $X_2, \dots, X_k$  we use no charge on  $P_2, \dots, P_k$ .
  - b.  $f$  contains exactly one endpoint of  $P$ . In this case,  $R \setminus f$  has one connected component  $A$  that contains an endpoint of  $P$  and one or more connected components  $X_1, \dots, X_k$  where each  $X_i$  has a chord path  $P_i$  on the boundary of  $f$ . The path  $P$  has a charge  $c \leq 2$ . When recursing on  $X_1$  we assign all of  $P$ 's charge to the chord path  $P_1$ , which is contained in the single pinched face  $f$ . When recursing on  $A$  we move the single unit of charge from  $f$  to the chord path of  $A$ .



■ **Figure 7** Discharging steps in the proof of Lemma 8.

c.  $f$  contains both endpoints of  $P$ . We claim that, in this case,  $P$  must be on the boundary of more than one pinched faces in  $L$ , otherwise  $P$  would be a keeper path. To see this, observe that the face  $f$  contains both the first edge  $e_1$  and last edge  $e_2$  of  $P$ . If  $e_1 = e_2$  because  $P$  is a single edge, then it is certainly a keeper, which is not possible. Otherwise, by Lemma 7,  $e_1$  and  $e_2$  are on the boundary of two different faces in  $L$ . By assumption, both of these faces are pinched by  $C$ .

Therefore  $P$  has at most one unit of charge assigned to it. Now,  $R \setminus f$  has one or more connected components  $X_1, \dots, X_k$  sharing chord paths  $P_1, \dots, P_k$  with  $f$  on which we recurse. When recursing on  $X_1$  we move the charge from  $P$  and the charge from  $f$  to  $P_1$ . When recursing on the remaining  $X_i$ ,  $i \in \{2, \dots, k\}$  we assign no charge to  $P_i$ .

3.  $R$  contains at least one pinched face, but no pinched face in  $R$  shares an edge with  $P$ . In this case, consider the face  $g$  of  $H$  that is contained in  $R$  and has  $P$  on its boundary. By definition,  $g$  contains no pinched faces of  $T^*$ , but  $g$  is touched by  $C$ , so  $g$  contains at least one caressed face<sup>2</sup>  $f$  of  $T^*$ . We move the  $c$  units of charge from  $P$  onto  $f$ .

Now,  $R$  still contains one or more pinched faces  $f_1, \dots, f_k$ , where each  $f_i$  shares part of a chord path  $P_i$  with  $g$ . On each such face  $f_i$ , we run the initialization procedure described above except that we recurse only on the chord paths of  $f_i$  that do not share edges with  $g$ . i.e., we do not recurse on the chord path  $P_i$ .

This completes the description of the discharging procedure, and the proof. ◀

<sup>2</sup> In fact  $g$  contains at least two caressed faces, one for each endpoint of  $P$ .

### 2.3 Bad Nodes

We say that a node of  $T_i$  is *bad* if it has degree 2 and contains no face of  $T^*$  that is caressed by  $C$ . We now move from studying individual nodes of  $T_0$  and  $T_1$  to studying global quantities associated with  $T_0$  and  $T_1$ . From this point on, for each  $i \in \{0, 1\}$ ,

1.  $\tau_i$ ,  $\rho_i$ , and  $\kappa_i$  refer the total numbers of faces contained in nodes of  $T_i$  that are touched, pinched, and caressed by  $C$ , respectively;
2.  $n_i$  refers to the number of nodes of  $T_i$ ;
3.  $\delta_i = 2(n_i - 1)$  is the total degree of all nodes in  $T_i$ ; and
4.  $b_i$  is the number of bad nodes in  $T_i$ .

► **Lemma 9.** *If  $\kappa_i \leq \tau_i/6$  then  $n_i \geq \tau_i/8$ .*

**Proof.** From Lemma 8 we know  $\rho_i \leq 2(\kappa_i + \delta_i)$ , so

$$\tau_i = \kappa_i + \rho_i \leq 3\kappa_i + 2\delta_i = 3\kappa_i + 4(n_i - 1) \leq \tau_i/2 + 4n_i ,$$

and reorganizing the left- and right-hand sides gives the desired result. ◀

► **Lemma 10.** *For any  $0 < \epsilon < 1$ , if  $b_i \leq (1 - \epsilon)n_i$ , then  $\kappa_i = \Omega(\epsilon\tau_i)$ .*

**Proof.** Partition the nodes of  $T_i$  into the following sets:

1. the set  $B$  of bad nodes;
2. the set  $N_1$  of leaves;
3. the set  $N_{\geq 3}$  of nodes having degree at least 3;
4. the set  $N_2$  of nodes having degree 2 that are not bad.

$$\begin{aligned} b_i &= n_i - |N_1| - |N_{\geq 3}| - |N_2| \\ &> n_i - 2|N_1| - |N_2| && \text{since } |N_1| > |N_{\geq 3}| \\ &\geq n_i - 2\kappa_i - |N_2| && \text{(since, by Lemma 6, } \kappa_i \geq |N_1|) \\ &\geq n_i - 3\kappa_i && \text{(since each node in } N_2 \text{ contains a caressed face)} \end{aligned}$$

Thus, we have

$$n_i - 3\kappa_i \leq b_i \leq (1 - \epsilon)n_i$$

and rewriting gives

$$\kappa_i \geq \epsilon n_i / 3 . \tag{1}$$

If  $\kappa_i \geq \tau_i/6$ , then the proof is complete. On the other hand, if  $\kappa_i \leq \tau_i/6$  then, by Lemma 9,  $n_i \geq \tau_i/8$ . Combining this with (1) gives

$$\kappa_i \geq \epsilon n_i / 3 \geq \epsilon \tau_i / 24 = \Omega(\epsilon \tau_i) . \quad \blacktriangleleft$$

### 2.4 Interactions Between Bad Nodes

We have now reached a point in which we know that the vast majority of nodes in  $T_0$  and  $T_1$  are bad nodes, otherwise Lemma 10 implies that a constant fraction of the faces touched by  $C$  are caressed by  $C$ . At this point, we are ready to study interactions between bad nodes of  $T_0$  and bad nodes of  $T_1$ . The proof of the following lemma is omitted due to space constraints but can be found in the full version of the paper.

► **Lemma 11.** *If  $u$  is a bad node then there is a single face  $f$  of  $T^*$  that is contained in  $u$  and that contains all edges of  $C \cap u$ .*

The following lemma shows that a bad node  $u$  in  $T_0$  and a bad node  $w$  in  $T_1$  share at most one edge of  $C$ .

► **Lemma 12.** *If nodes  $u$  in  $T_0$  and  $w$  in  $T_1$  are bad nodes that share at least one edge of  $C$ , then  $u$  and  $w$  share exactly one edge of  $C$ .*

**Proof.** Suppose  $u$  and  $w$  share two edges  $e_1$  and  $e_2$  of  $C$ . Then, by Lemma 11, there is a common face  $f_u$  in  $u$  that contains  $e_1$  and  $e_2$ . Similarly, there is a common face  $f_w$  contained in  $w$  that contains both  $e_1$  and  $e_2$ . But this contradicts Lemma 7. ◀

► **Lemma 13.** *If  $u$  and  $w$  are bad nodes of  $T_i$  sharing a common chord path  $P$ , then  $P$  is a single edge.*

**Proof.** By Lemma 11,  $u$  and  $w$  have the first edge of  $P$  in common and the last edge of  $P$  in common. Lemma 7 therefore implies that the first and last edge of  $P$  are the same, so  $P$  has only one edge. ◀

## 2.5 Really Bad Nodes

At this point we will start making use of the assumption that the triangulation  $T$  has maximum degree  $\Delta$ , which is equivalent to the assumption that each face of  $T^*$  has at most  $\Delta$  edges on its boundary.

► **Observation 14.** *If  $T$  has maximum degree  $\Delta$  and  $C$  has length  $\ell$ , then the number of faces  $\tau$  of  $T^*$  touched by  $C$  is at least  $2\ell/\Delta$ . At least  $\ell/\Delta$  of these faces are in the interior of  $C$  and at least  $\ell/\Delta$  of these faces are in the exterior of  $C$ .*

**Proof.** Orient the edges of  $C$  counterclockwise so that, for each edge  $e$  of  $C$ , the face of  $T^*$  to the left of  $e$  is in  $C$ 's interior and the face of  $T^*$  to the right of  $e$  is in  $C$ 's exterior. Each face of  $T^*$  has at most  $\Delta$  edges. Therefore, the number of faces to the right of edges in  $C$  is at least  $\ell/\Delta$ . The same is true for the number of faces of  $T^*$  to the left of edges in  $C$ . ◀

For a node  $u$  of  $T_i$ , we define  $N(u)$  as the set of nodes in  $T_0$  and  $T_1$  (excluding  $u$ ) that share an edge of  $T^*$  with  $u$ . Note that  $N(u)$  contains the neighbours of  $u$  in  $T_i$  as well as nodes of  $T_{1-i}$  with which  $u$  shares an edge of  $C$ . We say that a node  $u$  is *really bad* if  $u$  and all nodes in  $N(u)$  are bad. The proof of the following lemma – which is similar to that of Lemma 10 – is omitted due to space constraints.

► **Lemma 15.** *For every sufficiently small  $0 < \alpha < 1/2$ , if  $T$  has maximum degree  $\Delta$ ,  $C$  has length  $\ell$ , and the number  $\kappa$ , of faces caressed by  $C$  is at most  $\alpha\ell/\Delta^2$ , then the number of really bad nodes in  $T_0$  is at least  $n_0 - O(\alpha n_0)$ .*

For a node  $u$  of  $T_i$ , we define  $N^0(u) = \{u\}$  and, for any  $r \in \mathbb{N}$ , we define  $N^r(u) = \bigcup_{w \in N^{r-1}(u)} N(w)$ . We say that a node  $u$  in  $T_i$  is *really<sup>r</sup> bad* if  $u$  is bad and all nodes in  $N^r(u)$  are bad. The proof of the following lemma is a straightforward generalization of the proof of Lemma 15.

► **Lemma 16.** *For any constant  $i \in \mathbb{N}$  and every sufficiently small  $0 < \alpha < 1/2$ , if  $T$  has maximum degree  $\Delta$ ,  $C$  has length  $\ell$ , and the number,  $\kappa$ , of faces caressed by  $C$  is at most  $\alpha\ell/\Delta^{i+1}$ , then the number of really<sup>i</sup> bad nodes in  $T_0$  is at least  $n_0 - O(\alpha n_0)$ .*

For our purposes, it will be sufficient to work with bad ( $i = 0$ ), really bad ( $i = 1$ ), and really really bad ( $i = 2$ ) nodes.

## 2.6 Tree/Cycle Surgery

We summarize the situation so far. We are left with the case where  $C$  has length  $\ell$  and therefore touches  $\Omega(\ell/\Delta)$  faces. To complete the proof of Theorem 1 we must deal with the situation where  $C$  caresses  $o(\ell/\Delta^4)$  faces and therefore each of  $T_0$  and  $T_1$  has  $o(\ell/\Delta^4)$  leaves (Lemma 6),  $\Omega(\ell/\Delta)$  nodes (Lemma 9), and the fraction of really really bad nodes in  $T_0$  and  $T_1$  is  $1 - O(1/\Delta^2)$  (Lemma 16).

To handle cases like these, the only option is to perform surgery on the cycle  $C$  to increase the number of caressed nodes. In particular, our strategy is to perform modifications to  $C$  that increase the number of faces caressed by  $C$ . At this point we are ready to complete the proof of Theorem 1.

**Proof of Theorem 1.** By Lemma 4, it suffices to prove the existence of a cycle  $C$  in  $T^*$  that caresses  $\Omega(\ell/\Delta^4)$  faces. We begin by applying Lemma 16 with  $i = 2$  and  $\alpha = \epsilon/\Delta$ . For sufficiently small, but constant,  $\epsilon$ , Lemma 16 implies that  $\kappa = \Omega(\ell/\Delta^4)$  or the number of nodes in  $T_0$  that are not really really bad is at most  $O(\epsilon n_0/\Delta)$ . In the former case,  $C$  caresses  $\Omega(\ell/\Delta^4)$  faces of  $T^*$  and we are done.

In the latter case, consider the forest obtained by removing all nodes of  $T_0$  that are not really really bad. This forest has  $(1 - O(\epsilon/\Delta))n_0$  nodes. We claim that it also has  $O(\epsilon n_0/\Delta)$  components. To see why this is so, let  $L$  be the set of leaves in  $T_i$  and let  $S$  be the set of non-leaf nodes in  $T_i$  that are not really really bad. Observe that it is sufficient to upper bound the number,  $k$  of components in  $T_i - S$ .

We have  $|L| \leq |S| + |L| = O(\epsilon n_0/\Delta)$  and  $k = \sum_{u \in S} (\deg_{T_i}(u) - 1)$ . Furthermore,

$$O(\epsilon n_0/\Delta) \geq |L| \geq \sum_{u \in S} (\deg_{T_i}(u) - 2) = |S| + \sum_{u \in S} (\deg_{T_i}(u) - 1) = |S| + k .$$

Therefore  $k \leq |S| + k = O(\epsilon n_0/\Delta)$ , as claimed.

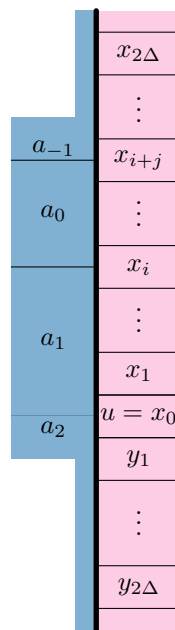
Thus the forest obtained by removing all really really bad nodes from  $T_i$  has at most  $O(\epsilon n_0/\Delta)$  components, each of which is a path. At least one of these paths contains  $\Omega(\Delta/\epsilon)$  nodes. In particular, for a sufficiently small constant  $\epsilon$ , one of these components,  $X$ , has at least  $5\Delta$  nodes.

Consider some node  $u$  in  $X$ , and let  $C_a$  and  $C_b$  be the two components of  $u \cap C$ . Observe that  $T_1[N(u)]$  consists of two paths  $a_1, \dots, a_r$  and  $b_1, \dots, b_s$  of bad nodes where each  $a_1, \dots, a_r$  contains an edge of  $C_a$  and each of  $b_1, \dots, b_s$  contains an edge of  $C_b$ . Note that it is possible that  $a_i = b_j$  for some values of  $i$  and  $j$ , but everything stated thus far, and subsequently, is still true. It follows from Lemma 12 that among any sequence of  $\Delta$  consecutive nodes in  $X$ , at least one node has  $r \geq 2$  and therefore  $|N(u)| \geq 5$ . Let  $u$  be any such node that is not among the first  $2\Delta$  or last  $2\Delta$  nodes of  $X$ . Such a  $u$  always exists because  $X$  contains at least  $5\Delta$  nodes.

Let  $x_0 = u$ . We now describe some of the nodes in the vicinity of  $u$  (refer to Figure 8):

1. there is a path  $x_{2\Delta}, \dots, x_1, x_0, y_1, \dots, y_{2\Delta}$  in  $T_0$  consisting entirely of really really bad nodes.
2. some really bad node  $a_1$  of  $T_1$  shares an edge with each of  $x_0, \dots, x_i$  for some  $i \in \{1, \dots, \Delta - 4\}$ .
3. some really bad node  $a_2$  of  $T_1$  shares an edge with  $a_1$  and an edge with  $x_0$ .
4. some really bad node  $a_0$  of  $T_1$  shares an edge with  $a_1$  and with each of  $x_i, \dots, x_{i+j}$  for some  $j \in \{0, \dots, \Delta - 4\}$ .

The surgery we perform focuses on the nodes  $u$  and  $a_1$ . Consider the two components of  $C \cap a_1$ . One of these components,  $p$ , shares an edge with  $u$ . By Lemma 12, the other component,  $q$ , does not share an edge with  $u$ . Imagine removing  $u$  from  $T_0$ , thereby separating



■ **Figure 8** Nodes in the vicinity of  $u = x_0$ .

$T_0$  into a component  $T_x$  containing  $x_1$  and a component  $T_y$  containing  $y_1$ . Equivalently, one can think of removing the edges of  $u$  from  $C$  separating  $C$  into two paths  $C_x$  and  $C_y$  on the boundary of  $T_x$  and  $T_y$ , respectively. We distinguish between two major cases (see Figure 9):

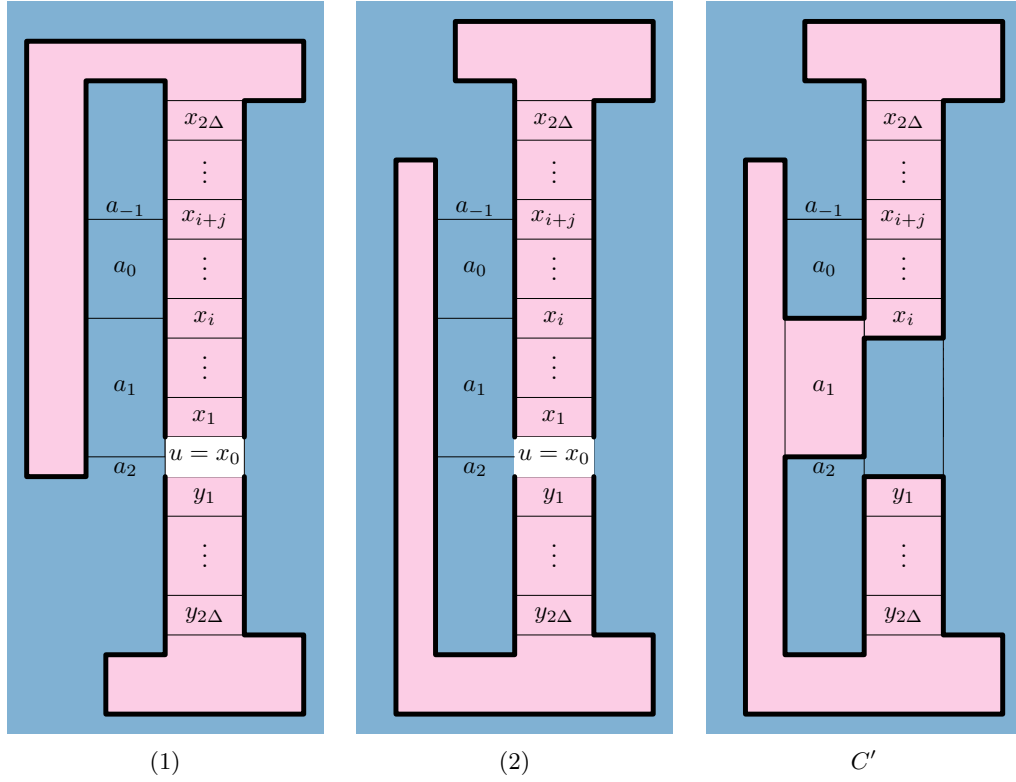
1.  $q \subset C_x$ . In this case, we punt to Case 2. By Lemma 13  $a_1 - C$  consists of two edges and exactly one of these edges,  $e$ , is not incident to  $u$ . Instead,  $e$  is incident to  $x_i$ . We set  $u' = x_i$ ,  $x'_1 = x_{i-1}$ ,  $y'_1 = x_{i+1}$ , and  $a'_1 = a_1$ . Observe that  $a'_1$  connects the two components of  $T_0 - u'$  and shares edges with  $u'$  and  $x'_1$ . This is exactly the situation considered in Case 2.
2.  $q \subset C_y$ . At this point it is helpful to think of  $T_0$ ,  $T_1$ , and  $C$  as a partition of  $\mathbb{R}^2$ , where nodes of  $T_0$  are coloured red, nodes of  $T_1$  are coloured blue and  $C$  is the (purple) boundary between red and blue. To describe our modifications of  $C$ , we imagine changing the colours of nodes. The effect that such a recolouring has on  $C$  is immediately obvious: It produces a 1-dimensional set  $C'$  that contains every (purple) edge contained in the red-blue boundary. The set  $C'$  is a collection of vertices and edges of  $T^*$ . Therefore, if  $C'$  is a simple cycle, then  $C'$  defines a new pair of trees  $T'_0$  and  $T'_1$ .

Refer to the right two thirds of Figure 9 for a simple (and misleading) example of what follows. For a full example, refer to Figure 10. The surgery we perform, recolours  $x_0, x_1, \dots, x_{i-1}$  blue and recolours  $a_1$  red. Observe that, because  $q \subset C_y$  and  $p$  contain an edge of  $x_1$ , this implies that the red subset of  $\mathbb{R}^2$  is simply-connected and its boundary  $C'$  is a simple cycle consisting of edges of  $T^*$ . The new trees  $T'_0$  and  $T'_1$  are therefore well defined. We now make two claims that will complete our proof.

▷ **Claim 17.** For each  $i \in \{0, 1\}$ , and every node of  $w T_i$  that is not bad,  $C \cap w = C' \cap w$ . (Equivalently, for every face  $f$  of  $T^*$  that is not a bad node of  $T_0$  or  $T_1$ ,  $C \cap f = C' \cap f$ .)

▷ **Claim 18.** The face  $a_0$  is caressed by  $C'$ .





■ **Figure 9** Cases 1 and 2 in the proof of Theorem 1 and the surgery performed in Case 2.

These two claims complete the proof because, together, they imply that  $C'$  caresses one more node than  $C$ . Indeed, by definition,  $C$  did not caress any faces belonging to bad nodes. Therefore, the first claim implies that the faces of  $T^*$  caressed by  $C'$  are a superset of those caressed by  $C$ . The face  $a_0$  is a bad node of  $T_i$  so it is not caressed by  $C$  but the second claim states that it is caressed by  $C'$ . Therefore  $C'$  caresses at least one more face than  $C$ .

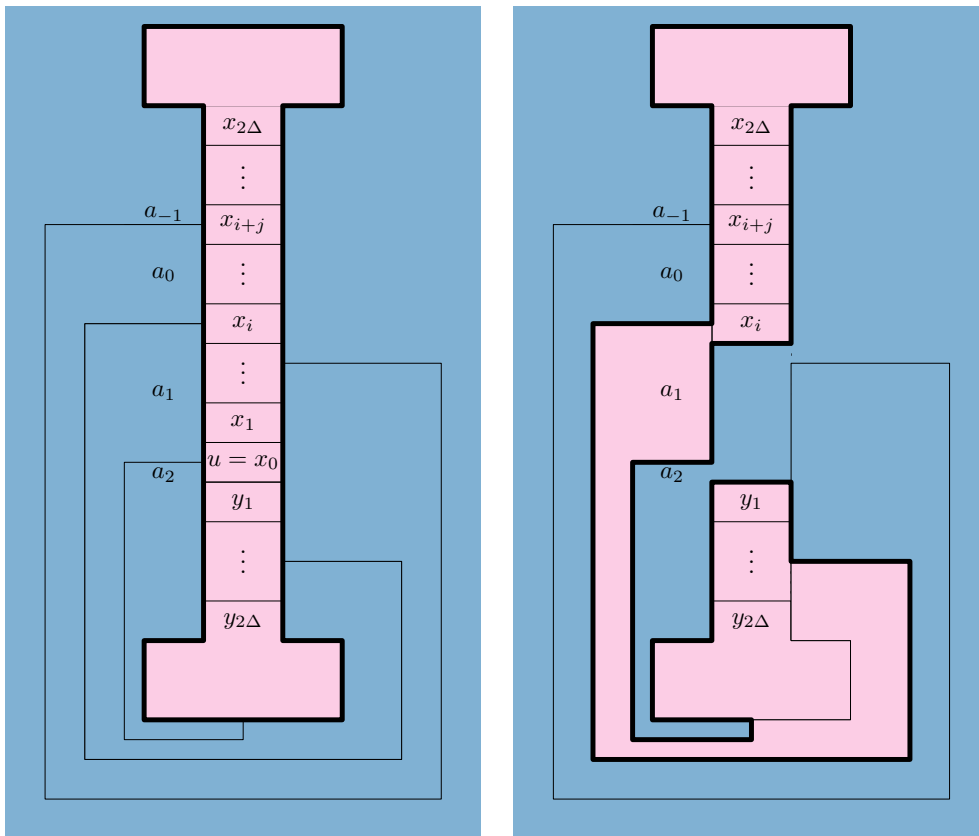
This surgery recolours at most  $\Delta - 2 \leq \Delta$  nodes of  $T_0$  and  $T_1$ , so the difference in length between  $C$  and  $C'$  is at most  $\Delta^2$ . If we start with a cycle  $C$  of length  $\ell$ , then we can perform this surgery at least  $\ell/(4\Delta^2)$  times before the length of  $C$  decreases to less than  $\ell' = \ell/2$ . If at some point during this process, we are no longer able to perform this operation, it is because  $C$  caresses  $\Omega(\ell'/\Delta^4) = \Omega(\ell/\Delta^4)$  faces of  $T^*$  and we are done. By the end of this process, the number of faces caressed by  $C$  is at least  $\ell/(4\Delta^2) \in \Omega(\ell/\Delta^2) \subset \Omega(\ell/\Delta^4)$  and we are also done.

Thus, all that remains is to prove Claim 17 and Claim 18.

To prove Claim 17 we observe that  $C$  and  $C'$  differ only on the boundaries of nodes that are recoloured. Thus, it is sufficient to show that all nodes in  $R = \cup\{N(v) : v \in \{x_0, \dots, x_{i-1}, a_1\}\}$  are bad. But this is immediate since  $x_0, \dots, x_{i-1}$  are really really bad and  $a_1 \in N(x_0)$ , so  $a_1$  is bad. Since every node in  $R$  share an edge with at least one of  $\{x_0, \dots, x_{i-1}, a_1\}$ , every node in  $R$  is therefore bad, as required.

To prove Claim 18 we consider the boundary of the face  $a_0$  of  $T^*$  after the recolouring operation. This boundary consists of, in cyclic order:

- a. An edge shared between  $a_0$  and  $a_1$ . This edge is in  $C'$  since  $a_0$  is in  $T'_1$  and  $a_1$  is in  $T'_0$ .
- b. A path of edges shared with  $x_i, \dots, x_{i+j}$ . The nodes  $x_i, \dots, x_{i+j}$  are in  $T_0$  and are distinct from  $x_0, \dots, x_{i-1}$ , so these nodes are in  $T'_0$ . Therefore, this part of the boundary of  $a_0$  is contained in  $C'$ .



■ **Figure 10** Performing surgery on  $C$  to obtain  $C'$  that caresses  $a_0$ .

- c. An edge shared between  $a_0$  and another node  $a_{-1} \neq a_1$  of  $T_1$ . The faces of  $a_{-1}$  are in  $T'_1$  because  $a_1$  is the only face that moves from  $T_1$  to  $T'_0$ . ( $T'_1$  is the only face whose colour goes from blue to red.)
- d. A path of edges that contains at least one edge of  $C_y$ . If we fix an embedding in which the outer face is some face of  $T_1$  other than  $a_0$ , then this path contains a portion of  $C$  that is traversed in clockwise order. By Lemma 12 This path does not contain any edge of  $x_i$ . Furthermore, this path does not contain any edges of  $x_0, x_1, \dots, x_{i-1}$  that are not on the outer face of  $T_0 \cup a_1$ . Therefore, this path consists of a (possibly empty) sequence of edges that are shared with  $x_0, \dots, x_{i-1}$  followed by a sequence of edges from  $C_y$ . The former part of this path is shared with nodes in  $T'_1$ , so these edges are not in  $C'$ . The latter part of this path is shared with nodes in  $T_y$ , which are all contained contained in  $T'_0$ .

Therefore the intersection  $C' \cap a_0$  consists of one connected component so  $a_0$  is caressed by  $C'$ . ◀

### 3 Discussion

It remains an open problem to eliminate the dependence of our results on the maximum degree,  $\Delta$ , of  $T$ . The next significant step is to resolve the following conjecture:

► **Conjecture 19.** *If  $T$  is a triangulation whose dual  $T^*$  has a cycle of length  $\ell$ , then  $T^*$  has a cycle that caresses  $\Omega(\ell)$  faces. (Therefore, by Lemma 4 and Theorem 3,  $T$  has a collinear set of size  $\Omega(\ell)$ .)*

## References

- 1 Patrizio Angelini, Carla Binucci, William S. Evans, Ferran Hurtado, Giuseppe Liotta, Tamara Mchedlidze, Henk Meijer, and Yoshio Okamoto. Universal Point Subsets for Planar Graphs. In Kun-Mao Chao, Tsan-sheng Hsu, and Der-Tsai Lee, editors, *Algorithms and Computation - 23rd International Symposium, ISAAC 2012, Taipei, Taiwan, December 19-21, 2012. Proceedings*, volume 7676 of *Lecture Notes in Computer Science*, pages 423–432. Springer, 2012. doi:10.1007/978-3-642-35261-4\_45.
- 2 Patrizio Angelini, William S. Evans, Fabrizio Frati, and Joachim Gudmundsson. SEFE without mapping via large induced outerplane graphs in plane graphs. *Journal of Graph Theory*, 82(1):45–64, 2016. doi:10.1002/jgt.21884.
- 3 Luis Barba, William Evans, Michael Hoffmann, Vincent Kusters, Maria Saumell, and Bettina Speckmann. Column planarity and partially-simultaneous geometric embedding. *J. Graph Algorithms Appl.*, 21(6):983–1002, 2017. doi:10.7155/jgaa.00446.
- 4 Luis Barba, Michael Hoffmann, and Vincent Kusters. Column planarity and partial simultaneous geometric embedding for outerplanar graphs. In *Abstracts of the 31st European Workshop on Computational Geometry (EuroCG)*, pages 53–56, 2015.
- 5 David Barnette. Trees in polyhedral graphs. *Canadian Journal of Mathematics*, 18:731–736, 1966.
- 6 Mark Bilinski, Bill Jackson, Jie Ma, and Xingxing Yu. Circumference of 3-connected claw-free graphs and large Eulerian subgraphs of 3-edge connected graphs. *J. Combin. Theory Ser. B*, 101:214–236, 2011.
- 7 T. Bläsius, S. G. Kobourov, and I. Rutter. Simultaneous embedding of planar graphs. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, pages 349–381. Chapman and Hall/CRC Press, 2013.
- 8 J.A. Bondy and M. Simonovits. Longest cycles in 3-connected 3-regular graphs. *Canadian Journal of Mathematics*, 32:987–992, 1980.
- 9 Prosenjit Bose, Vida Dujmović, Ferran Hurtado, Stefan Langerman, Pat Morin, and David R. Wood. A Polynomial Bound for Untangling Geometric Planar Graphs. *Discrete & Computational Geometry*, 42(4):570–585, 2009. doi:10.1007/s00454-008-9125-3.
- 10 Javier Cano, Csaba D. Tóth, and Jorge Urrutia. Upper Bound Constructions for Untangling Planar Geometric Graphs. *SIAM J. Discrete Math.*, 28(4):1935–1943, 2014. doi:10.1137/130924172.
- 11 Josef Cibulka. Untangling Polygons and Graphs. *Discrete & Computational Geometry*, 43(2):402–411, 2010. doi:10.1007/s00454-009-9150-x.
- 12 Giordano Da Lozzo, Vida Dujmović, Fabrizio Frati, Tamara Mchedlidze, and Vincenzo Roselli. Drawing planar graphs with many collinear vertices. *Journal of Computational Geometry*, 9(1):94–130, 2018.
- 13 Vida Dujmović. The Utility of Untangling. *J. Graph Algorithms Appl.*, 21(1):121–134, 2017. doi:10.7155/jgaa.00407.
- 14 Vida Dujmović, Fabrizio Frati, Daniel Gonçalves, Pat Morin, and Günter Rote. Every Collinear Set is Free. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA19)*, 2019.
- 15 William Evans, Vincent Kusters, Maria Saumell, and Bettina Speckmann. Column Planarity and Partial Simultaneous Geometric Embedding. In Christian A. Duncan and Antonios Symvonis, editors, *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*, volume 8871 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2014. doi:10.1007/978-3-662-45803-7\_22.
- 16 Emilio Di Giacomo, Giuseppe Liotta, and Tamara Mchedlidze. How many vertex locations can be arbitrarily chosen when drawing planar graphs? *CoRR*, abs/1212.0804, 2012. arXiv:1212.0804.

- 17 Xavier Goaoc, Jan Kratochvíl, Yoshio Okamoto, Chan-Su Shin, Andreas Spillner, and Alexander Wolff. Untangling a Planar Graph. *Discrete & Computational Geometry*, 42(4):542–569, 2009. doi:10.1007/s00454-008-9130-6.
- 18 Branko Grünbaum and Hansjoachim Walther. Shortness Exponents of Families of Graphs. *J. Comb. Theory, Ser. A*, 14(3):364–385, 1973. doi:10.1016/0097-3165(73)90012-5.
- 19 Bill Jackson. Longest cycles in 3-connected cubic graphs. *J. Combin. Theory Ser. B*, 41:17–26, 1986.
- 20 Mihyun Kang, Oleg Pikhurko, Alexander Ravsky, Mathias Schacht, and Oleg Verbitsky. Untangling planar graphs from a specified vertex position - Hard cases. *Discrete Applied Mathematics*, 159(8):789–799, 2011. doi:10.1016/j.dam.2011.01.011.
- 21 Goos Kant and Hans L. Bodlaender. Triangulating Planar Graphs while Minimizing the Maximum Degree. *Inf. Comput.*, 135(1):1–14, 1997. doi:10.1006/inco.1997.2635.
- 22 Qinghai Liu, Xingxing Yu, and Zhao Zhang. Circumference of 3-connected cubic graphs. *J. Comb. Theory, Ser. B*, 128:134–159, 2018. doi:10.1016/j.jctb.2017.08.008.
- 23 János Pach and Gábor Tardos. Untangling a Polygon. *Discrete & Computational Geometry*, 28(4):585–592, 2002. doi:10.1007/s00454-002-2889-y.
- 24 Alexander Ravsky and Oleg Verbitsky. On collinear sets in straight line drawings. *CoRR*, abs/0806.0253, 2008. arXiv:0806.0253.
- 25 Alexander Ravsky and Oleg Verbitsky. On Collinear Sets in Straight-Line Drawings. In Petr Kolman and Jan Kratochvíl, editors, *37th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2011)*, volume 6986 of *LNCS*, pages 295–306. Springer, 2011. doi:10.1007/978-3-642-25870-1\_27.
- 26 Neil Robertson, Daniel P. Sanders, Paul D. Seymour, and Robin Thomas. The Four-Colour Theorem. *J. Comb. Theory, Ser. B*, 70(1):2–44, 1997. doi:10.1006/jctb.1997.1750.
- 27 Peter Guthrie Tait. Remarks on the colouring of maps. *Proc. Roy. Soc. Edinburgh Sect. A*, 10:729, 1880.
- 28 W. T. Tutte. On Hamilton circuits. *J. Lond. Math. Soc.*, 21:98–101, 1946.
- 29 Mamoru Watanabe. Open problem. 5th Czech–Slovak Symposium on Combinatorics, 1998.



# A Product Inequality for Extreme Distances

Adrian Dumitrescu 

Department of Computer Science, University of Wisconsin–Milwaukee, USA

<http://www.cs.uwm.edu/faculty/ad/>

dumitres@uwm.edu

---

## Abstract

Let  $p_1, \dots, p_n$  be  $n$  distinct points in the plane, and assume that the minimum inter-point distance occurs  $s_{\min}$  times, while the maximum inter-point distance occurs  $s_{\max}$  times. It is shown that  $s_{\min}s_{\max} \leq \frac{9}{8}n^2 + O(n)$ ; this settles a conjecture of Erdős and Pach (1990).

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures

**Keywords and phrases** Extreme distances, repeated distances

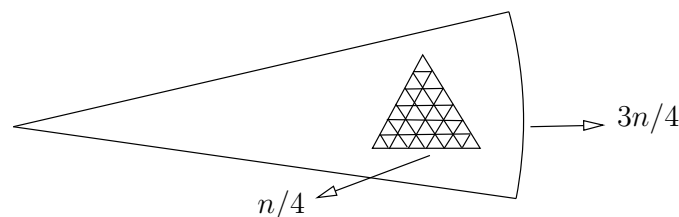
**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.30

## 1 Introduction

Let  $p_1, \dots, p_n$  be  $n$  distinct points in the plane, and assume that the minimum inter-point distance occurs  $s_{\min}$  times, while the maximum inter-point distance occurs  $s_{\max}$  times. It is well-known that  $s_{\min} \leq 3n$  and  $s_{\max} \leq n$ ; see, e.g., [3, Ch. 13]; and these classical bounds immediately imply that  $s_{\min}s_{\max} \leq 3n^2$ . Erdős and Pach [1] asked for a proof or disproof of the following sharper product inequality:

$$s_{\min}s_{\max} \leq \frac{9}{8}n^2 + o(n^2).$$

The authors also remarked that this inequality, if true, essentially cannot be improved; and this would follow from a construction of E. Makai Jr. (not discussed in their paper). Indeed, the main term in the inequality cannot be improved: the point configuration exhibited in Fig. 1 has  $s_{\min} = \frac{3}{4}n + \frac{3}{4}n - O(\sqrt{n}) = \frac{3}{2}n - O(\sqrt{n})$ , and  $s_{\max} = \frac{3}{4}n$  (provided that the circular arc subtends an angle of  $60^\circ$ ), and so  $s_{\min}s_{\max} = \frac{9}{8}n^2 - O(n\sqrt{n})$ . The  $m = \frac{1}{4}n$  interior points make a section of a unit triangular lattice with  $\lfloor 3m - \sqrt{12m - 3} \rfloor$  unit distances, where the minimum inter-point distance is equal to 1; see [2] or [3, p. 211].



**Figure 1** An  $n$ -element point set with  $\frac{3}{4}n$  points on the convex hull and  $\frac{1}{4}n$  interior points.  $\frac{3}{4}n - 1$  boundary points are evenly distributed on a circular arc of radius  $\Theta(\sqrt{n})$  centered at the leftmost point.

Here we prove the claimed inequality in a slightly stronger form (with a linear lower order term).

► **Theorem 1.** *Let  $p_1, \dots, p_n$  be  $n$  distinct points in the plane, and let  $s_{\min}$  and  $s_{\max}$  denote the multiplicity of the minimum and maximum inter-point distance, respectively. Then  $s_{\min}s_{\max} \leq \frac{9}{8}n^2 + O(n)$ .*



© Adrian Dumitrescu;

licensed under Creative Commons License CC-BY

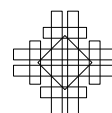
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 30; pp. 30:1–30:12

Leibniz International Proceedings in Informatics



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



## 30:2 A Product Inequality for Extreme Distances

**Definitions and notations.** A convex polygon is one in *strictly convex* position, i.e., no three boundary points are collinear.

**Preliminaries.** Let  $S = \{p_1, \dots, p_n\}$  be a set of  $n$  distinct points in the plane. Given two points  $p$  and  $q$ , let  $\ell(p, q)$  denote the line determined by  $p$  and  $q$ . Let  $\delta$  and  $\Delta$  denote the minimum and maximum pairwise distance of  $S$ , respectively. We may assume that  $\delta = 1$ ; a standard packing argument then yields  $\Delta = \Omega(\sqrt{n})$ . Let  $G_\delta$  and  $G_\Delta$  denote the respective graphs. It is well-known that  $|E(G_\delta)| \leq 3n$  and  $|E(G_\Delta)| \leq n$ ; see, e.g., [3, Ch. 13].

For any point  $u \in S$ , let  $\deg(u)$  denote its degree in  $G_\delta$ ; it is well known that  $\deg(u) \leq 6$  for any  $u \in S$ . For any point  $u \in S$ , let  $\Gamma(u) = \{v \in S : uv \in E(G_\delta)\}$ ; i.e.,  $\Gamma(u)$  is the set of vertices adjacent to  $u$  in  $G_\delta$ . For a point  $u$ , let  $x(u)$  and  $y(u)$  denote its  $x$ - and  $y$ -coordinates respectively.

For a point set  $S$ ,  $\text{conv}(S)$  denotes the convex hull of  $S$ , and  $\partial\text{conv}(S)$  denotes the boundary of  $\text{conv}(S)$ . The perimeter of a polygon  $P$  is denoted by  $\text{per}(P)$ .

### 2 Setup of the proof

Let  $H \subseteq S$  denote the set of (extreme) vertices of  $\text{conv}(S)$  labeled in a clockwise manner:  $H = \{u_1, u_2, \dots, u_h\}$ , and so that indices can be read in a circular fashion, i.e.,  $u_{h+1} = u_1$ . We say that a vertex  $u_i \in H$  has a *flat neighborhood* if the interior angles of the seven vertices  $\{u_{i-3}, u_{i-2}, u_{i-1}, u_{i+1}, u_{i+2}, u_{i+3}\}$  all belong to the interval  $(179^\circ, 180^\circ)$ . Observe that the number of vertices of  $\text{conv}(S)$  that do *not* have flat neighborhoods is  $O(1)$ .

Let  $F \subseteq H$  denote the set of vertices of  $\text{conv}(S)$  that have flat neighborhoods. Let  $D \subseteq H$  denote the set of vertices of  $\text{conv}(S)$  that are endpoints of some diameter pair. Put  $|D| = d$ ,  $f = |F|$ , and recall that  $h = |H|$ ; as such,  $d \leq h$  and  $f \leq h$ .

The set of points  $S$  can be partitioned into three parts as  $S = H \cup H' \cup I$ , where

- $H$  is the set of extreme vertices of  $\text{conv}(S)$ ; an element of  $H$  can be in any of the following sets  $D \cap F$ ,  $D \setminus F$ ,  $F \setminus D$ , or  $S \setminus (D \cup F)$ . Let  $u_1, \dots, u_h$  (where  $u_{h+1} = u_1$ ) be the extreme vertices of  $\text{conv}(S)$  in clockwise order.
- $H'$  is the set of points on  $\partial\text{conv}(S)$  that are not in  $H$  (the interior angle of each vertex in  $H'$  is  $180^\circ$ ).
- $I$  is the set of interior vertices, i.e., those that are not on  $\partial\text{conv}(S)$ .

As mentioned earlier, we have

$$s_{\max} \leq d \leq h. \tag{1}$$

Indeed, the endpoints of any diameter pair must be extreme points on the boundary of  $\text{conv}(S)$ . If  $d \leq n/3$ , then  $s_{\max} \leq d \leq n/3$  and consequently,  $s_{\min}s_{\max} \leq 3n \cdot \frac{1}{3}n = n^2$ , as required (with room to spare). We therefore subsequently assume that  $d \geq n/3$ ; and so we have  $h \geq d \geq n/3$ .

► **Lemma 2.** *If  $h \geq n/3$ , then  $\Delta \geq \frac{n}{3\pi}$ ; in particular  $\Delta = \Omega(n)$ .*

**Proof.** Let  $p = \text{per}(\text{conv}(S))$ ; since  $\delta = 1$  and  $h \geq n/3$ , we have  $p \geq n/3$ . By a standard isoperimetric inequality,  $p \leq \pi\Delta$ ; see, e.g., [4]. Putting the two inequalities together yields  $\Delta \geq \frac{n}{3\pi}$ , as required. ◀

### 3 Charging scheme

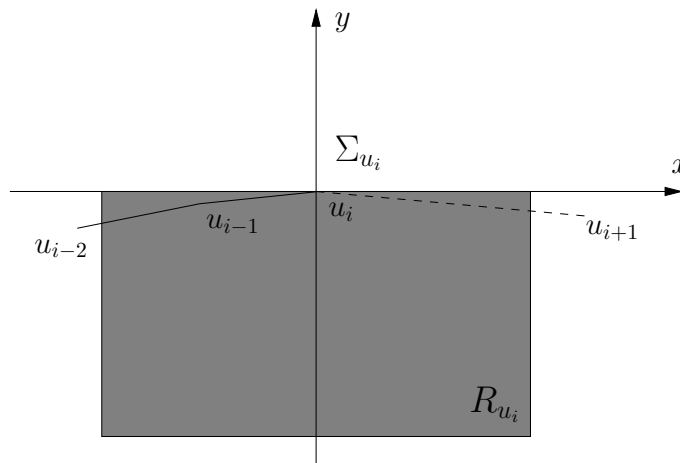
Assume that each point in  $S$  carries an initial charge equal to its degree in  $G_\delta$  (at most 6). The scheme we discuss below transfers a unit charge from each convex hull vertex of degree 3 that belongs to  $D \cap F$  to one or two interior vertices, in such a way that the final charge of each interior vertex is at most 6. This achieves the desired effect that the endpoints on the convex hull of these edges are left with a charge of 2 (while their initial charge was 3). Once this goal is achieved, the upper bound we need on the number of unit distances follows from Lemma 5 (in Section 5).

The main difficulties posed by this plan are (i) deciding how to implement the charging scheme; and (ii) verifying its validity (namely that the final charge of each interior vertex is bounded from above by 6). We next describe the charging scheme, after which we show in Lemma 4 that it works as intended.

#### Overview

Recall that  $u_1, \dots, u_h$  (where  $u_{h+1} = u_1$ ) are the extreme vertices of  $\text{conv}(S)$  in clockwise order. For any extreme vertex with a flat neighborhood  $u_i \in F$ , let  $\Sigma_{u_i}$  be an orthogonal coordinate system whose origin is  $u_i$ , and where the  $x$ -axis is a supporting line of  $\text{conv}(S)$  incident to  $u_i$ , and  $S$  lies in the closed halfplane below the  $x$ -axis. See Fig. 2. More precisely: if  $u_i u_{i+1} \in G_\delta$  and there exists  $v \in I$  s.t.  $u_i v, u_{i+1} v \in G_\delta$  (i.e.,  $\Delta u_i u_{i+1} v$  is an equilateral triangle), the  $x$ -axis will be chosen as  $\overrightarrow{u_i u_{i+1}}$ ; otherwise, the  $x$ -axis will be chosen so that  $S \setminus \{u_i\}$  lies strictly below this line and the bisector of the interior angle  $\angle u_{i-1} u_i u_{i+1}$  is the negative direction of the  $y$ -axis.

Having defined  $\Sigma_{u_i}$ , consider the rectangle  $R_{u_i} = [x(u_i) - 7/4, x(u_i) + 7/4] \times [y(u_i) - 2, y(u_i)]$  in this system. When sending charge from  $u_i$ , a reference will be made to  $R_{u_i}$  (in the details of the charging scheme).



■ **Figure 2** The coordinate system  $\Sigma_{u_i}$  and the axis-aligned rectangle  $R_{u_i}$  for a vertex  $u_i \in H$  with a flat neighborhood.

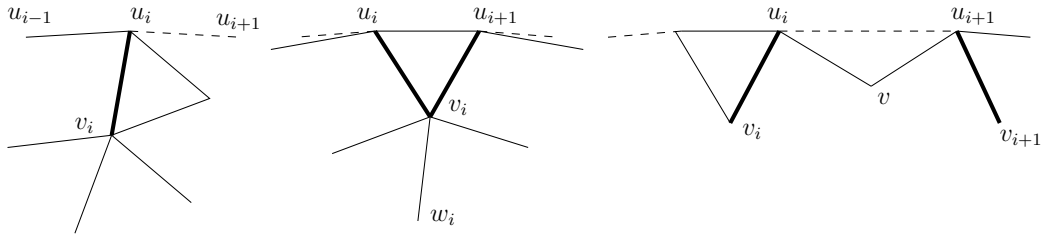
Vertices in  $H = \{u_1, \dots, u_h\}$  are processed one by one in this order (pairs of adjacent vertices of  $H$  corresponding to equilateral triangles in  $G_\delta$  are processed at the same time). Equivalently, we keep the coordinate system fixed (with the two axes horizontal and vertical) and rotate  $S$  counterclockwise so that  $u_i$  is the highest vertex in  $S$  at the time it is processed;



see Fig. 3 (middle) for the case (a) when the  $x$ -axis will be chosen as  $\overrightarrow{u_i u_{i+1}}$ ; and see Fig. 3 (left) for the remaining case (b). In either case, the point set  $S$  is contained in the closed halfplane below the  $x$ -axis.

Let  $u_i \in D \cap F$  be an extreme vertex of degree 3; if the vertices in  $\Gamma(u_i)$  are ordered from left to right, let  $v_i \in \Gamma(u_i)$  be the second (middle) element. We refer to the edge  $u_i v_i$  as the *middle edge* associated (and incident) to  $u_i$ .

(c) If  $u_i v$  and  $u_{i+1} v$  are unit edges incident to  $v$  connecting  $v$  with two non-adjacent extreme vertices  $u_i$  and  $u_{i+1}$  (i.e.,  $|u_i u_{i+1}| > 1$ ), then  $u_i v$  and  $u_{i+1} v$  are *not* middle edges, and so we are in the situation described in (a) or (b); see Fig. 3 (right), where middle edges  $u_i v_i$  and  $u_{i+1} v_{i+1}$  will be those charged to interior vertices.



■ **Figure 3** Illustrations to the scenarios described in (a), (b) and (c). Middle edges are drawn in bold.

**Charging rules.** When handling the current vertex  $u_i$ , or two consecutive vertices  $u_i, u_{i+1}$  that belong to a unit equilateral triangle, we use the coordinate system  $\Sigma_{u_i}$ . Let  $u_i v_i$  be the middle edge from  $u_i$ , where  $v_i$  is an interior vertex. We distinguish four cases, depending on whether (i) the degree of  $v_i$  is 6 or less than 6; and (ii)  $v_i$  is connected to one or two vertices on  $\partial \text{conv}(S)$ . The following charging rules are observed:

1. Every middle edge has its unit charge distributed to one or two interior vertices.
2. Charging amounts can be  $1/2$  or  $1$ : we sometimes transfer the entire unit charge of a middle edge to an interior vertex and sometimes split the unit charge into two equal parts,  $1/2$ , that are sent to two different interior vertices.
3. The unit charge on the middle edge incident to  $u_i$  is distributed to one or two interior vertices at distance at most 2 in  $G_\delta$ ; i.e., this charging process can only affect vertices in  $\Gamma(u_i) \cup \Gamma(\Gamma(u_i))$ .

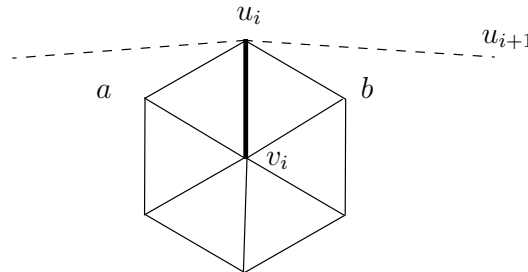
Before the execution of the charging scheme, we clearly have  $s_{\min} = \frac{1}{2} \sum_{p \in S} \deg(p)$ . The charging scheme that is put in place transfers one unit from each extreme vertex of degree 3 that is an element of  $D \cap F$  to one or two interior vertices. After completion,  $s_{\min}$  can be calculated in an alternative way, as half the sum of final charges of all vertices (Lemma 5 in Section 5).

**Details**

**Case 1:**  $\deg(v_i) = 6$ , and  $u_i v_i$  is the unique unit edge incident to  $v_i$  connecting  $v_i$  with an extreme vertex; see<sup>1</sup> Fig. 4. Note that the six vertices in  $\Gamma(v_i)$  form a regular hexagon of unit side-length. Let  $a, b \in \Gamma(u_i) \cap \Gamma(v_i)$  be the other two common neighbors of  $u_i$  and  $v_i$

<sup>1</sup> In all subsequent figures, solid edges are of unit length and middle edges are drawn in bold.

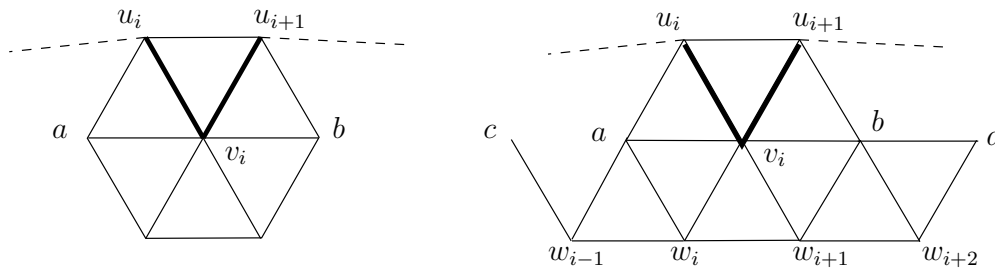
on the left and right, respectively. Note that  $\deg(a) \leq 5$ , and similarly,  $\deg(b) \leq 5$ ; indeed, if  $\deg(a) = 6$  (or  $\deg(b) = 6$ ), one element in  $\Gamma(a)$  (resp.,  $\Gamma(b)$ ) would lie strictly above  $u_i$ , a contradiction. Distribute the unit charge on edge  $u_i v_i$  into two equal parts:  $1/2$  to the left interior vertex  $a$  and  $1/2$  to the right interior vertex  $b$ . Observe that  $a, b \in R_{u_i}$ .



■ **Figure 4** Case 1. All points lie in the closed halfplane below the horizontal line incident to  $u_i$ .

**Case 2:**  $\deg(v_i) = 6$ , where  $u_i v_i$  and  $u_{i+1} v_i$  are unit edges incident to  $v_i$  connecting  $v_i$  with two adjacent extreme vertices  $u_i$  and  $u_{i+1}$ ; see Fig. 5 (left). The argument assumes that both  $u_i$  and  $u_{i+1}$  are elements of  $D \cap F$ , since otherwise, there is no need to transfer charge from the respective unit edges. Note that the six vertices in  $\Gamma(v_i)$  form a regular hexagon of unit side-length. Let  $a \in \Gamma(u_i) \cap \Gamma(v_i)$  be the interior vertex on the left, and  $b \in \Gamma(u_{i+1}) \cap \Gamma(v_i)$  be the interior vertex on the right. Note that  $\deg(a) \leq 5$  and  $\deg(b) \leq 5$ ; indeed, if say,  $\deg(a) = 6$  (or  $\deg(b) = 6$ ), the interior angle at  $u_i$  (resp., at  $u_{i+1}$ ) would be  $180^\circ$ , a contradiction, since we have assumed that  $u_i, u_{i+1} \in D$ .

We further identify other vertices of low degree that will be charged. Let  $w_i, w_{i+1} \in \Gamma(v_i)$  be the two neighbors of  $v_i$  below it, as in Fig. 5 (right). Our charging scheme is symmetric, and here we show how to distribute the unit charge of edge  $u_{i+1} v_i$  to  $b$  and some other interior vertex (the distribution of the unit charge of edge  $u_i v_i$  is analogous, involving  $a$  and some other interior vertex).

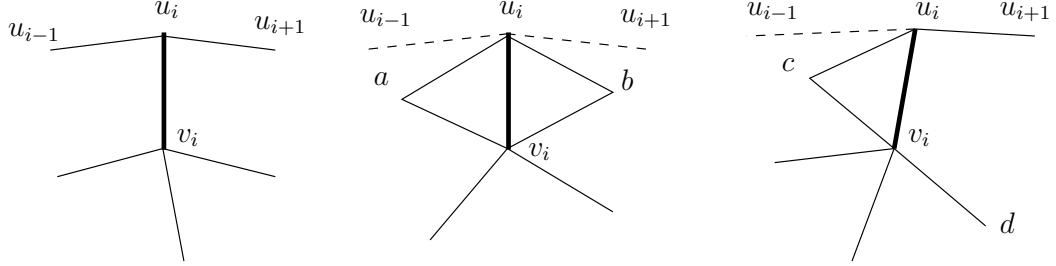


■ **Figure 5** Case 2. All points lie in the closed halfplane below the horizontal line  $\ell(u_i, u_{i+1})$ .

If  $\deg(w_{i+1}) \leq 5$ , distribute the unit charge on edge  $u_{i+1} v_i$  into two equal parts:  $1/2$  to interior vertex  $b$  and  $1/2$  to the interior vertex  $w_{i+1}$ . We subsequently assume that  $\deg(w_{i+1}) = 6$ . Let  $w_{i+2} \in \Gamma(b) \cap \Gamma(w_{i+1})$  be the interior vertex on the line  $\ell(w_i, w_{i+1})$  to the right. If  $\deg(w_{i+2}) \leq 5$ , distribute the unit charge on edge  $u_{i+1} v_i$  into two equal parts:  $1/2$  to interior vertex  $b$  and  $1/2$  to the interior vertex  $w_{i+2}$ . We subsequently assume that  $\deg(w_{i+2}) = 6$ . Let  $d \in \Gamma(b) \cap \Gamma(w_{i+2})$  be the interior vertex on the line  $\ell(v_i, b)$  to the right. Observe that  $\deg(d) \leq 4$ : since each element of  $\Gamma(d) \setminus \{b, w_{i+2}\}$  must lie strictly below the line  $\ell(w_{i+2}, d)$ , there are at most two such vertices. In this last case, distribute the unit charge on edge  $u_{i+1} v_i$  into two equal parts:  $1/2$  to the interior vertex  $b$  and  $1/2$  to the interior vertex  $d$ . Observe that  $b, d, w_{i+1}, w_{i+2} \in R_{u_{i+1}}$ , and similarly that  $a, c, w_i, w_{i-1} \in R_{u_i}$ .

30:6 A Product Inequality for Extreme Distances

**Case 3:**  $\deg(v_i) \leq 5$ , and  $u_i v_i$  is the unique unit edge incident to  $v_i$  connecting  $v_i$  with an extreme vertex; see Fig. 6. If  $\deg(v_i) \leq 4$ , charge edge  $u_i v_i$  to the interior vertex  $v_i$ ; i.e.,  $v_i$  receives a unit charge; see Fig. 6 (left).

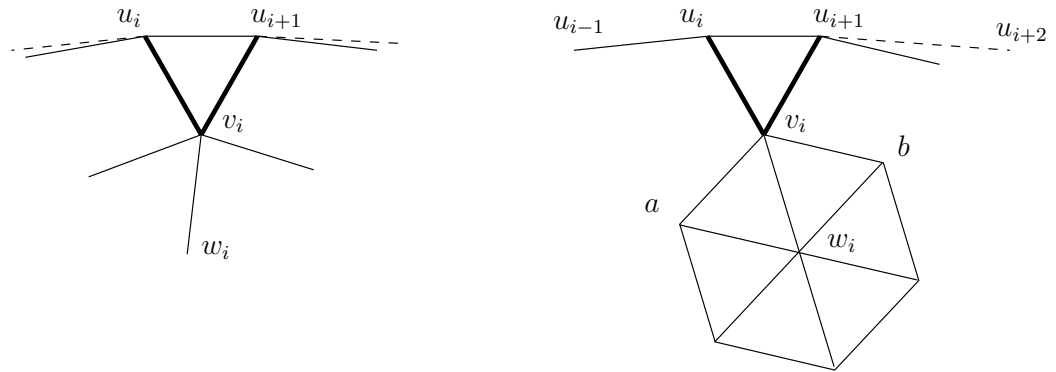


■ **Figure 6** Case 3. Left:  $\deg(v_i) = 4$ . Middle:  $b$  is the highest among  $\{a, b\}$ . Right:  $c$  is the highest among  $\{c, d\}$ . All points lie in the closed halfplane below the horizontal line incident to  $u_i$ .

If  $\deg(v_i) = 5$ , refer to Fig. 6 (middle and right). Let  $a$  and  $b$  be the two neighbors of  $v_i$  left and right of  $u_i$ , respectively; see Fig. 6 (middle). Let  $\text{high}(a, b)$  denote the element of  $\{a, b\}$  which is the highest (i.e., closest to the  $x$ -axis of  $\Sigma_{u_i}$ ). Observe that  $\text{high}(a, b)$  has degree at most 5; since otherwise, the  $y$ -coordinate of one of its neighbors (w.r.t. this coordinate system) would be non-negative, a contradiction. Further observe that  $\text{high}(a, b)u_i$  is an edge in  $G_\delta$ ; since otherwise,  $u_i$  would not have degree 3 or its interior angle would be  $180^\circ$ , either of which is a contradiction. Distribute the unit charge on edge  $u_i v_i$  into two equal parts:  $1/2$  unit to  $v_i$  and  $1/2$  unit to  $\text{high}(a, b)$  (with ties broken arbitrarily). Observe that  $v_i, a, b \in R_{u_i}$ .

**Case 4:**  $\deg(v_i) \leq 5$ , where  $u_i v_i$  and  $u_{i+1} v_i$  are unit edges incident to  $v_i$  connecting  $v_i$  with two adjacent extreme vertices  $u_i$  and  $u_{i+1}$ ; see Fig. 7 (left). If  $\deg(v_i) \leq 4$ , distribute the two unit charge on  $u_i v_i$  and  $u_{i+1} v_i$  to  $v_i$ . Assume now that  $\deg(v_i) = 5$  and let  $w_i$  denote the vertex in  $\Gamma(v_i)$  below  $v_i$  that is farthest from  $\ell(u_i, u_{i+1})$ .

If  $\deg(w_i) \leq 5$ , distribute the two units of charge for edges  $u_i v_i$  and  $u_{i+1} v_i$  into two equal parts: one unit to  $v_i$  and one unit to  $w_i$ . Observe that  $v_i, w_i \in R_{u_i}$ .



■ **Figure 7** Case 4. Left:  $\deg(w_i) = 5$ . Right:  $\deg(w_i) = 6$ . All points lie in the closed halfplane below the horizontal line  $\ell(u_i, u_{i+1})$ .

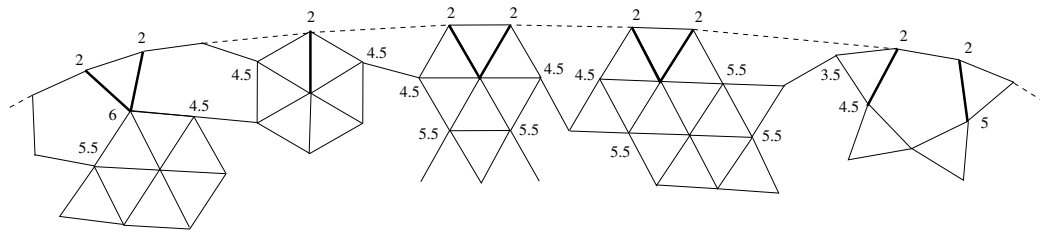
Assume now that  $\deg(w_i) = 6$ ; observe that the six vertices in  $\Gamma(w_i)$  form a regular hexagon of unit side-length. Let  $a, b \in \Gamma(v_i) \cap \Gamma(w_i)$  be as in Fig. 7 (right). We claim that  $\deg(a) \leq 5$  and  $\deg(b) \leq 5$ . We may assume that  $\angle av_i u_i \geq 90^\circ \geq \angle bv_i u_{i+1}$ .

If  $\deg(a) = 6$ , let  $v_{i-1}$  be the next counterclockwise vertex after  $v_i$  in  $\Gamma(a)$ . Since the triangle  $\Delta v_{i-1}v_i$  is equilateral, this implies that  $v_{i-1}v_i$  is yet another edge in  $G_\delta$ , which is in contradiction with the assumption that  $\deg(v_i) = 5$ .

If  $\deg(b) = 6$ , then  $bu_{i+1}$  is an edge in  $G_\delta$ , thus  $v_i b \parallel u_i u_{i+1}$  and so  $v_i b$  is horizontal. Let  $c$  be the next clockwise vertex after  $u_{i+1}$  in  $\Gamma(b)$ . Then  $u_{i+1}c$  is also horizontal, thus  $c \in \partial\text{conv}(S)$ , which implies that the interior angle at  $u_{i+1}$  is  $180^\circ$ , which is a contradiction (we have assumed that  $u_i, u_{i+1} \in D$ ).

Since each of the two assumptions  $\deg(a) = 6$  and  $\deg(b) = 6$  leads to a contradiction, this proves the claim. Distribute the two unit charges for edges  $u_i v_i$  and  $u_{i+1} v_i$  as one unit to  $v_i$ ,  $1/2$  unit to  $a$  and  $1/2$  unit to  $b$ . (This can be also viewed as distributing the unit charge for  $u_i v_i$  as  $1/2$  unit to  $v_i$  and  $1/2$  unit to  $a$ , and distributing the unit charge for  $u_{i+1} v_i$  as  $1/2$  unit to  $v_i$  and  $1/2$  unit to  $b$ .) Observe that  $v_i, a \in R_{u_i}$  and  $v_i, b \in R_{u_{i+1}}$ .

**Illustration.** An example illustrating the final charges in a few representative cases is shown in Fig. 8.



**Figure 8** Charging illustrations for vertices on the upper hull; middle edges adjacent to extreme vertices of degree 3 in  $D \cap F$  are drawn in thick lines.

#### 4 Charging scheme analysis

By direct inspection of the scheme we note the two properties announced prior to describing the charging scheme:

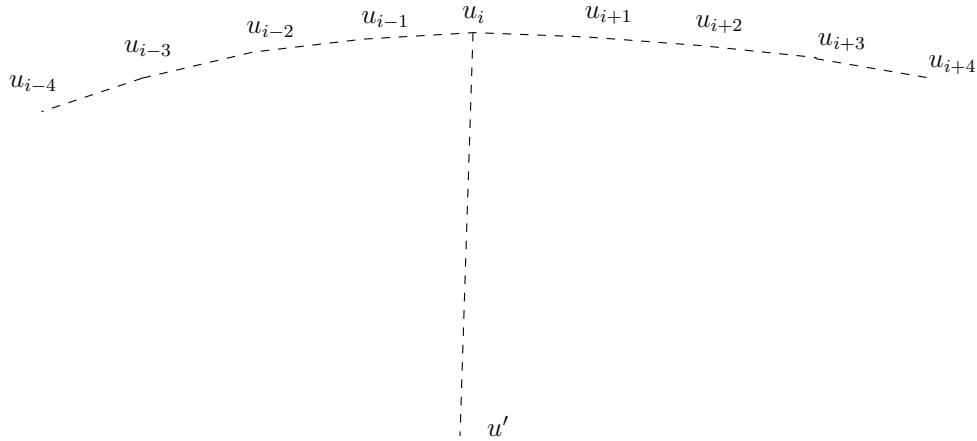
► **Observation.** *The following hold: (i) Unit charges associated to middle edges are distributed to interior vertices in amounts of  $1/2$  or  $1$ . (ii) The unit charge on the middle edge incident to  $u_i$  is distributed to one or two interior vertices at distance at most 2 in  $G_\delta$ ; i.e., this process can only affect vertices in  $\Gamma(u_i) \cup \Gamma(\Gamma(u_i))$ .*

The following lemma specifies the range affected by one charge distribution.

► **Lemma 3.** *Let  $u_i \in D \cap F$  be a vertex of degree 3 that sends charge to some interior vertex  $v \in \Gamma(u_i) \cup \Gamma(\Gamma(u_i))$ , where  $v$  is not necessarily unique. Then  $v$  can only receive charges from elements of  $\{u_{i-3}, u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, u_{i+3}\}$ .*

**Proof.** Write  $u = u_i$  (for short). Consider the coordinate system  $\Sigma_u$ , and the rectangle  $R_u = [x(u) - 7/4, x(u) + 7/4] \times [y(u) - 2, y(u)]$  in this system; refer to Fig 9. By the charging scheme,  $u$  can only send charges to interior vertices contained in  $R_u$ . Consider the larger rectangle  $R'_u = [x(u) - 15/4, x(u) + 15/4] \times [y(u) - 4, y(u)] \supset R_u$ . Since  $v$  can only receive charges from vertices at distance at most 2 from it, any element sending charges to  $v$  would be contained in  $R'_u$ .

Since  $u \in D \cap F$ ,  $u$  is an endpoint of a diameter pair, say,  $uu'$ , where  $u' \in D$ . Observe that the ray  $uu'$  makes an angle of at most  $1^\circ$  with  $\vec{r}_u$ , the vertical ray from  $u$  pointing downwards. Indeed, otherwise one of the two distances  $|u_{i-1}u'|$  and  $|u_{i+1}u'|$  would be larger



■ **Figure 9** The flat neighborhood of  $u = u_i$  and a diameter pair  $(u, u')$ . All points lie in the closed halfplane below the horizontal line incident to  $u_i$ .

than  $|uu'| = \Delta$ , as the longest side in an obtuse triangle. Recall that  $\Delta = \Omega(n)$  by Lemma 2; we may assume that  $n$  is large enough, e.g.,  $n \geq 100$ , so that  $\Delta \geq 10$ . By convexity,  $\text{conv}(u_{i-3}, u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, u_{i+3}, u')$  is empty of points from  $H$  in its interior. Recall that  $u$  has a flat neighborhood, and intuitively, this implies that  $v$  cannot receive charges from the 'other side' of the boundary. More precisely, since  $u$  has a flat neighborhood, the rectangle  $R'_u$  does not contain any elements of  $H \setminus \{u_{i-3}, u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, u_{i+3}\}$ , and so  $v$  cannot receive charges from elements in  $H \setminus \{u_{i-3}, u_{i-2}, u_{i-1}, u_i, u_{i+1}, u_{i+2}, u_{i+3}\}$ , as required. ◀

We next formulate and prove the main property accomplished by the charge distribution.

► **Lemma 4.** *The final charge for any interior vertex is at most 6.*

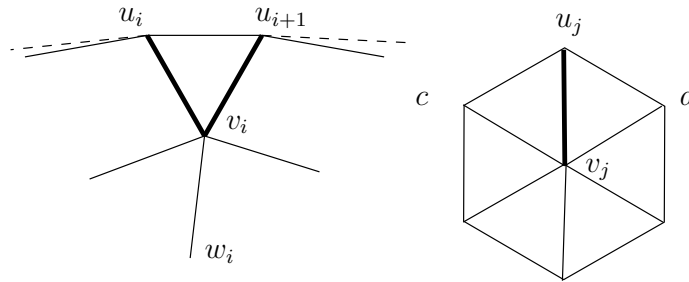
**Proof.** By Lemma 3 it suffices to bound from above the charge received by an interior vertex from the left and from the right. Specifically, we show that the maximum final charge for any such vertex is at most 6.

**Overcharging by Case 1 only:** Let  $v$  be an interior vertex of degree 5 that is charged as  $b$  in Fig. 4 from the left (i.e., from an edge  $u_i v_i$  on the left), and as  $a$  in Fig. 4 from the right (i.e., from an edge  $u_i v_i$  on the right). The charges received sum up to at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.

**Overcharging by Case 1 and Case 2:** The argument is similar to that for the previous case. Let  $v$  be an interior vertex that is charged from the left as  $b$  or  $d$  in Fig. 5 (right) and is charged from the right as  $a$  in Fig. 4. The charges received sum up to at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.

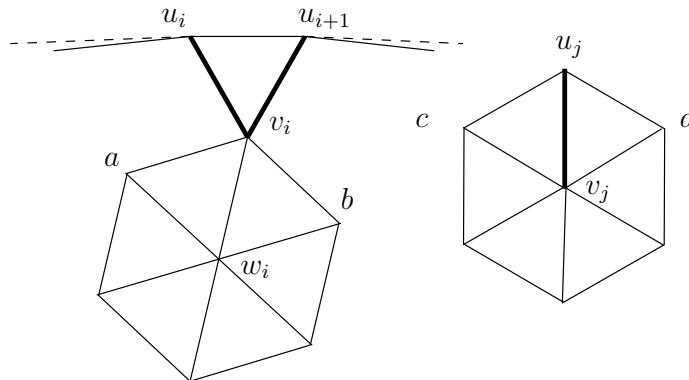
**Overcharging by Case 1 and Case 3:** Assume that an interior vertex receives a unit charge as vertex  $v_i$  in Fig. 6 according to Case 3. This happens only when  $\text{deg}(v_i) \leq 4$ ; and then it is easy to see that no overcharging can occur even if  $v_i$  receives  $1/2$  unit from the left and from the right (according to Case 1). In the remaining case,  $\text{deg}(v_i) = 5$ , both vertices that get charged according to Case 3, only receive  $1/2$  unit charge each. Since charges received from Case 1 are limited to  $1/2$  unit, the charges received by  $v_i$  sum up to at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.

**Overcharging by Case 1 and Case 4:** Assume that some interior vertex receives a unit charge from the left according to Case 4 and  $1/2$  unit charge from the right according to Case 1. See Fig. 10 and Fig. 11, and observe that  $j = i + 2$ . First, consider the situation in Fig. 10, when  $\deg(w_i) = 5$ . If the overcharged vertex is  $v_i = c$ , the interior vertex  $c$  would be adjacent to three extreme vertices  $(u_i, u_{i+1}, u_{i+2})$ , a contradiction. If the overcharged vertex is  $w_i = c$ , the distance from  $w_i$  to  $\ell(u_i u_{i+1})$  is at least  $2\frac{\sqrt{3}}{2} = \sqrt{3}$ ; on the other hand, the distance from  $c$  to the same line is less than 1, since  $u_{i+1}$  has a flat neighborhood. Therefore such an overcharging cannot occur.



■ **Figure 10** Overcharging by Case 4 (left) and Case 1 (right);  $\deg(w_i) = 5$ .

Second, consider the situation in Fig. 11, when  $\deg(w_i) = 6$ . If the overcharged vertex is  $b = c$ , the charges received sum up to at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.



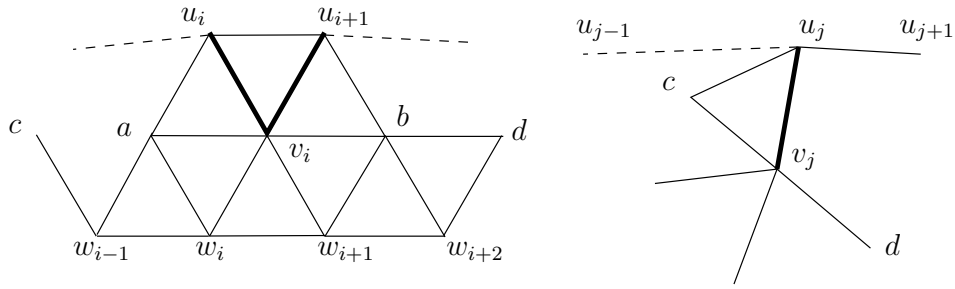
■ **Figure 11** Overcharging by Case 4 (left) and Case 1 (right);  $\deg(w_i) = 6$ .

**Overcharging by Case 2 only:** Each charge received by an interior vertex in Case 2 is equal to  $1/2$ ; as such, an interior vertex of degree at most 5 can receive at most  $1/2$  units from the left and at most  $1/2$  units from the right, as required.

**Overcharging by Case 2 and Case 3:** Refer to Fig. 12. Assume for contradiction that a vertex of degree at most 5 receives a  $1/2$  unit charge from the left according to Case 2 (as vertex  $b$ ,  $w_{i+1}$ ,  $w_{i+2}$ , or  $d$ ), and a  $1/2$  unit charge as vertex  $c$  or  $v_j$  according to Case 3. It is clear that  $j \geq i + 2$ . The charge received is at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required. The situation when  $\deg(v_j) \leq 4$  is similarly easy to analyze.

**Overcharging by Case 2 and Case 4:** Assume for contradiction that an interior vertex has degree 5 and receives a  $1/2$  unit charge from the left according to Case 2, and a one unit charge from the right according to Case 4; see Fig. 13. Then

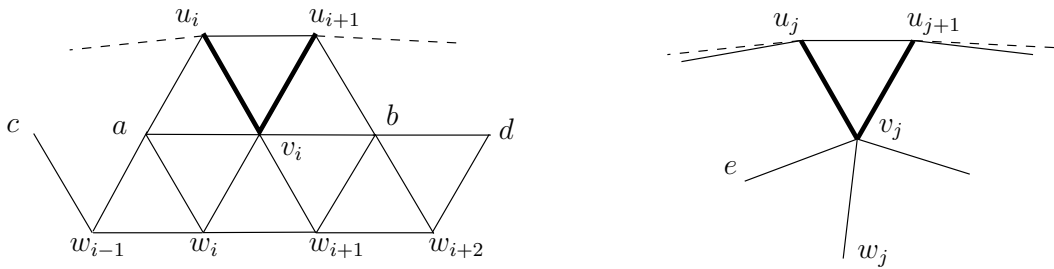
30:10 A Product Inequality for Extreme Distances



■ **Figure 12** Overcharging by Case 2 (left) and Case 3 (right).

- vertex  $w_{i+2}$  on the left must coincide with vertex  $w_j$  on the right.
- vertex  $b$  on the left must coincide with vertex  $e$  on the right.
- vertex  $d$  on the left must coincide with vertex  $v_j$  on the right.

However this is impossible to achieve with  $u_i, u_{i+1}, u_j, u_{j+1}$  consecutive extreme vertices with flat neighborhoods, since  $u_j, u_{j+1}$  would need to lie strictly below  $\ell(u_i, u_{i+1})$ , while the triangle  $\Delta v_j u_j u_{j+1}$  is equilateral and  $\ell(u_j, u_{j+1})$  is almost horizontal (recall from Case 2 that  $\deg(d) \leq 4$  due to some restrictions imposed on its neighbors). The above conditions imply that  $v_j$  is lower than  $d$  and so the two points cannot coincide.



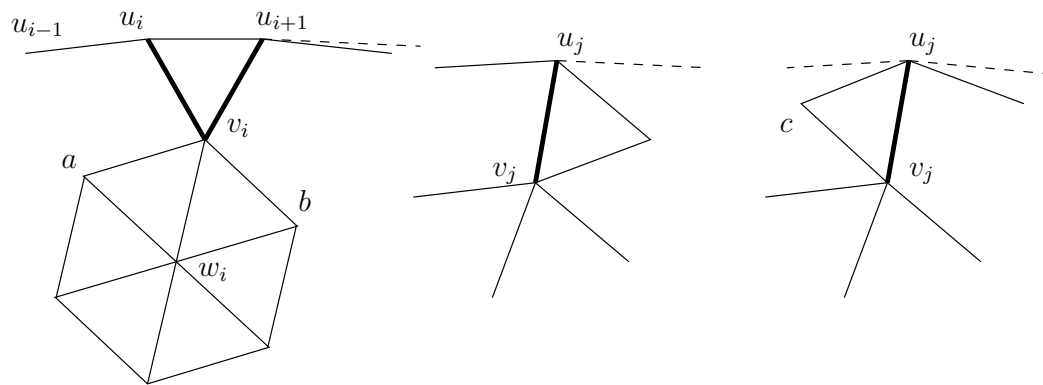
■ **Figure 13** Overcharging by Case 2 (left) and Case 4 (right).

**Overcharging by Case 3 only:** Since  $v_i$  is adjacent to exactly one extreme vertex,  $v_i$  cannot receive multiple charges. Any other vertex can only receive a  $1/2$  unit charge from the left and a  $1/2$  unit charge from the right. As such, the charge received is bounded from above by  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.

**Overcharging by Case 3 and Case 4:** Observe that vertex  $v_i$  in Case 4, see Fig. 7 (left or right), is adjacent to exactly two extreme vertices; consequently it cannot receive any charge according to the procedure in Case 3. Similarly, observe that vertex  $w_i$  in Case 4, see Fig. 7 (left), is not adjacent to any extreme vertex; consequently it cannot receive any charge according to the procedure in Case 3.

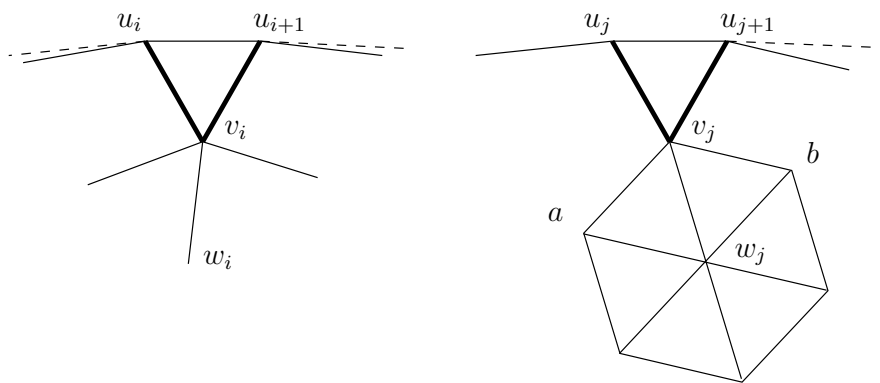
Consider now the scenario illustrated in Fig. 14 in which  $\deg(v_i) = \deg(v_j) = 5$ ; observe that  $j = i + 2$ . If  $b = v_j$  receives a  $1/2$  unit charge according to Case 4 and another  $1/2$  unit charge according to Case 3, the charge received is bounded from above by  $\frac{1}{2} + \frac{1}{2} = 1$ , as required; see Fig. 14 (middle). Similarly, if  $b = c$  receives a  $1/2$  unit charge according to Case 4 and another  $1/2$  unit charge according to Case 3, the charge received is at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required; see Fig. 14 (right). Therefore such an overcharging cannot occur.

**Overcharging by Case 4 only:** Refer to Fig. 15 and Fig. 16; observe that  $j = i + 2$ . One possibility is having  $w_i = a$ , with  $w_i$  receiving a unit charge according to Case 4 (on the left) and  $a$  receiving a  $1/2$  unit charge according to the same case (on the right); see



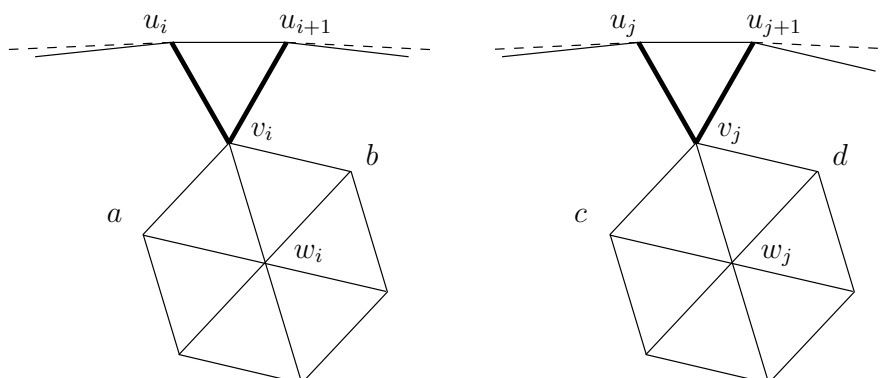
■ **Figure 14** Overcharging by Case 4 (left) and Case 3 (middle and right).

Fig. 15. However this requires  $v_i v_j$  to be yet another edge in  $G_\delta$  beyond the five incident to  $v_j$ , contradicting the assumption of Case 4 that  $\deg(v_j) = 5$ .



■ **Figure 15** Overcharging by Case 4 only.

Another possibility is having  $b = c$ , with  $b$  receiving a  $1/2$  unit charge according to Case 4 (on the left) and  $c$  receiving a  $1/2$  unit charge according to the same case (on the right); see Fig. 16. The charge received is at most  $\frac{1}{2} + \frac{1}{2} = 1$ , as required.



■ **Figure 16** Overcharging by Case 4 only.

With all possibilities of potential overcharging having been analyzed, the proof of Lemma 4 is now complete. ◀



## 5 Conclusion of the proof

In this section we finalize the proof of Theorem 1.

► **Lemma 5.**  $s_{\min} \leq 3n - 2d + O(1)$ .

**Proof.** Assume that each element of  $I$  carries an initial charge equal to its degree in  $G_\delta$  (at most 6). Note that each element of  $H$  has degree at most 3; indeed, if  $\deg(u) = 4$ , then the interior angle at  $u$  equals  $180^\circ$ , and so  $u$  is not an extreme vertex of  $\text{conv}(S)$ . In particular, each element of  $D \cap F$  has degree at most 3.

Each vertex of  $D \cap F$  of degree 3 in  $G_\delta$  sends (distributes) a unit charge to one or two interior vertices of degree at most 5; so that the final charge of each interior vertex is at most 6; with each vertex receiving a charge at most 2 (and the final charge of each element of  $D \cap F$  is 2); all these are consequences of Lemma 4. A key observation is that  $|F \cap D| \geq |D| - O(1)$ , since there are only  $O(1)$  elements of  $D$  that do not have flat neighborhoods. Assuming the charging procedure finalized, we have

$$\begin{aligned} 2s_{\min} &= \sum_{p \in S} \deg(p) \leq 3|H \setminus F \cap D| + 2|F \cap D| + 6|S \setminus H| \\ &= 3h - 3|F \cap D| + 2|F \cap D| + 6n - 6h \\ &= 6n - 3h - |F \cap D| \leq 6n - 3d - d + O(1) \\ &= 6n - 4d + O(1), \end{aligned}$$

as required. ◀

**Proof of Theorem 1.** Using the inequalities on  $s_{\min}$  and  $s_{\max}$  stated in Lemma 5 and Equation (1), respectively, we obtain

$$s_{\min}s_{\max} \leq (3n - 2d + O(1))d \leq \frac{9}{8}n^2 + O(n),$$

as required. Indeed, setting  $x = d/n$  yields the quadratic function  $f(x) = x(3 - 2x)$ , which attains its maximum value  $\frac{9}{8}$  for  $x = \frac{3}{4}$ . Thus  $(3n - 2d)d \leq \frac{9}{8}n^2$  and we also have  $O(1)d = O(d) = O(n)$ ; adding these two inequalities yields the one claimed above. This concludes the proof of Theorem 1.

---

### References

- 1 Paul Erdős and János Pach. Variation on the theme of repeated distances. *Combinatorica*, 10(3):261–269, 1990. doi:10.1007/BF02122780.
- 2 Heiko Harborth. Solution to problem 664A. *Elem. Math*, 29:14–15, 1974.
- 3 János Pach and Pankaj K. Agarwal. *Combinatorial Geometry*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1995.
- 4 Paul R. Scott and Poh Wah Awyong. Inequalities for convex sets. *Journal of Inequalities in Pure and Applied Mathematics*, 1(1):6, 2000.

# Topological Data Analysis in Information Space

**Herbert Edelsbrunner**

IST Austria (Institute of Science and Technology Austria)  
Am Campus 1, 3400 Klosterneuburg, Austria  
edels@ist.ac.at

**Žiga Virk**

Faculty of Computer and Information Science, University of Ljubljana  
Vecna pot 113, 1000 Ljubljana, Slovenia  
ziga.virk@fri.uni-lj.si

**Hubert Wagner**

IST Austria (Institute of Science and Technology Austria)  
Am Campus 1, 3400 Klosterneuburg, Austria  
hwagner@ist.ac.at

---

## Abstract

Various kinds of data are routinely represented as discrete probability distributions. Examples include text documents summarized by histograms of word occurrences and images represented as histograms of oriented gradients. Viewing a discrete probability distribution as a point in the standard simplex of the appropriate dimension, we can understand collections of such objects in geometric and topological terms. Importantly, instead of using the standard Euclidean distance, we look into dissimilarity measures with information-theoretic justification, and we develop the theory needed for applying topological data analysis in this setting. In doing so, we emphasize constructions that enable the usage of existing computational topology software in this context.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Computational topology, persistent homology, information theory, entropy

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.31

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1903.08510>.

**Funding** This research is partially supported by the Office of Naval Research, through grant no. N62909-18-1-2038, and the DFG Collaborative Research Center TRR 109, “Discretization in Geometry and Dynamics”, through grant no. I02979-N35 of the Austrian Science Fund (FWF).

## 1 Introduction

The field of *computational topology* has a very short history, at least if compared with standard subfields of mathematics [15]. Starting as a broadening of the geometric tool-set to answer low-dimensional topology questions, it soon extended its scope to include topological problems in high-dimensional data analysis, giving rise to the subfield of *topological data analysis* (TDA) [11]. At the very foundation of topology is the notion of neighborhood, which in practice is often constructed from the Euclidean metric in real space. One reason for its popularity is the extensive tool-set associated with this metric, but there is evidence that this choice is often suboptimal [22]. Other popular options are the Hamming distance and discrete distances, which are preferred for their simplicity.

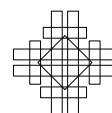
While the choice of distance is important, there is a paucity in our understanding of the consequences. We contribute to the study by connecting distances with information-theoretic foundations [14] to the tool-set of topological data analysis. These distances include the *Fisher information metric* [2], and the *relative entropy*, also known as *Kullback–Leibler divergence* [23]. The former is obtained by integrating the square root of the relative entropy



© Herbert Edelsbrunner, Žiga Virk, and Hubert Wagner;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 31; pp. 31:1–31:14  
Leibniz International Proceedings in Informatics



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



along shortest paths. Restricting the points to the standard simplex, we get a metric between discrete probability distributions, which has information-theoretic significance. While the latter is not a distance in the strict sense, it plays an important role in many application areas as preferred measurement between probability distributions. The Kullback–Leibler divergence is also an example of the broader class of Bregman divergences [10], which have been studied from a geometric point of view in [8, 27]. To clearly distinguish the spaces equipped with dissimilarity measures based on information-theoretic notions from the standard Euclidean setting, we call them *information spaces*.

Our interest in information-theoretic distances complements the common theme of incorporating statistical tools into the topological data analysis pipeline. Like-minded efforts advocate that results obtained with topological methods need calibration through statistics, and that new developments in both fields will be needed to achieve this [1]. The goal of this paper is more modest: the import of concepts from information theory into the inner workings of the topological tools. This can be compared with the *persistent entropy* introduced by Rucco and collaborators [25] to sharpen the sensitivity of persistent homology in the assessment of network organization. Unrelated to this thought but important to our effort is the work of Antonelli and collaborators [4] on the isometry between Fisher information space and Euclidean space, which we recast in Section 3. We follow up with a list of concrete contributions the reader finds in this paper:

- In Section 2, we prove that relative entropy balls are convex. In contrast, the balls based on the Burg entropy, which is popular in speech recognition, are convex only for small radii, as elaborated in the full version of this paper.
- In Section 3, we give an explicit description of the Fisher information metric derived from the relative entropy and of Antonelli’s isometry, prove that Fisher information balls are convex, and generalize the Antonelli isometry to decomposable Bregman divergences.
- In Section 4, we prove tight bounds relating the Jensen–Shannon divergence with the Fisher information metric, recalling that the square root of the former is a metric [19].

In a final technical section, we tie things together by explaining how these notions of distance are used to compute the persistence diagrams of data in this space. We compare the results for the three notions of distance studied in this paper: the *relative entropy*, the *Fisher information metric*, and the *Jensen–Shannon divergence*. In the full version of this paper, we expand these consideration to the case of Burg entropy.

**Scope of the paper.** Apart from reporting the above technical results, our aim is to provide a self-contained reference for researchers interested in the theoretical and applied side of information theory in the context of computational topology. To this end, we provide intuitive explanations of basic topological objects measured with information-theoretic distances.

**Overview.** To put our results in perspective, we mention that this paper is third in a series. The entry point is [18], where Bregman divergences are shown to be compatible with basic topological tools. In subsequent work [20], we studied properties of Bregman balls, in particular bounding the location of first intersection of such balls. Our previous work was concerned with shared properties of Bregman divergences; in this work, we focus on information-theoretic distances based on Shannon’s entropy.

While working directly with the Kullback–Leibler divergence is ideal from an information-theoretic standpoint, its lack of symmetry and the triangle inequality prohibits the use with existing computational geometry and topology tools. This prompts the study of metrics derived from Bregman divergences in general, and from Kullback–Leibler divergence

specifically. The latter include the Fisher metric and the Jensen-Shannon metric. Studying properties of these metrics is the main focus of this paper, which is structured as follows: Section 2 presents the background on Shannon entropy and relative entropy. Section 3 relates the Fisher information metric to the relative entropy and to the Euclidean metric. Section 4 proves inequalities relating the Jensen–Shannon divergence with the Fisher information metric. Section 5 uses persistent homology to compare these notions of distance if applied to data. Section 6 concludes this paper.

## 2 Entropy and Relative Entropy

Starting with the Shannon entropy, we define the relative entropy of an ordered pair of points, and provide an intuitive explanation of this measurement. Considering the ball consisting of all points for which the relative entropy from the center does not exceed a given threshold, we prove that it is convex, in the original coordinate system and in all dimensions.

**Convex function and divergence.** We write  $\mathbb{R}_+$  for the set of positive real numbers,  $\mathbb{R}_+^n$  for the positive orthant in  $n$  dimensions, and  $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}_+^n$  for a point in the orthant. Its components satisfy  $x_i > 0$  for  $1 \leq i \leq n$ . The (negative, extended) *Shannon entropy* is the function  $E: \mathbb{R}_+^n \rightarrow \mathbb{R}$  defined by mapping  $x$  to

$$E(x) = \sum_{i=1}^n [x_i \ln x_i - x_i]. \quad (1)$$

It is strictly convex and infinitely often differentiable; see Figure 1 for the graph of the function in one dimension. The *relative entropy* from  $x$  to  $y$  is the difference between  $E$  and the best linear approximation of  $E$  at  $y$ , both evaluated at  $x$ :

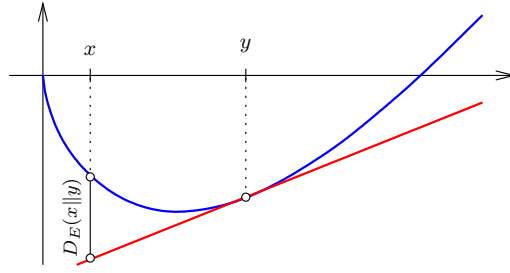
$$D_E(x||y) = E(x) - [E(y) + \langle \nabla E(y), x - y \rangle] \quad (2)$$

$$= \sum_{i=1}^n [x_i \ln \frac{x_i}{y_i} - x_i + y_i]. \quad (3)$$

The relative entropy is also known as the *Kullback–Leibler divergence*; see [3, page 57]. The construction of the relative entropy is illustrated in Figure 1, making it obvious that  $D_E(x||y)$  is neither symmetric in its two arguments, nor does it satisfy the triangle inequality. Notwithstanding, the primary interpretation of the relative entropy is as a measure of distance. Intuitively, the relative entropy from a distribution  $x$  to another distribution  $y$  measures the average *surprisal* when we expect items according to  $y$  but observe items according to  $x$ . More technically, this surprisal is expressed as the loss of coding efficiency, namely the expected number of extra bits needed to encode the messages from  $x$  using a code optimized for  $y$ . In practice, it quantifies how well  $y$  approximates  $x$ , and later we use this interpretation to understand the meaning of relative-entropy balls and their intersections.

► **Remark 1.** The definition of the relative entropy is an example of a more general construction. Letting  $\Omega \subseteq \mathbb{R}^n$  be open and convex, a function  $F: \Omega \rightarrow \mathbb{R}$  is of *Legendre type* if it is strictly convex, differentiable, and the gradient diverges when we approach the boundary of  $\Omega$ . Clearly, the Shannon entropy is of Legendre type. Named after Lev Bregman [10], the corresponding *Bregman divergence* is defined by substituting  $F$  for  $E$  in (2).

► **Remark 2.** If we restrict a Legendre type function to an affine subspace, we get another Legendre type function and therefore another divergence. An important example is the *standard  $(n-1)$ -simplex*,  $\Delta = \Delta^{n-1}$ , which is the intersection of  $\mathbb{R}_+^n$  with the  $(n-1)$ -plane defined by  $\sum_{i=1}^n x_i = 1$ . Since  $\mathbb{R}_+^n$  is open so is  $\Delta$ . Every point of  $\Delta$  can be interpreted as a discrete probability distribution for  $n$  elements. The Shannon entropy and the relative entropy are defined as in (1) and (3), except for a smaller domain.



■ **Figure 1** The graph of the Shannon entropy, the graph of its best linear approximation at  $y$ , and the relative entropy from  $x$  to  $y$ .

**Decomposability and convexity.** We note that a function  $F: \Omega \rightarrow \mathbb{R}$  whose restrictions obtained by fixing coordinates are convex is not necessarily convex. An example is the function  $F(x) = \prod_{i=1}^n x_i$ . On the other hand, if  $F$  is *decomposable*:  $F(x) = \sum_{i=1}^n F_i(x_i)$ , then the convexity of the components implies the convexity of  $F$ .

► **Lemma 3** (Composable Convexity). *Let  $\Omega \subseteq \mathbb{R}^n$  be convex and  $F: \Omega \rightarrow \mathbb{R}$  a decomposable function. The components of  $F$  are convex if and only if  $F$  is convex.*

**Proof.** It is clear that the convexity of  $F$  implies the convexity of its components. We thus limit ourselves to proving the other implication. Let  $x, y \in \Omega$ , let  $0 \leq \lambda \leq 1$ , and let  $u = (1 - \lambda)x + \lambda y$  be the corresponding convex combination. Since the components of  $F$  are convex, by assumption, we have  $F_i(u_i) \leq (1 - \lambda)F_i(x_i) + \lambda F_i(y_i)$  for every  $1 \leq i \leq n$ . Taking sums on the left and on the right, we get  $F(u) \leq (1 - \lambda)F(x) + \lambda F(y)$ ; that is:  $F$  is convex as claimed. ◀

► **Remark 4.** While nonconvex components imply a nonconvex function, they do not imply nonconvex sublevel sets. This stronger implication is valid for *strongly decomposable* functions  $F$ , by which we mean that all components are the same:  $F_i = F_j$  for all  $1 \leq i, j \leq n$ . To see this, let the component  $f$  of  $F$  be nonconvex, assume  $f$  is nonnegative, and let  $a < b$  such that  $f(\frac{a+b}{2}) > \frac{1}{2}f(a) + \frac{1}{2}f(b)$ . Supposing 1 is in the domain of  $f$ , we set  $x = (a, b, 1, \dots, 1)$  and  $y = (b, a, 1, \dots, 1)$  and note that  $\frac{x+y}{2} = (\frac{a+b}{2}, \frac{a+b}{2}, 1, \dots, 1)$ . Hence,

$$F(\frac{x+y}{2}) = 2f(\frac{a+b}{2}) + (n-2)f(1) > f(a) + f(b) + (n-2)f(1) = \frac{1}{2}[F(x) + F(y)]. \quad (4)$$

Setting  $r^2 = F(x) = F(y)$ , we see that  $F^{-1}[0, r^2]$  is a nonconvex subset of  $\mathbb{R}^n$ .

**Convexity of entropy balls.** To study the local behavior of a divergence, we fix a point  $x \in \Omega$  and call  $D_x: \Omega \rightarrow \mathbb{R}$  defined by  $D_x(y) = D_F(x||y)$  the *divergence function* of  $x$ . Its sublevel sets are balls of points whose divergence from  $x$  is bounded by a non-negative threshold:

$$B_r(x) = D_x^{-1}[0, r^2] = \{y \in \Omega \mid D_F(x||y) \leq r^2\}. \quad (5)$$

Assuming  $F$  is the Shannon entropy, we call  $B_r(x)$  an *entropy ball*. Observe that such a ball is the set of all probability distributions that approximate the center distribution,  $x$ , with a loss of at most  $r^2$  bits. We recall that in practice each distribution characterizes an object, perhaps an image or a text document. In this case, it is clear that a ball represents a collection of similar objects, for a chosen similarity threshold  $r^2$ .

For general Legendre type functions, the balls are not necessarily convex, and we will encounter nonconvex balls shortly. However, entropy balls are necessarily convex.

► **Theorem 5** (Convexity of Entropy Balls). *Let  $E: \mathbb{R}_+^n \rightarrow \mathbb{R}$  be the Shannon entropy. Then the entropy ball  $B_r(x)$  is convex for every  $x \in \mathbb{R}_+^n$  and every  $r^2 \geq 0$ .*

**Proof.** Since the Shannon entropy is strongly decomposable, so is the divergence function of every point  $x \in \mathbb{R}_+^n$ :  $D_x(y) = E(x) - E(y) - \langle \nabla E(y), x - y \rangle = \sum_{i=1}^n [x_i \ln x_i - x_i \ln y_i - x_i + y_i]$ , in which the  $y_i$  are variables and the  $x_i$  are constants. Each component is convex because  $-\ln t$  is convex, By Lemma 3, this implies that  $D_x$  is convex, so all its sublevel sets are convex. ◀

### 3 Fisher Information Metric

Like any other twice differentiable Legendre type function, the Shannon entropy induces a metric that integrates infinitesimal steps along shortest paths. This metric has an isometry to Euclidean space, which facilitates topological methods applied to data in information space.

**From divergence to distance.** Recall that the Shannon entropy is twice differentiable, so the *Hessian matrix* of second derivatives at any point  $y \in \mathbb{R}_+^n$  is well defined:

$$H_E(y) = \left[ \frac{\partial^2 E}{\partial x_i \partial x_j}(y) \right]_{1 \leq i, j \leq n}. \tag{6}$$

This symmetric square matrix defines a scalar product at  $y$  by mapping two vectors  $u, v \in \mathbb{R}^n$  to  $\langle u, v \rangle_y = \frac{1}{2} u^T H_E(y) v$ . The corresponding Riemannian metric is called the *Fisher information metric*:  $d_E: \mathbb{R}_+^n \times \mathbb{R}_+^n \rightarrow \mathbb{R}$ . Given points  $x, y \in \mathbb{R}_+^n$ , the distance is the length of the shortest path  $\gamma: [0, 1] \rightarrow \mathbb{R}_+^n$  with  $\gamma(0) = x$  and  $\gamma(1) = y$ , in which the length is

$$\text{Length}(\gamma) = \int_{t=0}^1 \sqrt{\langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}} dt = \int_{t=0}^1 \sqrt{\frac{1}{2} \dot{\gamma}(t)^T H_E(\gamma(t)) \dot{\gamma}(t)} dt. \tag{7}$$

Since the Shannon entropy is decomposable, the only non-zero entries in the Hessian matrix are the second derivatives of the component functions in the diagonal:

$$H_E(y) = \begin{bmatrix} 1/y_1 & 0 & \dots & 0 \\ 0 & 1/y_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/y_n \end{bmatrix}. \tag{8}$$

Accordingly, the formula for the length of a smooth path simplifies to

$$\text{Length}(\gamma) = \int_{t=0}^1 \sqrt{\frac{1}{2} \sum_{i=0}^n \frac{\dot{\gamma}_i(t)^2}{\gamma_i(t)}} dt, \tag{9}$$

in which  $\gamma_i(t)$  and  $\dot{\gamma}_i(t)$  are the  $i$ -th components of the curve and its velocity vector at  $t$ .

► **Remark 6.** The construction of the metric can be restricted to any smooth submanifold of the domain. Since this limits the set of available paths, the distance between two points cannot be less than without the restriction. For example, we may restrict the Fisher information metric in  $\mathbb{R}_+^n$  to the standard simplex:  $d_{E|\Delta}: \Delta \times \Delta \rightarrow \mathbb{R}$ . While the relative entropy in  $\Delta$  is the restriction of the relative entropy in  $\mathbb{R}_+^n$ , this is not the case for the Fisher information metric.

**Euclidean isometries.** As discovered by Antonelli [4], there is a simple diffeomorphism that maps  $\mathbb{R}_+^n$  equipped with the Fisher information metric to  $\mathbb{R}_+^n$  equipped with the Euclidean metric. Such an isometry exists for every length metric defined as explained by a twice differentiable decomposable Legendre type function. We explain its construction in this more general setting. Suppose  $\varphi: \Omega \rightarrow \Omega_2$  is a diffeomorphism. Given a metric  $d_2: \Omega_2 \times \Omega_2 \rightarrow \mathbb{R}$ , we get an *induced metric*,  $d_\varphi: \Omega \times \Omega \rightarrow \mathbb{R}$ , defined by  $d_\varphi(x, y) = d_2(\varphi(x), \varphi(y))$ . By construction,  $\varphi$  is an isometry between the two metric spaces. We are interested in the case in which  $d_2$  is the Euclidean metric.

► **Lemma 7 (Euclidean Isometry).** *Let  $\Omega$  be an open convex subset of  $\mathbb{R}^n$  and  $F: \Omega \rightarrow \mathbb{R}$  a twice differentiable decomposable Legendre type function. Then there exists a decomposable isometry  $\varphi: \Omega \rightarrow \Omega_2 \subseteq \mathbb{R}^n$  connecting the length metric defined by  $F$  in  $\Omega$  with the Euclidean metric in  $\Omega_2$ .*

**Proof.** We construct a decomposable diffeomorphism  $\varphi: \Omega \rightarrow \Omega_2$  such that the length metric  $d_F$  is induced by  $\varphi$  and the Euclidean metric on  $\Omega_2$ . Write  $\varphi_i(x_i)$  for the components of the diffeomorphism and  $\sum_{i=1}^n (du_i)^2$  for the differential form of the Euclidean metric in  $\Omega_2$ . Setting  $u_i = \varphi_i(x_i)$  so that  $du_i = \varphi'_i(x_i) dx_i$ , we get

$$\sum_{i=1}^n (du_i)^2 = \sum_{i=1}^n \varphi'_i(x_i)^2 (dx_i)^2 = \sum_{i=1}^n \frac{1}{2} F''_i(x_i) (dx_i)^2, \quad (10)$$

in which the last sum is the differential form of the metric defined by  $F$ , and the second equality is required to get an isometry. This condition simplifies to

$$\varphi'_i(x_i) = \sqrt{\frac{1}{2} F''_i(x_i)}, \quad (11)$$

for  $1 \leq i \leq n$ . Since  $F$  is strictly convex, we have  $F''_i(x_i) > 0$  for every component function, so this equation has a solution. ◀

**From Fisher information to Euclidean distance.** We make use of Lemma 7 by constructing the isometry that relates the Fisher information metric in  $\mathbb{R}_+^n$  and in  $\Delta = \Delta^{n-1}$  with the Euclidean metric in  $\mathbb{R}_+^n$  and in  $\sqrt{2}\mathbb{S}_+^{n-1}$ , the latter being our notation for the positive orthant of the  $(n-1)$ -sphere with radius  $\sqrt{2}$  centered at the origin in  $\mathbb{R}^n$ . As mentioned before, this isometry is well known [2, 4], and we give the construction for completeness.

► **Theorem 8 (Fisher Information to Euclidean Distance).** *Let  $E: \mathbb{R}_+^n \rightarrow \mathbb{R}$  be the Shannon entropy, and  $x, y \in \mathbb{R}_+^n$ . Then*

$$d_E(x, y) = \sqrt{2 \sum_{i=1}^n (\sqrt{x_i} - \sqrt{y_i})^2}, \quad (12)$$

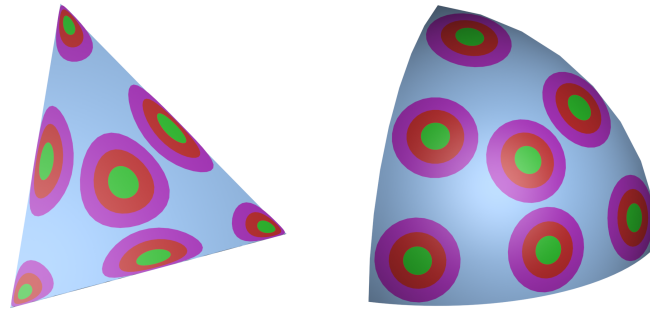
$$d_{E|\Delta}(x, y) = \sqrt{2} \arccos \sum_{i=1}^n \sqrt{x_i} \sqrt{y_i}, \quad (13)$$

where we assume  $x, y \in \Delta \subseteq \mathbb{R}_+^n$  in (13). Furthermore, the balls under  $d_E$  and  $d_{E|\Delta}$  are convex.

**Proof.** We prove both equations with the diffeomorphism  $\varphi: \mathbb{R}_+^n \rightarrow \mathbb{R}_+^n$  that distorts the Fisher information metric to the Euclidean metric. Since  $E$  decomposes into identical components, so does  $\varphi$ . By Equation (11) in the proof of Lemma 7, the derivative of this component satisfies  $p'(t) = 1/\sqrt{2t}$ . We get  $p(t) = \sqrt{2t}$  by integration. As illustrated in Figure 2, the diffeomorphism maps a point  $x$  with coordinates  $x_i > 0$  to the point  $\varphi(x)$  with coordinates  $p(x_i) = \sqrt{2x_i} > 0$ . The squared Euclidean distance between  $\varphi(x)$  and  $\varphi(y)$  is  $\sum_{i=1}^n (\sqrt{2x_i} - \sqrt{2y_i})^2$ , which implies (12).



The image of the standard simplex,  $\varphi(\Delta)$ , consists of all points with coordinates  $u_i = p(x_i) > 0$  that satisfy  $u_1^2 + u_2^2 + \dots + u_n^2 = 2$ . This is the equation of the sphere with radius  $\sqrt{2}$  centered at the origin in  $\mathbb{R}^n$ . Denoting the positive portion of this sphere by  $\sqrt{2}\mathbb{S}_+^{n-1}$ , we get an isometry  $\varphi|_\Delta: \Delta \rightarrow \sqrt{2}\mathbb{S}_+^{n-1}$ . Given points  $u, v \in \sqrt{2}\mathbb{S}_+^{n-1}$ , the shortest spherical path connecting them is a portion of the great-circle that passes through  $u$  and  $v$ . Its length is  $\sqrt{2}$  times the angle between the vectors  $u, v \in \mathbb{R}_+^n$ . This implies (13).



■ **Figure 2** *Left:* three Fisher information disks each for seven points inside the standard triangle. *Right:* the images of the disks in the positive portion of the sphere. For aesthetic reasons, we scale the sphere by a factor  $1/\sqrt{2}$  relative to the triangle.

To prove convexity, let  $x \in \mathbb{R}_+^n$ , consider the function  $d_x: \mathbb{R}_+^n \rightarrow \mathbb{R}$  defined by  $d_x(y) = d_E^2(x, y)$ , and note that  $d_x^{-1}[0, r^2]$  is the ball with center  $x$  and radius  $r \geq 0$  under the Fisher information metric. It is the preimage of the function  $f_u: \mathbb{R}_+^n \rightarrow \mathbb{R}$  defined by  $f_u(v) = \sum_{i=1}^n (v_i - u_i)^2$ , with  $u_i = \sqrt{2x_i}$  and  $v_i = \sqrt{2y_i}$ , for  $1 \leq i \leq n$ . Hence,  $d_x$  is decomposable with  $d_x(y) = 2 \sum_{i=1}^n (\sqrt{y_i} - \sqrt{x_i})^2$ . The components are convex because  $-\sqrt{t}$  is convex. By Lemma 3,  $d_x$  is convex, and because  $x$  can be any point in  $\mathbb{R}_+^n$ , all balls under  $d_E$  are convex. To finally see that every ball under  $d_{E|\Delta}$  is convex, we note that it is the preimage of the intersection between  $\sqrt{2}\mathbb{S}_+^{n-1}$  and a Euclidean ball. This preimage is the intersection between  $\Delta$  and a ball under  $d_E$ , which is again convex. ◀

► **Remark 9.** Using the isometry specified in Theorem 8, we can map data from Fisher information space to Euclidean space and apply standard mathematical and computational tools there. Furthermore, the presented derivation works for any Bregman divergence, provided it is decomposable, which ensures the Hessian is diagonal.

#### 4 Information and Loss

In this section, we reverse the direction and measure divergences *to* a point. We use these readings to define the Jensen–Shannon divergence, whose square root we compare to the Fisher information metric.

**Jensen–Shannon divergence.** The *Jensen–Shannon divergence* of a pair  $x, y \in \mathbb{R}_+^n$  is the average relative entropy from  $x$  and  $y$  to the average of the two points:

$$JS(x, y) = \frac{1}{2}[D_E(x||\mu) + D_E(y||\mu)] = \frac{1}{2}[E(x) + E(y)] - E(\mu) \tag{14}$$

$$= \frac{1}{2} \sum_{i=1}^n [x_i \ln \frac{2x_i}{x_i+y_i} + y_i \ln \frac{2y_i}{x_i+y_i}], \tag{15}$$



in which  $\mu = \frac{x+y}{2}$  and we get (15) by noting that  $x + y - 2\mu = 0$ . As pointed out in [19], it measures the loss of efficiency when we encode signals from two probability distributions using their average.

If we substitute any other point  $z \in \mathbb{R}_+^n$  for  $\mu$ , the average relative entropy increases. To prove this, we define  $f(z) = \frac{1}{2}[D_E(x||z) + D_E(y||z)]$ , noting that  $f(\mu) = \text{JS}(x, y)$ . The following lemma can also be found in [5].

► **Lemma 10 (Local Minimum).** *Let  $\mu$  be the average of the points  $x, y \in \mathbb{R}_+^n$ . Then  $f(\mu) \leq f(z)$  for every  $z \in \mathbb{R}_+^n$ , with equality if and only if  $z = \mu$ .*

**Proof.** Computing  $f(z) - f(\mu)$  from the definition, most terms cancel and only  $E(\mu) - E(z) - \langle \nabla E(z), \mu - z \rangle$  remains, which implies  $f(z) - f(\mu) = D_E(\mu||z)$ . It is non-negative by definition and 0 if and only if  $z = \mu$ . ◀

As proved in [19],  $\mathbb{R}_+^n$  together with the square root of the Jensen–Shannon divergence is a metric space. It is however not a *length metric space* because it does not agree with the corresponding *intrinsic metric*, in which the length of a path is measured by taking infinitesimal steps; see also Section 3. It is not difficult to see that the intrinsic metric defined by the Jensen–Shannon divergence is half the intrinsic metric defined by the relative entropy, which is of course the Fisher information metric. Being a metric, the square root of the Jensen–Shannon divergence satisfies the triangle inequality, which implies that it is bounded from above by the corresponding intrinsic metric.

**Jensen–Shannon divergence versus Fisher information metric.** Since the Fisher information metric is twice the intrinsic metric defined by the Jensen–Shannon divergence, we get  $4\text{JS}(x, y) \leq d_E^2(x, y)$ . We will confirm this inequality with an independent argument shortly, and prove  $\frac{4}{\ln 2} = 5.770\dots$  as a tight upper bound on the expansion.

► **Theorem 11 (Jensen–Shannon Divergence vs. Fisher Information).** *The Jensen–Shannon divergence and the squared Fisher information metric satisfy*

$$4\text{JS}(x, y) \leq d_E^2(x, y) \leq \frac{4}{\ln 2} \text{JS}(x, y), \quad (16)$$

$$4\text{JS}(x, y) \leq d_{E|\Delta}^2(x, y) \leq \frac{\sqrt{2}\pi}{\ln 2} \text{JS}(x, y), \quad (17)$$

in which (16) applies to  $\mathbb{R}_+^n$  and (17) to the standard  $(n-1)$ -simplex.

**Proof.** Restricting  $\mathbb{R}_+^n$  to  $\Delta$ , the Jensen–Shannon divergence is unaffected, but the Fisher information metric expands because the shortest paths correspond to great-circle arcs rather than line segments in Euclidean space. The expansion is larger for longer paths, and the supremum expansion rate is  $\pi/\sqrt{8}$ , which is the length of a quarter great-circle over the length of the straight edge connecting its endpoints. Since  $\frac{4}{\ln 2} \frac{\pi}{\sqrt{8}} = \frac{\sqrt{2}\pi}{\ln 2}$ , (16) implies (17).

Returning to  $\mathbb{R}_+^n$ , we note that both the Jensen–Shannon divergence and the squared Fisher information metric are decomposable. It suffices to prove (16) in  $n = 1$  dimension, where we write  $a$  for  $x$  and  $b$  for  $y$ . From (12) and (15), we get  $d_E^2(a, b) = 2(\sqrt{a} - \sqrt{b})^2$  and  $\text{JS}(a, b) = \frac{1}{2}[a \ln \frac{2a}{a+b} + b \ln \frac{2b}{a+b}]$ . Observe that  $d_E^2(Ca, Cb) = C d_E^2(a, b)$  and  $\text{JS}(Ca, Cb) = C \text{JS}(a, b)$  for every  $C > 0$ . We use these properties to eliminate one degree of freedom by setting  $b = t^2 a$  to get

$$\frac{1}{a} d_E^2(a, b) = d_E^2(1, t^2) = 2(1 - t)^2, \quad (18)$$

$$\frac{1}{a} \text{JS}(a, b) = \text{JS}(1, t^2) = \frac{1}{2} \left[ \ln \frac{2}{1+t^2} + t^2 \ln \frac{2t^2}{1+t^2} \right]. \quad (19)$$

We will see shortly that the ratio of these two functions behaves monotonically within  $1 < t < \infty$ , attaining its extreme values in the two limits. These extreme values are the constants in (16), with monotonicity proving the inequalities. Setting  $f(t) = \text{JS}(1, t^2)/d_E^2(1, t^2)$ , the inequalities in (16) are equivalent to

$$\frac{\ln 2}{4} \leq f(t) \leq \frac{1}{4}. \tag{20}$$

Plugging the right-hand sides of (18) and (19) into the definition of the ratio, we get

$$f(t) = \frac{1}{4(t-1)^2} \left[ \ln \frac{2}{t^2+1} + t^2 \ln \frac{2t^2}{t^2+1} \right]. \tag{21}$$

When  $t$  goes to infinity, the first term in (21) goes to 0, while the second term goes to  $\frac{\ln 2}{4}$ , the lower bound in (20). To take the other limit, when  $t$  goes to 1, we use the l'Hopital rule and differentiate the numerator and the denominator twice. For the numerator, we get

$$g(t) = t^2 \ln t^2 - (t^2 + 1) \ln \frac{t^2+1}{2}, \tag{22}$$

$$g'(t) = 2t \ln t^2 - 2t \ln \frac{t^2+1}{2}, \tag{23}$$

$$g''(t) = 2 \ln t^2 - 2 \ln \frac{t^2+1}{2} + \frac{4}{t^2+1}. \tag{24}$$

Setting  $t = 1$ , we get  $g(1) = g'(1) = 0$  and  $g''(1) = 2$ . Similarly, the denominator and its first derivative vanish at  $t = 1$ , its second derivative is 8, and  $f(t)$  goes to  $\frac{1}{4}$ , the upper bound in (20). It remains to show that the ratio is monotonically decreasing, from  $\frac{1}{4} = 0.25$  at  $t = 1$  to  $\frac{\ln 2}{4} = 0.173\dots$  at  $t = \infty$ . We accomplish this by computing the derivative of  $f$  as written in (22):

$$f'(t) = \frac{(t \ln t^2 + t)(t-1)^2}{2(t-1)^4} - \frac{t^2(t-1) \ln t^2}{2(t-1)^4} - \frac{(t \ln \frac{t^2+1}{2} + t)(t-1)^2}{2(t-1)^4} + \frac{(t^2+1) \ln \frac{t^2+1}{2}(t-1)}{2(t-1)^4} \tag{25}$$

$$= \frac{t(t-1) \ln \frac{2t^2}{t^2+1}}{2(t-1)^3} - \frac{t^2 \ln \frac{2t^2}{t^2+1}}{2(t-1)^3} + \frac{\ln \frac{t^2+1}{2}}{2(t-1)^3} = \frac{1}{2(t-1)^3} \left[ \ln \frac{t^2+1}{2} - t \ln \frac{2t^2}{t^2+1} \right]. \tag{26}$$

To prove that  $f'(t)$  is negative for all  $t > 1$ , we rewrite the numerator and compute its first two derivatives:  $u(t) = (t + 1) \ln \frac{t^2+1}{2} - t \ln t^2$ ,  $u'(t) = \ln \frac{t^2+1}{2} + \frac{2t(t+1)}{t^2+1} - \ln t^2 - 2$ ,  $u''(t) = \frac{1}{t(t^2+1)^2} [-2t^3 + 2t^2 + 2t - 2]$ . The numerator of the second derivative factors into  $(t - 1)(-2t^2 + 2)$ , which is clearly negative for all  $t > 1$ . Note that  $u(1) = u'(1) = 0$ . Because  $u''$  is negative, we have  $u'(t) < 0$  and  $u(t) < 0$  for all  $t > 1$ . The latter inequality is equivalent to  $f'(t) < 0$  for all  $t > 1$ , which implies that  $f$  is monotonically decreasing. This implies (20) and completes the proof. ◀

## 5 TDA in Information Space

While the preceding sections shed light on distances with information-theoretic foundations, we will now connect them to topological data analysis. We begin with the introduction of standard results from topological data analysis, which we will then use to shed light on relations between different notions of distance.

**Half-diameter functions.** It will be convenient to consolidate notation by writing  $\mathbb{X}$  for a topological space and  $\theta: \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$  for a distance. Here we consider  $\mathbb{X}$  to either be  $\mathbb{R}_+^n$  or  $\Delta^{n-1}$ , and  $\theta$  to either be the square root of the Jensen–Shannon divergence,  $\sqrt{\text{JS}}$ , or the Fisher information metric,  $d_E$ . When we consider the set of all subsets, we will always exclude the empty set and write  $\mathcal{P}_0(\mathbb{X}) = \{Q \subseteq \mathbb{X} \mid Q \neq \emptyset\}$ . Given  $\mathbb{X}$  and  $\theta$ , the *half-diameter function*  $\mathcal{D}_\theta: \mathcal{P}_0(\mathbb{X}) \rightarrow \mathbb{R}$  is defined by mapping every non-empty subset  $Q \subseteq \mathbb{X}$  to

$$\mathcal{D}_\theta(Q) = \frac{1}{2} \sup_{x,y \in Q} \theta(x,y), \tag{27}$$

which we call the *half-diameter* of  $Q$ . Since  $\theta$  is a metric, we can define the *Hausdorff distance* between two subsets of  $\mathbb{X}$  as the larger of the two directed such distances:  $\mathcal{H}_\theta(P, Q) = \max\{\mathcal{H}_\theta(P\|Q), \mathcal{H}_\theta(Q\|P)\}$ , in which  $\mathcal{H}_\theta(P\|Q) = \max_{x \in P} \min_{y \in Q} \theta(x, y)$ . This is a metric on the compact sets in  $\mathcal{P}_0(\mathbb{X})$ , and under it, the half-diameter function is 1-Lipschitz and therefore continuous. Finally, we note that the half-diameter function is *monotonic*, that is:  $\mathcal{D}_\theta(P) \leq \mathcal{D}_\theta(Q)$  whenever  $\emptyset \neq P \subseteq Q \subseteq \mathbb{X}$ .

Data analysis starts with a finite set of points,  $X \subseteq \mathbb{X}$ . Let  $K = \mathcal{P}_0(X)$ , noting that this is a full simplex, and write  $f: K \rightarrow \mathbb{R}$  for the restriction of the half-diameter function. For each  $r \geq 0$ , we write  $K_r = f^{-1}[0, r]$  for the corresponding sublevel set, which consists of all simplices in  $K$  with half-diameter  $r$  or less. By the monotonicity of  $f$ , each sublevel set is a subcomplex of  $K$ . We refer to  $K_r$  as the *Vietoris–Rips complex* of  $X$  and  $\theta$  for radius  $r$ . By construction,  $K_r$  is a *clique complex*: a simplex belongs to  $K_r$  if and only if all edges of the simplex belong to  $K_r$ . It follows that the edges in  $K_r$  determine the entire complex. This has computational advantages because triangles and higher-dimensional simplices can be treated implicitly, computing their properties from the edges only when needed; see e.g. the Ripser software of Bauer [6]. It has modeling disadvantages because triangles and higher-dimensional simplices carry no new structural information. We will return to this point when we consider Čech complexes as an alternative construction.

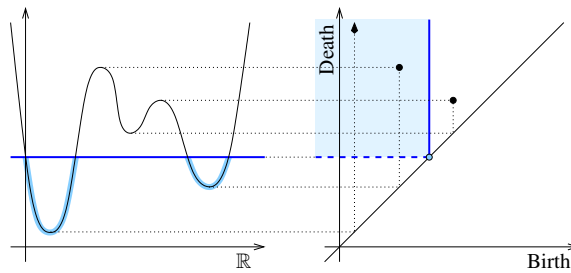
**Persistent homology.** Persistent homology groups have been introduced in [16] to measure holes in proteins. The concept has since found many applications inside and outside mathematics. The idea is however older, and the earliest reference we know is a paper by Morse [26], which develops the concept as a tool in the study of minimal surfaces.

The concept starts with the classic notion of homology groups, which formalize our notion of how a complex or really any topological space is connected; see e.g. [21]. It does so by counting the holes as ranks of abelian groups. Choosing the coefficients in the construction of these groups as elements of a field, the groups are vector spaces and their ranks are the familiar dimensions from linear algebra. Returning to  $f: \mathcal{P}_0(X) \rightarrow \mathbb{R}$ , we index the finitely many sublevel sets consecutively from 1 to  $m$ . Mapping each complex  $K_\ell$  to its  $p$ -th homology group, for some fixed integer  $p$ , we get a sequence of vector spaces,  $H_\ell = H_p(K_\ell)$ :

$$\begin{array}{ccccccccccc} \dots & \subseteq & K_{i-1} & \subseteq & K_i & \subseteq & \dots & \subseteq & K_{j-1} & \subseteq & K_j & \subseteq & \dots \\ & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow & & \\ \dots & \rightarrow & H_{i-1} & \rightarrow & H_i & \rightarrow & \dots & \rightarrow & H_{j-1} & \rightarrow & H_j & \rightarrow & \dots \end{array}$$

The inclusions  $K_i \subseteq K_j$  induce linear maps  $h_{i,j}: H_i \rightarrow H_j$ , and by the functoriality of homology, the inclusions commute with these maps. We call the row of complexes a *filtration* and the row of vector spaces together with the maps connecting them a *persistence module*. This is the essential concept that permits us to measure how long holes persist in the filtration. To be specific, let  $1 \leq i \leq j \leq m$  and call the image of the map from  $H_i$  to  $H_j$  a *persistent homology group*,  $\text{im } h_{i,j} \subseteq H_j$ . Its rank is the number of holes in  $K_i$  that are still holes in  $K_j$ . A homology class  $\alpha$  in  $H_i$  is *born* at  $K_i$  if  $\alpha$  does not belong to the image of  $H_{i-1}$ , and it *dies entering*  $K_j$  if  $\alpha$  enters the preimage of  $H_{i-1}$  when we go from  $K_{j-1}$  to  $K_j$ . Writing  $r_i$  for the smallest value such that  $K_i = f^{-1}[0, r_i]$ , we define the *persistence* of  $\alpha$  as  $|r_j - r_i|$ .

To summarize the information in the persistence module, we represent a coset of homology classes born at  $K_i$  and dying entering  $K_j$  by the point  $(r_i, r_j)$ ; see Figure 3, setting  $r_j = \infty$  if the classes never die. The result is a multiset of points in the extended plane, which we refer to as the *persistence diagram* of  $f$ , denoted  $\text{Dgm}(f)$ . The ranks of the persistent homology groups can be read as the numbers of points in the upper-left quadrants defined by corners on or above the diagonal; see again Figure 3.



■ **Figure 3** *Left:* the graph of a function on  $\mathbb{R}$ . *Right:* the persistence diagram of the function, with two finite points and one point at infinity. The components of the highlighted sublevel set are counted by the two points in the corresponding upper-left quadrant.

**Stability.** An important property of persistence diagrams is their stability with respect to perturbations. To explain this, we define the *bottleneck distance* between two persistence diagrams as the length of the longest edge in a minimizing matching. To finesse the difficulty caused by different cardinalities, we add infinitely many copies of every point along the diagonal to each diagram and define  $W_\infty(\text{Dgm}(f), \text{Dgm}(g)) = \inf_\beta \sup_A \|A - \beta(A)\|_\infty$ , in which  $\beta: \text{Dgm}(f) \rightarrow \text{Dgm}(g)$  is a bijection and  $A$  is a point in  $\text{Dgm}(f)$ . The original proof of stability in [13] bounds the bottleneck distance by the  $L_\infty$ -distance between the functions. In later developments, [7, 12] compare the diagrams directly with the persistence modules they summarize. Letting  $\mathcal{F}$  and  $\mathcal{G}$  be the persistence modules defined by the sublevel sets of  $f$  and  $g$ , indexed by real numbers for convenience, we say they are  $\varepsilon$ -interleaved if there are maps  $\phi_r: \mathbb{F}_r \rightarrow \mathbb{G}_{r+\varepsilon}$  and  $\psi_r: \mathbb{G}_r \rightarrow \mathbb{F}_{r+\varepsilon}$  that commute with the maps within the modules. The *interleaving distance*, denoted  $I(\mathcal{F}, \mathcal{G})$ , is the infimum  $\varepsilon \geq 0$  for which  $\mathcal{F}$  and  $\mathcal{G}$  are interleaved. With this notation, we have

$$W_\infty(\text{Dgm}(f), \text{Dgm}(g)) = I(\mathcal{F}, \mathcal{G}) \leq \|f - g\|_\infty. \tag{28}$$

Suppose  $\theta$  is a metric, and  $X, Y \subseteq \mathbb{X}$  are finite sets with Hausdorff distance  $\varepsilon$ . Letting  $f$  and  $g$  be the restrictions of the half-diameter function to  $\mathcal{P}_0(X)$  and to  $\mathcal{P}_0(Y)$ , it is not difficult to see that the interleaving distance between the corresponding persistence modules is at most  $\varepsilon$ . We state this straightforward consequence of stability for later reference.

► **Theorem 12 (Stability for Metrics).** *Let  $\theta: \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$  be a metric,  $X, Y \subseteq \mathbb{X}$  finite, and  $f: \mathcal{P}_0(X) \rightarrow \mathbb{R}$ ,  $g: \mathcal{P}_0(Y) \rightarrow \mathbb{R}$  induced by the half-diameter function of  $\mathbb{X}$  and  $\theta$ . Then the bottleneck distance between  $\text{Dgm}(f)$  and  $\text{Dgm}(g)$  is bounded from above by the Hausdorff distance between  $X$  and  $Y$ .*

**Approximation.** Since  $\sqrt{\text{JS}}$  and  $d_E$  are both metrics, Theorem 12 applies. Specifically, the mapping from the subsets of  $\mathbb{X}$  to the space of persistence diagrams defined by  $\sqrt{\text{JS}}$  is 1-Lipschitz, and so is the mapping defined by  $d_E$ . Writing  $f_j, f_i: X \rightarrow \mathbb{R}$  for the maps in which  $j = \sqrt{\text{JS}}$  and  $i = \frac{1}{2}d_E$ , the persistence diagrams of  $f_j$  and  $f_i$  are generally different, and we can use this difference to compare the two distances. By Theorem 11, we have

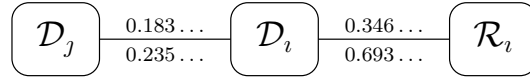
$$\frac{1}{C} j(x, y) \leq i(x, y) \leq C j(x, y) \tag{29}$$

for  $C^2 = 1/\ln 2 = 1.442\dots$ , in which the two distances between  $x$  and  $y$  are measured in  $\mathbb{R}_+^n$ . Rewriting (29) in terms of sublevel sets, we get  $f_j^{-1}[0, \frac{1}{C}r] \subseteq f_i^{-1}[0, r] \subseteq f_j^{-1}[0, Cr]$ . To get constant interleaving distance, we re-index the vector spaces in the persistence modules  $\mathcal{F}_j$

## 31:12 Topological Data Analysis in Information Space

of  $f_j$  and  $\mathcal{F}_i$  of  $f_i$  by substituting  $\ln r$  for  $r$ . Hence,  $I(\mathcal{F}_j, \mathcal{F}_i) \leq \ln C = 0.183\dots$ . By (28), this implies that the persistence diagrams of  $\mathcal{F}_j$  and  $\mathcal{F}_i$  are very similar, with bottleneck distance at most  $0.183\dots$ . We get slightly different results if we measure the distances in  $\Delta$ . Specifically, we get  $C^2 = \sqrt{2}\pi/(4 \ln 2) = 1.602\dots$  in (29) and interleaving distance at most  $\ln C = 0.235\dots$ .

► **Theorem 13 (Approximation).** *Let  $j = \sqrt{JS}$  and  $i = \frac{1}{2}d_E$ ,  $X \subseteq \mathbb{X}$  finite, and  $\mathcal{F}_j, \mathcal{F}_i$  the persistence modules defined by the restrictions of  $\ln \mathcal{D}_j, \ln \mathcal{D}_i$  to  $\mathcal{P}_0(X)$ . Then  $\mathcal{F}_j$  and  $\mathcal{F}_i$  are interleaved, with the interleaving distance labeling the first edge in Figure 4.*



■ **Figure 4** Recall that  $j = \sqrt{JS}$ ,  $i = \frac{1}{2}d_E$ , and  $\mathcal{D}_j, \mathcal{D}_i, \mathcal{R}_i$  are the corresponding half-diameter and radius functions. Each edge is labeled by two interleaving distances: above the edge for  $\mathbb{X} = \mathbb{R}_+^n$  and below the edge for  $\mathbb{X} = \Delta$ .

► **Remark 14.** The two constants labeling the first edge in Figure 4 can be improved by taking advantage of the asymmetry of the inequalities in (16) and (17). Indeed, while being symmetric, (29) is strictly weaker than (16).

**Radius function by comparison.** We contrast the half-diameter functions with a different construction, which we introduce for the Fisher information metric. Given  $\mathbb{X}$ , the *radius function*  $\mathcal{R}_i: \mathcal{P}_0(\mathbb{X}) \rightarrow \mathbb{R}$  is defined by mapping every non-empty subset to

$$\mathcal{R}_i(Q) = \inf_{z \in \mathbb{X}} \sup_{x \in Q} d_E(z, x), \quad (30)$$

which we call the *radius* of  $Q$ . Again, the radius function is monotonic, but in contrast to the half-diameter functions, the simplices of dimension 2 and higher encode structural information that is not already contained in the edges. To explain, we write  $B_r(x)$  for the set of points  $d_E(x, z) \leq r$ . The radius function signals at which radius the balls defined by  $Q$  have a non-empty common intersection:  $r \geq \mathcal{R}_i(Q)$  if and only if  $\bigcap_{x \in Q} B_r(x) \neq \emptyset$ .

To see why this property is useful, consider a finite set  $X \subseteq \mathbb{X}$ , write  $g: \mathcal{P}_0(X) \rightarrow \mathbb{R}$  for the restriction of the radius function, and note that  $G_r = g^{-1}[0, r]$  is a subcomplex of  $G = \mathcal{P}_0(X)$ . We refer to  $G_r$  as the *Čech complex* of  $X$  and  $i$  for radius  $r$ . Furthermore, write  $X_r = \bigcup_{x \in X} B_r(x)$  for the union of the balls with radius  $r$ . We thus have two filtrations: the complexes  $G_r$  and the subspaces  $X_r$  of  $\mathbb{X}$ . The Nerve Theorem implies that  $G_r$  and  $X_r$  have isomorphic homology groups [9, 24], and this property extends:

► **Theorem 15 (Isomorphism of Modules).** *The persistence modules defined by the complexes  $G_r = g^{-1}[0, r]$  and the subspaces  $X_r = \bigcup_{x \in X} B_r(x)$  are isomorphic and the corresponding persistence diagrams are equal.*

Theorem 15 suggests we consider the common persistence diagram of the subspaces  $X_r$  and the Čech complexes  $G_r$  the “correct” diagram of the data set,  $X$ . Not surprisingly, the persistence diagram of the corresponding Vietoris–Rips complexes is generally different. We therefore have the opportunity to quantify the error we tolerate when we compute persistence for the Vietoris–Rips instead of the Čech complexes. Clearly,  $\mathcal{D}_i(Q) \leq \mathcal{R}_i(Q)$ , but how different can the two values be? Considering first the case  $\mathbb{X} = \mathbb{R}_+^n$ , we recall Theorem 8 (12). As proved in [15, page 62], the radius of a simplex in Euclidean space is at most  $\sqrt{2}$  times the

half-length of its longest edge. The isometry between the Fisher information distance and the Euclidean distance implies  $\mathcal{R}_i(Q) \leq \sqrt{2}\mathcal{D}_i(Q)$  in  $\mathbb{R}_+^n$  equipped with the Fisher information metric. Hence,  $\frac{1}{C}\mathcal{D}_i(Q) \leq \mathcal{R}_i(Q) \leq C\mathcal{D}_i(Q)$  with  $C^2 = 2$ . The corresponding interleaving distance is  $\ln C = 0.346\dots$ , as recorded in Figure 4. The calculation of the interleaving distance for points in  $\Delta$  is similar, except that we do not have a Euclidean result ready to use. We will prove shortly that for simplices  $Q \subseteq \mathbb{S}_+^{n-1}$ , we have  $\frac{1}{C}\mathcal{D}_i(Q) \leq \mathcal{R}_i(Q) \leq C\mathcal{D}_i(Q)$  with  $C^2 = 4$ . The corresponding interleaving distance is  $\ln C = 0.693\dots$ . We thus observe that there is a higher penalty for using Vietoris–Rips complexes in  $\Delta$  than there is in  $\mathbb{R}_+^n$ . We finally prove the required Euclidean result. We recall that the radius and half-diameter of a set  $Q$  are defined in (27) and (30).

► **Lemma 16** (Interleaving on the Sphere). *Let  $Q$  be a finite set of points in  $\mathbb{S}_+^{n-1}$ . Measuring distances in the  $(n-1)$ -sphere, the radius of  $Q$  is at most twice the half-diameter of  $Q$ , and this bound is tight as  $n$  goes to infinity.*

**Proof.** Suppose first that  $Q$  is a regular  $(n-1)$ -simplex. It cannot be larger than the  $(n-1)$ -simplex  $Q_0$  that covers the entire positive orthant of the sphere. Its vertices are the  $n$  unit coordinate vectors, and its center lies on the diagonal, with all coordinates equal to  $1/\sqrt{n}$ . The scalar product of the vectors connecting the origin to the center and a vertex of  $Q_0$  is  $1/\sqrt{n}$ , which goes to 0 as  $n$  goes to infinity. It follows that the spherical distance between the two points approaches  $\frac{\pi}{2}$ . Since the half-diameter of  $Q_0$  is  $\frac{\pi}{4}$  and the radius of  $Q_0$  approaches  $\frac{\pi}{2}$  as  $n$  goes to infinity, the claimed inequality holds for regular  $(n-1)$ -simplices and cannot be improved unless we assume a fixed finite dimension.

To see that the claimed inequality holds for general sets  $Q$ , we recall that the spherical distance is a metric. Let  $B_r(x)$  be the ball with Fisher information radius  $r$  centered at  $x \in Q$ . Setting  $r$  equal to the diameter of  $Q$ , every such ball contains all points of  $Q$ , which implies that the common intersection is non-empty. Hence, the radius is at most the diameter. ◀

## 6 Discussion

The main contribution of this paper is a connection between information theory and topological data analysis, thus complementing the established field of information geometry [3] with topological methods. Concretely, we study notions of distance with information theoretic foundations, focusing on properties relevant in their application in topological data analysis. As an immediate practical gain, we can now use computational topology tools that work for Euclidean distance to analyze data measured with Fisher information metric. It is thus easy to experiment with data in information space without developing new and specialized software. Another option is to construct Vietoris–Rips complexes with the Ripser software [6], which at the time of writing this paper is significantly more efficient than constructing nerves of entropy balls.

The first concrete application of the theory set up in this paper is described in [17], where Euclidean and Fisher information point processes are studied and compared through the lens of integral geometry. Many challenges remain before topology can be fully utilized in high-dimensional data analysis tasks. In particular, a proof of stability of topological descriptors based on Bregman divergences remains elusive, and we hope that the results presented here are a step in this direction.

The cosine dissimilarity is an effective measure for text documents in the vector space model [22]. It represents each document as a discrete probability distribution, and – just like the Antonelli isometry – maps this distribution to a point in the positive orthant of a sphere. This resembles the Fisher information distance in  $\Delta$ , except that the latter uses the angle

rather than its cosine. This suggests an information-theoretic interpretation for the cosine measure. Using the generalized Antonelli isometry, we can devise similarly simple measures for other decomposable Bregman divergences. Will they prove equally effective in practice?

---

### References

- 1 R. Adler. TOPOS, and why we should care about it. *IMS Bulletin*, 43, 2014.
- 2 E. Akin. *The Geometry of Population Genetics*. Springer, Berlin, 1979.
- 3 S. Amari and H. Nagaoka. *Methods of Information Geometry*. Amer. Math. Soc., Providence, Rhode Island, 2000.
- 4 P.L. Antonelli et al. The geometry of random drift I-VI. *Adv. Appl. Prob.*, 9-12, 1977-80.
- 5 A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, 2005.
- 6 U. Bauer. Fast Rips in the browser, 2016. URL: [www.ctralie.com/Software/jstDA](http://www.ctralie.com/Software/jstDA).
- 7 U. Bauer and M. Lesnick. Induced matchings and the algebraic stability of persistence barcodes. *J. Comput. Geom.*, 6:162–191, 2015.
- 8 J.-D. Boissonnat, F. Nielsen, and R. Nock. Bregman Voronoi diagrams. *Discrete Comput. Geom.*, 44:281–307, 2010.
- 9 K. Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.*, 35:217–234, 1948.
- 10 L.M. Bregman. The relaxation method of finding the common point of convex sets and its applications to the solution of problems in convex programming. *USSR Comput. Math. Math. Phys.*, 7:200–217, 1967.
- 11 G. Carlsson. Topology and data. *Bull. Amer. Math. Soc.*, 46:255–308, 2009.
- 12 F. Chazal, de Silva, V., M. Glisse, and S. Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics, Springer, 2016.
- 13 D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37:103–120, 2007.
- 14 I. Csiszár and J. Körner. *Coding Theory for Discrete Memoryless Systems*. Cambridge Univ. Press, Cambridge, England, 2011.
- 15 Edelsbrunner, H., and J.L. Harer. *Computational Topology. An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2010.
- 16 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.
- 17 H. Edelsbrunner and A. Nikitenko. Random inscribed polytopes have similar radius functions as Poisson–Delaunay mosaics, 2016. [arXiv:1705.02870](https://arxiv.org/abs/1705.02870).
- 18 H. Edelsbrunner and H. Wagner. Topological data analysis with Bregman divergences. In “*Proc. 33rd Ann. Symp. Comput. Geom., 2017*”, 1–16, pages 1–16, 2017.
- 19 D.M. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Trans. Inform. Theory*, 49:1858–1860, 2003.
- 20 Edelsbrunner H., Ž Virk, and H. Wagner. Smallest Enclosing Spheres and Chernoff Points in Bregman Geometry. In “*Proc. 34rd Ann. Symp. Comput. Geom., 2018*”, 2018.
- 21 A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.
- 22 A. Huang. Similarity measures for text document clustering. In “*Proc. 6th New Zealand Computer Science Research Student Conference*”, 49–56, pages 49–56, 2008.
- 23 S. Kullback and R.A. Leibler. On information and sufficiency. *Ann. Math. Stat.*, 22:79–86, 1951.
- 24 J. Leray. Sur la forme des espaces topologiques et sur les points fixes des représentations. *J. Math. Pure Appl.*, 24:95–167, 1945.
- 25 E. Merelli, M. Rucco, P. Sloot, and L. Tesei. Topological characterization of complex systems: using persistent entropy. *Entropy*, 17:6872–6892, 2015.
- 26 M. Morse. Rank and span in functional topology. *Ann. Math.*, 41:419–454, 1940.
- 27 F. Nielsen and R. Nock. On the smallest enclosing information disk. In “*Proc. 18th Canad. Conf. Comput. Geom.*”, 2006.



# Cubic Planar Graphs That Cannot Be Drawn On Few Lines

David Eppstein

Computer Science Department, University of California, Irvine, USA  
eppstein@uci.edu

---

## Abstract

---

For every integer  $\ell$ , we construct a cubic 3-vertex-connected planar bipartite graph  $G$  with  $O(\ell^3)$  vertices such that there is no planar straight-line drawing of  $G$  whose vertices all lie on  $\ell$  lines. This strengthens previous results on graphs that cannot be drawn on few lines, which constructed significantly larger maximal planar graphs. We also find apex-trees and cubic bipartite series-parallel graphs that cannot be drawn on a bounded number of lines.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Human-centered computing  $\rightarrow$  Graph drawings

**Keywords and phrases** graph drawing, universal point sets, collinearity

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.32

**Funding** *David Eppstein*: Supported in part by NSF grants CCF-1618301 and CCF-1616248.

## 1 Introduction

A number of works in graph drawing and network visualization have considered drawing graphs with line segments as edges and with the vertices placed on few lines, or on a minimal number of lines. Even very strong constraints, such as restricting the vertices of a drawing to only two lines, allow many graphs to be drawn [16]: every *weakly leveled* graph drawing (a planar drawing on any number of parallel lines with every edge connecting two vertices on the same or adjacent lines) can be converted into a drawing on two crossing lines that spirals around the crossing. This conversion allows, for instance, all trees, all outerplanar graphs, all Halin graphs, all squaregraphs (graphs in which all bounded faces have exactly four sides and all vertices not on the unbounded face have at least four neighbors) and all grid graphs (Figure 1) to be drawn on two lines [1, 2, 15].

Additional past results in this area include:

- Fixed-parameter tractable algorithms for drawing planar graphs without crossings with all vertices on  $\ell$  parallel lines, based on the fact that a graph with such a drawing must have pathwidth  $O(\ell)$  [8].
- NP-hardness of crossing minimization for graphs drawn with vertices on two parallel lines (with a fixed assignment of vertices to lines but variable placement of each vertex along each line) and of finding large crossing-free subgraphs [12, 13].
- NP-hardness of recognizing the graphs that can be drawn without crossing with all vertices on three parallel but non-coplanar lines in three-dimensional space, or on three rays in the plane with a common apex and bounding three wedges with angles less than  $\pi$  [2]. More generally, the number of parallel three-dimensional lines (in sufficiently general position) needed for a crossing-free drawing of a graph is the *track number* of the graph [10, 11]. It is closely related to the volume of three-dimensional grid drawings, and can be bounded by the pathwidth of the graph [9].
- $\exists\mathbb{R}$ -completeness and fixed-parameter tractability of deciding whether a given graph can be drawn without crossing with all edges on  $\ell$  lines (not required to be parallel) in two or three dimensions [4].

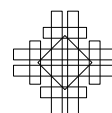


© David Eppstein;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

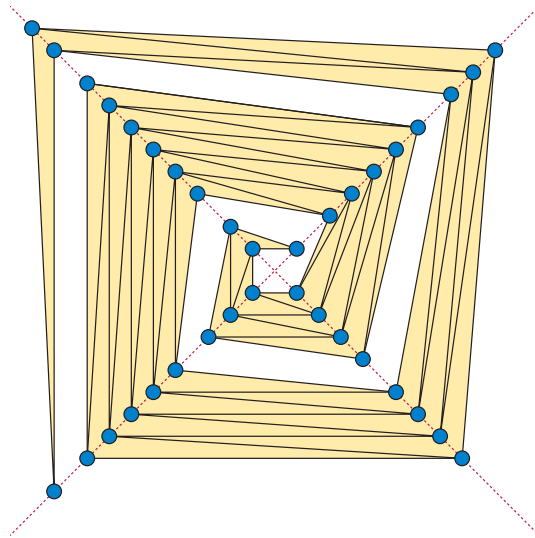
Editors: Gill Barequet and Yusu Wang; Article No. 32; pp. 32:1–32:15  
Leibniz International Proceedings in Informatics



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







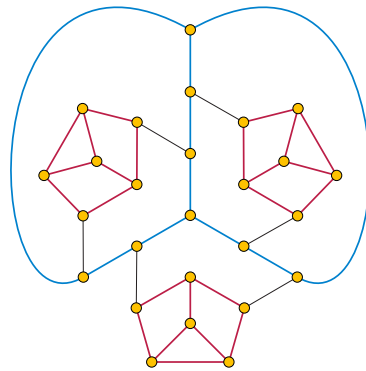
■ **Figure 1** Spiraling around the central crossing allows every weakly leveled drawing (in this case, a drawing of a grid graph) to be converted to a drawing with all vertices on two lines.

- Implementation of a tester for drawing graphs without crossings on two lines using integer linear programming and SAT solvers, and an examination of the subclasses of planar graphs that can be drawn without crossings with all vertices on two lines [16].
- The existence of families of planar graphs that cannot be drawn without crossings on any fixed number of lines, no matter how the lines are arranged in the plane [5, 14, 19].

In this paper we strengthen the final result of this listing, the existence of families of planar graphs that cannot be drawn on any fixed number of lines, in two ways.

First, we greatly improve the size bounds for these difficult-to-draw graphs. The previous bounds of Ravsky, Chaplick, et al. [5, 19] are based on the observation that, in a maximal planar graph, a line through  $q$  vertices implies the existence of a path in the dual graph of length  $\Omega(q)$ . However, there exist  $n$ -vertex maximal planar graphs for which the longest dual path has length  $O(n^c)$  for some constant  $c < 1$  called the *shortness exponent*. In these graphs, at most  $O(n^c)$  vertices can lie on one line, so the number of lines needed to cover the vertices of any drawing of such a graph is  $\Omega(n^{1-c})$ . Based on this reasoning, they showed the existence of  $n$ -vertex graphs requiring  $O(n^{0.01})$  lines to cover the vertices of any drawing. Inverting this relationship, graphs that cannot be drawn on  $\ell$  lines can have a number of vertices that is only polynomial in  $\ell$ , but that polynomial is roughly  $\ell^{100}$ . Alternatively, it can be proven by a straightforward induction that a special class of maximal planar graphs, the planar 3-trees, cannot be drawn on a constant number of lines, but the proof only shows that the required number of lines for these graphs is at least logarithmic [14]. Inverting this relationship, the graphs of this type that cannot be drawn on  $\ell$  lines have size exponential in  $\ell$ . In this paper, we prove polynomial bounds with a much smaller exponent than 100.

Second, we show that the property of requiring many lines extends to a broader class of graphs. Both classes of counterexamples discussed above involve maximal planar graphs, graphs in which every face of the embedding is a triangle. Maximality seems a necessary part of the proofs based on the shortness exponent, as the connection between numbers of vertices on a single line and lengths of dual paths does not necessarily hold for other classes of planar graphs. In contrast, based on computational experiments, Firman et al. [16] conjectured that cubic (that is, 3-regular) planar graphs can always be drawn without crossings on only



■ **Figure 2** A cubic planar graph that cannot be drawn on two lines, giving a counterexample to a conjecture of Firman et al. [16]. Its inability to be drawn on two lines has been verified computationally by Firman et al.

two lines. The cubic graphs are distinct from the known examples of graphs that require their vertices to lie on many lines, as a maximal planar graph larger than  $K_4$  cannot be cubic. Their conjecture inspired the present work, and a counterexample to it found by the author (Figure 2) led to our main results. In this work, we provide examples of graphs that require many lines but are cubic, providing stronger counterexamples to the conjecture of Firman et al. Moreover these graphs do not contain any triangles, showing that the presence of triangles is not a necessary component of graphs that require many lines.

More specifically, we prove:

► **Theorem 1.** *For every  $\ell$  there exists a graph  $G_\ell$  that is cubic, 3-vertex-connected, planar, and bipartite, with  $O(\ell^3)$  vertices, such that every straight-line planar drawing of  $G_\ell$  requires more than  $\ell$  lines to cover all vertices of the drawing.*

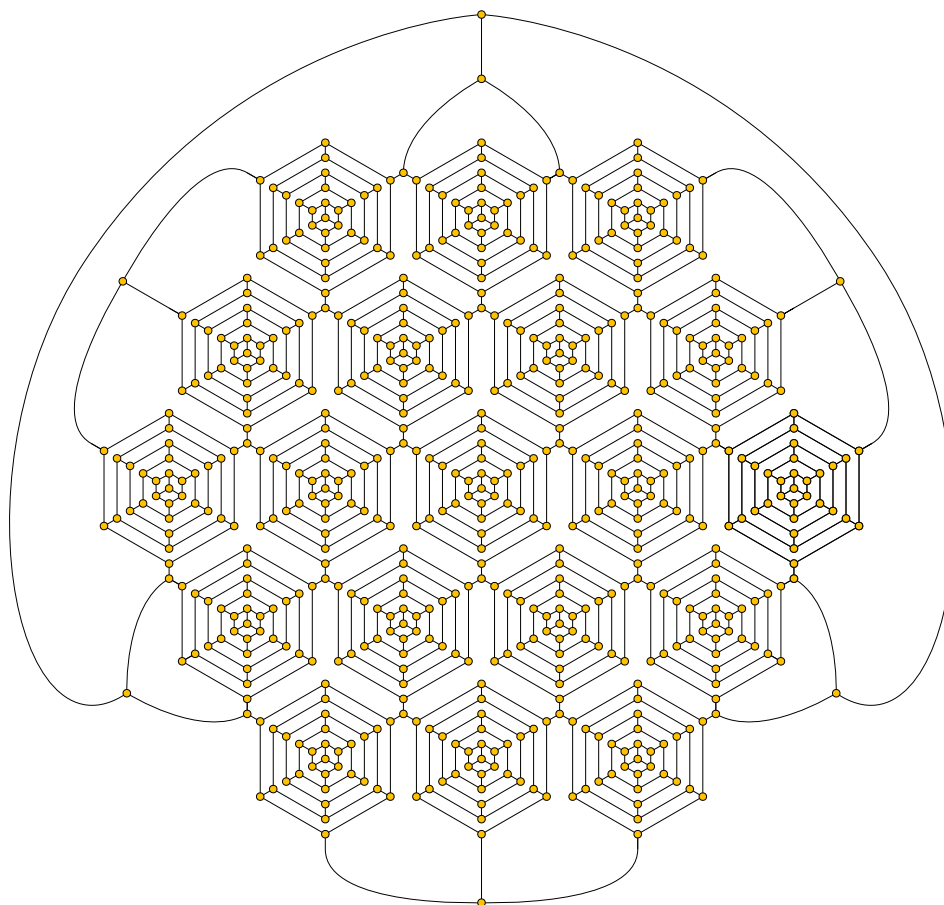
Additionally, we can construct  $G_\ell$  in such a way that it is drawable with its vertices on  $O(\ell)$  lines, or such that it has bounded pathwidth. In particular, this proves that the relation between pathwidth and number of lines proven by Dujmović et al. [8] is not bidirectional. Every graph that can be drawn without crossings on a given number of parallel lines has pathwidth bounded linearly in the number of lines, but the number of lines needed to draw a planar graph cannot be bounded by a function of its pathwidth, regardless of whether we constrain the lines to be parallel.

Using similar methods, we also prove:

► **Theorem 2.** *For every  $\ell$  there exists a subcubic series-parallel graph that cannot be drawn with its vertices on  $\ell$  lines, and an apex-tree (a graph formed by adding one vertex to a tree) that cannot be drawn with its vertices on  $\ell$  lines.*

Theorem 2 stands in contrast to the fact that all trees and all outerplanar graphs can be drawn on only two crossed lines. In Theorem 2, both the series-parallel graph and apex-tree can be made to be bipartite, and the apex-tree can be made subcubic at all tree vertices.

These results lower the treewidth of the known graphs that cannot be drawn on a constant number of lines from three [14] to two, and they show that adding one vertex to a graph can change the number of lines needed to draw it from two to any larger number. The apex-tree graphs are also central to recent research characterizing the minor-closed families of bounded layered pathwidth [7]. However, for Theorem 2 we do not have a polynomial size bound on the graphs that we construct; instead, they are exponential in size. At least in the case of apex-trees, this exponential size blowup is necessary, as we finally prove:



■ **Figure 3** The overall construction plan of our graph:  $O(\ell^2)$  copies of a subgraph formed from  $O(\ell)$  nested hexagons.

► **Theorem 3.** *Every apex-tree with  $n$  vertices has a planar embedding that can be drawn with its vertices on  $O(\log n)$  parallel lines.*

## 2 Counterexamples of cubic size

### 2.1 Overview

The overall strategy of our construction for graphs of size  $O(\ell^3)$  that cannot be drawn on  $\ell$  lines is illustrated in Figure 3. As can be seen in the figure, the graph consists of a number of subunits, each formed by a set of nested hexagons, connected to each other by triples of edges. These subunits resemble the *nested triangles graph* frequently used as a counterexample in graph drawing [6, 17], but are based on hexagons rather than triangles.

The figure shows faces of three types: triples of quadrilaterals at the center of each set of nested hexagons, non-convex L-shaped hexagonal faces between pairs of hexagons in each set of nested hexagons, and Y-shaped dodecagonal faces between triples of subunits. Because the graph is planar and each bounded face has an even number of sides, the graph is also bipartite. It is straightforward to add a small number of additional vertices surrounding the sets of nested hexagons (as shown) to complete the graph to one that is cubic, 3-vertex-connected,

and still bipartite. Because it is 3-vertex-connected, it has a unique planar embedding, the one shown, up to the choice of which face of the embedding is the outer face. With at most one exception (the subunit that includes the outer face), all subunits must be drawn as shown in the figure (topologically but not geometrically), within disjoint hexagonal regions of the plane.

The drawing also shows that each subunit can be drawn using only three lines. Indeed, the number of lines needed to cover all of the vertices in Figure 3 (and in similar figures with more subunits) is proportional only to the square root of the number of subunits. Nevertheless, we shall show that, if there are enough subunits relative to the number of lines, then at least one of the subunits will be difficult to draw on few lines. More specifically, for a given parameter  $\ell$  (a number of lines that should be too small to cover all vertices of the drawing) we will choose the number of subunits to be at least  $\binom{\ell}{2} + 2$ , two more than the largest possible number of crossing points that  $\ell$  lines can have. In this way, there will be at least one subunit that does not include the outer face of the embedding, and does not surround any crossing points of the lines.

We will show that, for subunits formed from a sufficiently large number of nested hexagons (linear in  $\ell$ ), it is not possible to draw the subunit on  $\ell$  lines without surrounding any crossing points. Because, nevertheless, one of the subunits must fail to surround any crossing points, it cannot be drawn on  $\ell$  lines. It follows that the whole graph also cannot be drawn on  $\ell$  lines.

## 2.2 Nested polygons with no surrounded crossings

To formalize the subunits of the drawing of Figure 3, we define a  $(p, r)$ -nest, for positive integers  $p$  and  $r$ , to be a collection of  $r$  disjoint simple  $p$ -gons (not necessarily convex), together with one additional point (the *egg*), such that each  $p$ -gon contains the egg. Because the  $p$ -gons are disjoint and all contain the egg, it is necessarily the case (by the Jordan curve theorem) that each two of the  $p$ -gons are nested, one inside the other. Then the subunits of Figure 3 form a  $(6, r)$ -nest, for some  $r$ , together with some additional graph edges between consecutive cycles that force the cycles to be nested within each other but play no additional role in our analysis.

In Figure 3, each  $(6, r)$ -nest is drawn with all hexagon vertices on three lines that cross at the egg of the nest. More generally, whenever  $p$  is even, a  $(p, r)$ -nest can be drawn on only two crossing lines, with the egg at the crossing point; when  $p$  is odd, three lines suffice. However, in all of these drawings, a crossing point of the lines is contained within at least one polygon of the nest. We will show that, when this does not happen, nests require  $\Omega(r/p)$  lines to cover all of their points.

► **Lemma 4.** *Let  $p$  be a positive integer, let  $P$  be a simple  $p$ -gon, and let  $L$  be a line. Then  $L$  intersects the interior of  $P$  in at most  $\lfloor p/2 \rfloor$  open line segments.*

**Proof.** Each line segment begins and ends at points where  $L$  intersects  $P$  either at a vertex or at an interior point of one of the sides of  $P$ . If the segment endpoint is at a crossing of  $L$  with an interior point of a side of  $P$ , then that point is the endpoint of only one segment of  $L$ , and is the only point of intersection of  $L$  with that side. If the segment endpoint is a vertex of  $P$ , then it may be the endpoint of two segments of  $L$ , but in that case it is the only point of intersection of  $L$  with both sides incident to the vertex. So in either case each endpoint of a segment of  $L$  uses up at least one side of  $P$ . As  $P$  has  $p$  sides, the number of segments is at most  $\lfloor p/2 \rfloor$ . ◀

► **Corollary 5.** *Let  $\mathcal{A}$  be an arrangement of  $\ell$  lines, and let  $P$  be a simple  $p$ -gon whose interior is disjoint from the crossings of  $\mathcal{A}$ . Then the lines of  $\mathcal{A}$  intersect the interior of  $P$  in at most  $\ell \cdot \lfloor p/2 \rfloor$  open line segments.*

► **Lemma 6.** *Let  $\mathcal{A}$  be an arrangement of lines and  $P$  be a simple polygon that does not contain any crossing point of  $\mathcal{A}$ . Then the lines of  $\mathcal{A}$  partition the interior of  $P$  into regions in such a way that the graph of regions and their adjacencies forms a tree.*

**Proof.** To show that the graph of regions and adjacencies is connected, consider any two regions  $R_i$  and  $R_j$ , and choose a curve  $C$  within the interior of  $P$  connecting any point in  $R_i$  to any point in  $R_j$ . Then the sequence of regions crossed by  $C$  forms a walk in the region adjacency graph connecting  $R_i$  to  $R_j$ .

To show that the graph of regions and adjacencies has no simple cycle, assume for a contradiction that there is such a cycle. Then by choosing a representative point within each region of the cycle, and connecting these points by curves that pass between adjacent regions without crossing any other regions, we can form a simple closed curve  $C$  in the plane that crosses the lines of  $\mathcal{A}$  in exactly the order given by the cycle. By the Jordan curve theorem, each line that crosses into the interior of  $C$  must cross out of  $C$  at another point. Two crossings of  $C$  by the same line cannot be adjacent in the cyclic order of crossings, for then the graph cycle corresponding to  $C$  would not be simple. Therefore,  $C$  is crossed by at least two lines, in alternating order. But this can happen only when  $C$  contains the crossing point of these lines, an impossibility as  $C$  is entirely contained in  $P$  which we assumed to enclose no crossings. This contradiction shows that a simple cycle does not exist.

As a connected graph with no simple cycles, the graph of regions and adjacencies must be a tree. ◀

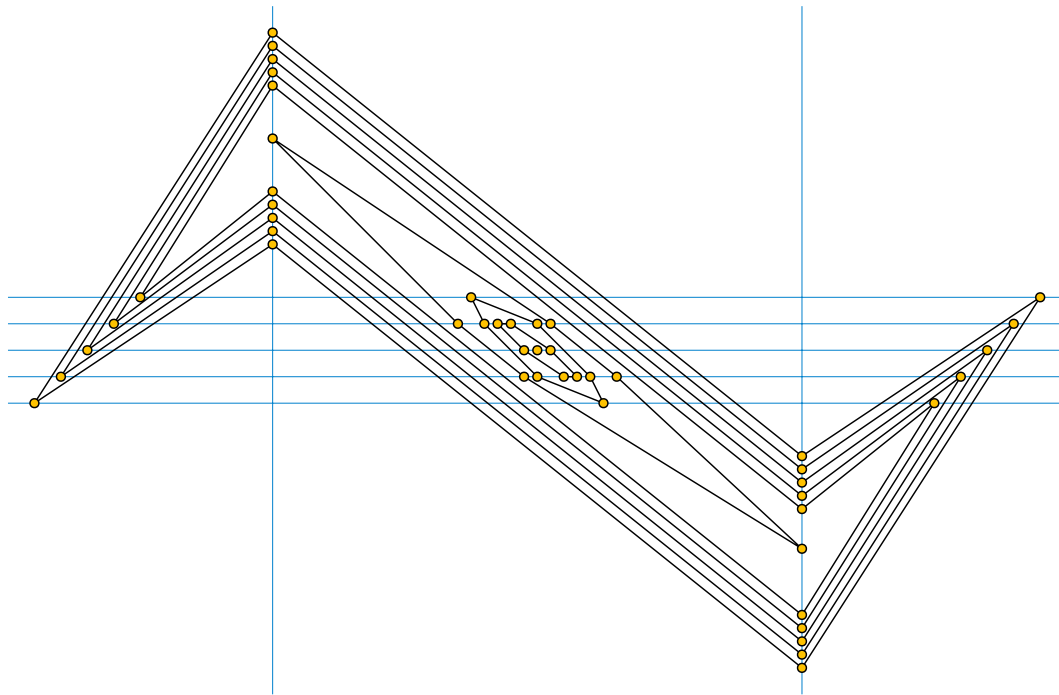
► **Lemma 7.** *Let  $\mathcal{A}$  be an arrangement of lines and  $P$  be a simple polygon that does not contain any crossing point of  $\mathcal{A}$ . Let  $\mathcal{S}$  be the system of disjoint open line segments formed by intersecting the lines of  $\mathcal{A}$  with the interior of  $P$ , and let  $Q$  be another simple polygon, disjoint from  $P$ , such that each vertex of  $Q$  lies on a segment of  $\mathcal{S}$ . Then at least two segments of  $\mathcal{S}$  are disjoint from the interior of  $Q$ .*

**Proof.** Because the graph of regions and adjacencies formed in  $P$  by  $\mathcal{A}$  is a tree (Lemma 6), it has at least two leaves. Let  $s$  be either of the two segments of  $\mathcal{S}$  separating one of these leaf regions from the rest of  $P$ . Then no edge of  $Q$  can enter the interior of this leaf region, because there is no other segment available to be the endpoint of this edge. Therefore,  $Q$  remains entirely on one side of  $s$ , and  $s$  is disjoint from the interior of  $Q$ . As there were at least two choices for  $s$ , there are at least two segments of  $\mathcal{S}$  that are disjoint from the interior of  $Q$ . ◀

Putting these observations together, we have:

► **Lemma 8.** *Let  $\mathcal{A}$  be an arrangement of  $\ell$  lines, let  $p$  and  $r$  be positive integers, and suppose that  $2(r - 1) > \ell \cdot \lfloor p/2 \rfloor$ . Then it is not possible to draw a  $(p, r)$ -nest in such a way that the polygon vertices of the nest and its egg all lie on lines of  $\mathcal{A}$ .*

**Proof.** Suppose for a contradiction that we have drawn a  $(p, r)$ -nest with all points on lines of  $\mathcal{A}$ . Let  $\mathcal{S}$  be the system of disjoint open line segments formed by intersecting the lines of  $\mathcal{A}$  with the outer polygon of the nest. Then  $|\mathcal{S}| \leq \ell \cdot \lfloor p/2 \rfloor$  by Corollary 5, and each of the  $r - 1$  remaining polygons of the nest use up at least two of the segments of  $\mathcal{S}$  by Lemma 7. Therefore, if  $2(r - 1) > \ell \cdot \lfloor p/2 \rfloor$  (as we supposed in the statement of the lemma), there will be no segments remaining for the egg to lie on. Therefore, a drawing meeting these conditions is impossible. ◀



■ **Figure 4** An arrangement of  $\ell$  lines can support  $\lfloor 3(\ell - 1)/2 - 1 \rfloor$  nested hexagons surrounding a central point, with the point and the hexagon vertices all on the lines, and all arrangement crossings exterior to all hexagons.

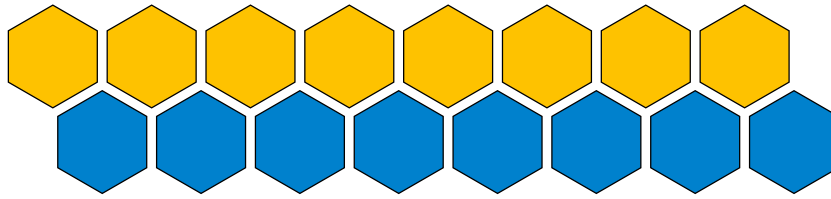
In the particular case of a  $(6, r)$ -nest (as used in Figure 3), Lemma 8 states that a drawing that does not contain an arrangement crossing cannot exist for  $r \geq 3\ell/2 + 2$ . This is close to tight: Figure 4 shows how to draw a  $(6, r)$ -nest with all polygon vertices and the egg on  $\ell$  lines, for  $r = 3\ell/2 - O(1)$ .

### 2.3 The main result

**Proof of Theorem 1.** We wish to show the existence of a planar cubic bipartite graph that cannot be drawn on  $\ell$  lines, for a given parameter  $\ell$ . Consider a cubic bipartite 3-vertex-connected planar graph formed, as in Figure 3, by at least  $\binom{\ell}{2} + 2$  subunits, each of which must be drawn as a  $(6, r)$ -nest, for  $r = \lceil 3\ell/2 \rceil + 2$ . Because there are  $O(\ell^2)$  subunits, each of size  $O(\ell)$ , and  $O(\ell)$  vertices surrounding the subunits, the total size of the resulting graph is  $O(\ell^3)$ .

To argue that the resulting graph cannot be drawn on  $\ell$  lines, we consider an arbitrary arrangement  $\mathcal{A}$  with  $\ell$  lines, and prove that the graph cannot be drawn with all of its vertices on  $\mathcal{A}$ . Among the graph's  $\binom{\ell}{2} + 2$  subunits, one subunit (the one from which the outer face was chosen) can surround all the others, but the rest must be drawn in disjoint regions of the plane. Because there are more remaining subunits than the number of crossing points of  $\mathcal{A}$  (which is at most  $\binom{\ell}{2}$ ), at least one subunit must be drawn in such a way that it does not contain any of the crossing points of  $\mathcal{A}$ . However, by Lemma 8, this is impossible. ◀

When the subunits of the graph are arranged as in Figure 3, into a compact hexagonal grid in the plane, then (as the figure shows) the vertices of the whole graph can be covered by  $O(\ell)$  lines, even though we have proved that there is no cover by  $\ell$  lines. Alternatively,



■ **Figure 5** Zigzag pattern of subunits (shown schematically as hexagons) used to construct a graph of pathwidth  $O(1)$  that cannot be drawn on few lines.

it is possible to arrange the subunits into a linear zig-zag pattern (Figure 5), preserving the 3-vertex-connectedness of the resulting graph and still requiring only  $O(\ell^2)$  vertices to surround the subunits. When the subunits are arranged in this way, the resulting graph might require a larger number of lines to cover its vertices (i.e., we do not have tight bounds on the number of lines needed for a graph of this form), but it has pathwidth  $O(1)$ .

### 3 Series-parallel graphs

A *two-terminal series-parallel graph* (series-parallel graph, for short) is a graph with two distinct designated *terminal* vertices  $s$  and  $t$  formed recursively from smaller graphs of the same type (starting from a single edge) by two operations:

- Series composition: given two series-parallel graphs  $G_1$  and  $G_2$  with terminals  $s_1, t_1, s_2$ , and  $t_2$ , form their disjoint union, and then merge vertices  $s_2$  and  $t_1$  into a single vertex. Let the terminals of the resulting merged graph be the unmerged terminals of the given graphs,  $s_1$  and  $t_2$ . Series composition forms an associative binary operation on these graphs (if we perform series compositions on a sequence of more than two graphs, the order in which we perform the compositions does not affect the result).
- Parallel composition: given two series-parallel graphs  $G_1$  and  $G_2$  with terminals  $s_1, t_1, s_2$ , and  $t_2$ , form their disjoint union, merge vertices  $s_1$  and  $s_2$  into a single vertex, and similarly merge  $t_1$  and  $t_2$  into a single vertex. Let the terminals of the resulting merged graph be the resulting merged vertices. Parallel composition forms an associative and commutative binary operation on these graphs (if we perform series compositions on a set of more than two graphs, neither the order of the two graphs in each composition nor the order in which we perform the compositions affects the result).

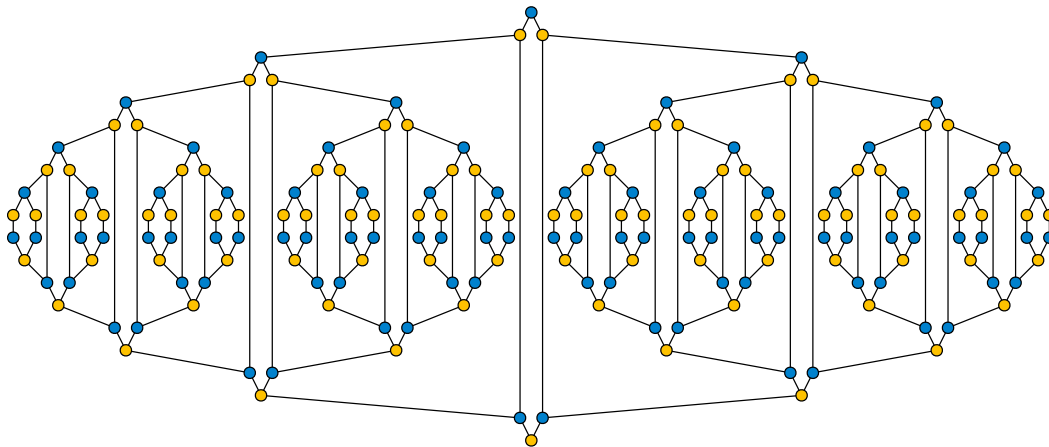
These graphs have treewidth two, and every graph of treewidth two is a subgraph of a series-parallel graph. They are automatically planar, and they include every outerplanar graph.

We will recursively construct two families of series-parallel graphs  $A_i$  and  $B_i$  that cannot be drawn on a bounded number of lines. Figure 6 shows the graph  $B_5$  from this family. To construct these graphs, let  $A_1$  be a series-parallel graph with one edge and two terminal vertices. Then:

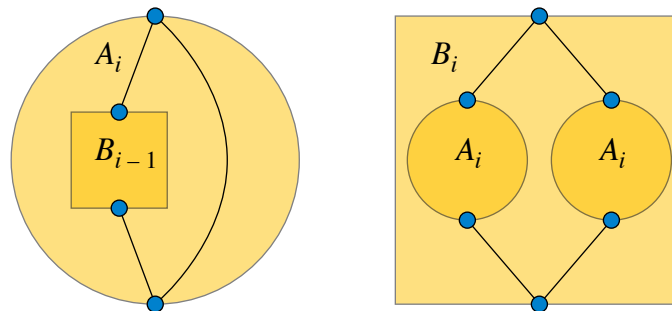
- For each  $i \geq 1$ , let  $B_i$  be the graph formed as the parallel composition of two subgraphs, each of which is the series composition of an edge,  $A_i$ , and another edge (Figure 7, right).
- For each  $i > 1$ , let  $A_i$  be the graph formed as the parallel composition of two subgraphs, one of which is a single edge and the other of which is the series composition of an edge,  $B_{i-1}$ , and another edge (Figure 7, left).

It follows by induction that these graphs are subcubic, with degree two at their two terminals, and that they are bipartite, with a 2-coloring (shown in the figure) in which the two terminals have different colors. In the figure, the upper blue and lower yellow vertices are the terminals of a graph  $B_i$  at some level of the construction, while the upper yellow and lower blue vertices are the terminals of a graph  $A_i$  at some level of the construction.





■ **Figure 6** A subcubic bipartite series-parallel graph that cannot be drawn on few lines.



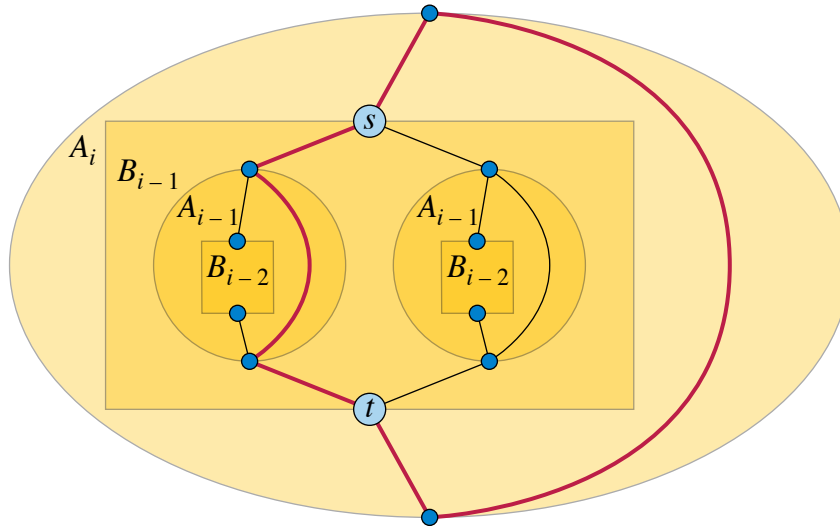
■ **Figure 7** Recursive construction of  $A_i$  (left) and  $B_i$  (right).

Because they are 2-vertex-connected but not 3-vertex-connected, these graphs have many planar embeddings. The planar embeddings of any 2-connected graph may be understood in terms of its SPQR tree [18], which in the case of a series-parallel graph is more or less the same as the expression tree of series and parallel compositions from which it was formed (associating consecutive compositions of the same type into a single multi-operand operation). Because of this equivalence, the embeddings of the graphs  $A_i$  and  $B_i$  may be generated from the embedding shown in the figure by two types of change: any collection of subgraphs connecting two opposite terminals of the graph may be flipped, giving a mirror-image embedding of that subgraph within the larger graph, and any face of the resulting embedding may be chosen as the outer face. We will show that these embeddings always contain large nested sets of hexagons; this property forms the basis for our argument about drawings on few lines.

► **Lemma 9.** *Every planar embedding of  $A_i$  in which the two terminals belong to the outer face contains a  $(6, i - 1)$ -nest.*

**Proof.** For  $i = 1$ , a  $(6, i - 1)$ -nest consists of a single point (the egg), and the result follows trivially. Otherwise, let  $s$  and  $t$  be the two terminals of the  $B_{i-1}$  subgraph from which the given  $A_i$  graph is formed (Figure 8). Then, in the  $A_i$  graph, there are three length-three paths from  $s$  to  $t$ : one through the two terminals of the  $A_i$  graph, and one through each of the two  $A_{i-1}$  subgraphs from which the  $B_{i-1}$  subgraph is formed.





■ **Figure 8** Illustration for Lemma 9. No matter how  $A_i$  is embedded, it will contain a 6-cycle (red) surrounding one of the copies of  $A_{i-1}$  from which it is formed.

By the assumption that the two terminals of  $A_i$  belong to the outer face, there is a six-vertex cycle combining two of these three length-three paths, one through the two terminals of  $A_i$  and one through one of the copies of  $A_{i-1}$ , that surrounds the other copy of  $A_{i-1}$ . By induction, this surrounded copy contains a  $(6, i - 2)$ -nest, which together with the cycle that surrounds it forms a  $(6, i - 1)$ -nest. ◀

► **Lemma 10.** *Let  $j \leq i$  be two positive integers. Then every planar embedding of  $B_i$  contains at least  $2^j - 1$  disjointly-embedded  $(6, i - j)$ -nests.*

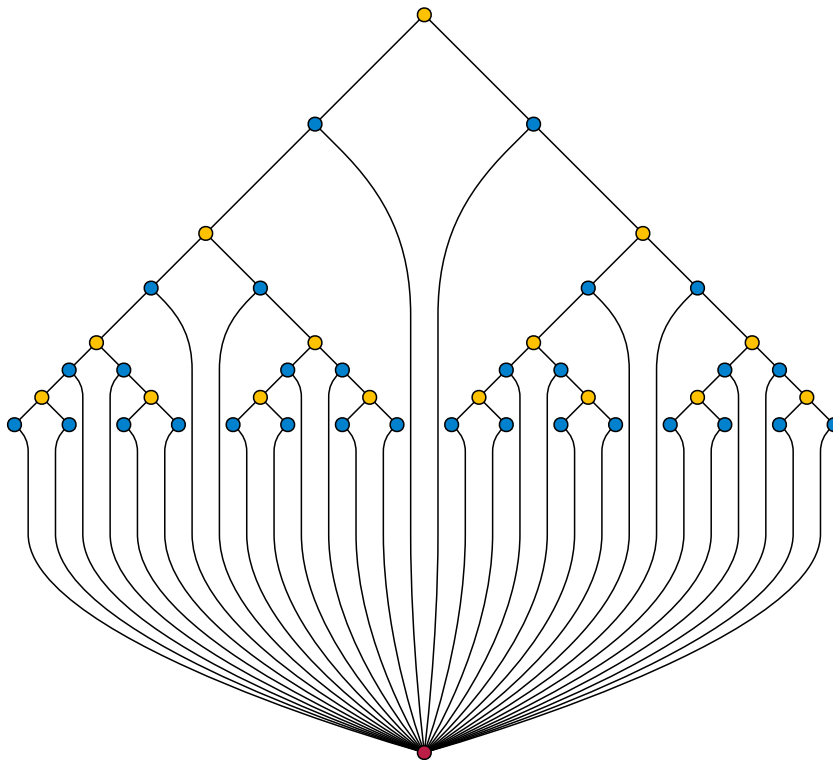
**Proof.**  $B_j$  is recursively constructed from  $2^j$  copies of  $A_{i-j+1}$ . At most one of these copies can contain the outer face of the embedding, so at least  $2^j - 1$  copies are embedded with their two terminals outermost. The result follows from Lemma 9. ◀

**Proof of Theorem 2, series-parallel case.** The theorem claims that there exists a cubic bipartite series-parallel graph that cannot be drawn on  $\ell$  lines. To prove this, choose  $j$  such that  $2^j \geq \binom{\ell}{2} + 2$  and choose  $i$  such that  $i - j \geq 3\ell/2 + 2$ . We claim that, for this case,  $B_i$  has the required properties. By Lemma 10, it contains at least  $\binom{\ell}{2} + 1$  disjointly embedded  $(6, i - j)$ -nests, enough to ensure that at least one of them does not contain any crossing points of any given arrangement of  $\ell$  lines. By Lemma 8, a nest of this depth that does not contain any crossing points cannot be drawn with its vertices on  $\ell$  lines. ◀

## 4 Apex-tree graphs

### 4.1 Apex-tree graphs requiring many lines

Figure 9 depicts a graph in the form of a tree (blue and yellow vertices) plus one additional vertex (red); such a graph has been called an *apex-tree*, and the additional vertex is the *apex*. The tree in the figure can be constructed from the series-parallel graph  $B_4$  of the previous section by contracting half of the vertices (one vertex from each pair of terminals) into a single supervertex. Alternatively, it can be constructed from a complete binary tree by subdividing every non-leaf edge and then connecting each subdivision vertex and each leaf vertex to the apex. These graphs are subcubic except at the apex, and bipartite.



■ **Figure 9** An apex-tree graph, subcubic except at the apex, that cannot be drawn on few lines.

As with the earlier series-parallel graphs, we will prove that these graphs cannot be drawn on a sublogarithmic number of lines. An obstacle to the proof, however, is that they contain no  $(p, r)$ -nests for  $r > 1$ , nor can any such nest exist in any apex-tree. The reason is that, in an apex-tree, all cycles contain the apex. Therefore, there can be no two disjoint cycles, and no nests of two or more disjoint cycles. Nevertheless, these graphs do contain nest-like structures. We define a  $(p, r)$ -near nest in an embedded plane graph to be a collection of  $p$   $r$ -cycles, plus one additional vertex (the egg), such that all cycles contain the egg in their interior, the cycles are edge-disjoint, and any two of them share at most one vertex with each other. Then the following lemma is an analogue of Lemma 8 for near-nests:

► **Lemma 11.** *Let  $\mathcal{A}$  be an arrangement of  $\ell$  lines, let  $p$  and  $r$  be positive integers, and suppose that  $r - 1 > \ell \cdot \lfloor p/2 \rfloor$ . Then it is not possible to draw a  $(p, r)$ -near-nest in such a way that the polygon vertices of the nest and its egg all lie on lines of  $\mathcal{A}$ .*

**Proof.** Suppose for a contradiction that we have drawn a  $(p, r)$ -near-nest with all points on lines of  $\mathcal{A}$ . Let  $\mathcal{S}$  be the system of disjoint open line segments formed by intersecting the lines of  $\mathcal{A}$  with the outer polygon of the nest. Then  $|\mathcal{S}| \leq \ell \cdot \lfloor p/2 \rfloor$  by Corollary 5, and each of the  $r - 1$  remaining polygons of the nest use up at least one of the segments of  $\mathcal{S}$  by Lemma 7 and by the fact that at most one of the two extreme segments of the polygon can be shared with other polygons interior to it. Therefore, if  $r - 1 > \ell \cdot \lfloor p/2 \rfloor$  (as we supposed in the statement of the lemma), there will be no segments remaining for the egg to lie on. Therefore, a drawing meeting these conditions is impossible. ◀

Analogously to Lemma 9 and Lemma 10, we have:

► **Lemma 12.** *For every planar embedding of a graph like the one of Figure 9 formed from a complete binary tree of height  $i$ , and for every  $j \leq i$ , the embedding contains  $2^j - 1$  disjoint  $(4, i - j)$ -near-nests.*

**Proof.** This result follows immediately from Lemma 10, which proves the existence of a  $(6, i - j)$ -nest in the corresponding series-parallel graphs, together with the observations that every planar embedding of our apex-trees can be expanded to a planar embedding of the corresponding series-parallel graphs, and that every 6-cycle of a  $(6, i - j)$ -nest in the expanded series-parallel graph has three of its vertices contracted into the apex of the apex-tree graph.

Alternatively, one could prove the result by repeating the proof of Lemma 10 with minor modifications. ◀

**Proof of Theorem 2, apex-tree case.** The theorem claims that there exists a bipartite apex-tree graph, subcubic except at its apex, that cannot be drawn on  $\ell$  lines. To prove this, choose  $j$  such that  $2^j \geq \binom{\ell}{2} + 2$  and choose  $i$  such that  $i - j - 1 > 2\ell$ . Form an apex-tree graph as above from a complete binary tree of height  $i$ . We claim that, for this case, the resulting apex-tree graph has the required properties. For, by Lemma 12, it contains at least  $\binom{\ell}{2} + 1$  disjointly embedded  $(4, i - j)$ -near-nests, enough to ensure that at least one of them does not contain any crossing points of any given arrangement of  $\ell$ -lines. By Lemma 11, a nest of this depth that does not contain any crossing points of the  $\ell$  lines cannot be drawn with its vertices on the lines. ◀

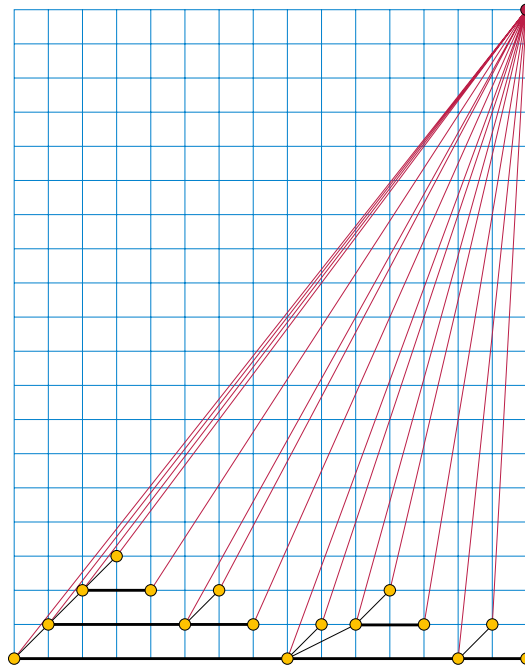
## 4.2 Drawing apex-tree graphs on few lines

Recall that Theorem 3 states that we can draw any apex-tree graph planarly on  $O(\log n)$  parallel lines. To do so, we adapt a standard tool from tree drawing, the *heavy path decomposition* [20], to draw any tree with its vertices on the points of a grid of height  $\log_2 n$  and width  $n$  (in particular, on  $O(\log n)$  horizontal lines) in such a way that the resulting drawing can be extended to a drawing of an apex-tree, by adding one more vertex adjacent to any subset of the tree vertices.

The heavy path decomposition of a tree is obtained by choosing one *heavy edge* for each non-leaf vertex of the tree, an edge connecting it to the subtree with the largest number of vertices (breaking ties arbitrarily). The connected components of the subgraph formed by the heavy edges are *heavy paths*, including as a special case length-zero paths for leaf vertices that were not chosen by their parent. The heavy paths partition the vertices of the tree. By induction, a vertex  $v$  that can reach a leaf by a path that includes  $i$  non-heavy edges must be the root of a subtree containing at least  $2^{i+1} - 1$  vertices (including  $v$  itself). Therefore, in a tree with  $n > 1$  vertices, every root-to-leaf path contains at most  $\log_2 n - 1$  non-heavy edges.

**Proof of Theorem 3.** To draw the given tree on a grid, we traverse the tree in preorder, ordering the children at each vertex so that the heavy edge is last. We let the  $x$ -coordinate of each vertex be its position in this preorder listing, and we let the  $y$ -coordinate be the number of non-heavy edges on the path from the vertex to the root. These choices give unique coordinates for each vertex on a grid of height  $\log_2 n$  and width  $n$ , as claimed. Each tree edge either connects two consecutive vertices on the same level of the grid (on the same heavy path), or it connects vertices on consecutive levels (a parent and child not connected by a heavy edge) whose  $x$ -coordinates both are less than the next vertex on the same level as the parent. Therefore, the drawing has no crossings.

All edges of this tree drawing have slope in the interval  $[0, 1]$ . The traversal ordering ensures that, for each vertex  $v$ , and each vertex  $w$  with a higher  $y$ -coordinate than  $v$ , one of the following is true:  $w$  has smaller  $x$ -coordinate than  $v$ ,  $w$  is a descendant of  $v$ , or  $w$  is a



■ **Figure 10** Drawing an apex-tree on a grid. The thick horizontal black lines depict the heavy path decomposition of the given tree. Note that although the grid size is approximately  $(n + \log_2 n) \times n$ , only the bottom  $\log_2 n$  horizontal grid lines and the top horizontal grid line are occupied by vertices.

descendant of a vertex that is placed below and to the right of  $v$ . In all three cases, neither  $w$  nor any edge incident to  $w$  can block the visibility from  $v$  upwards and to the right through lines of slope greater than one. Therefore, if we place the apex  $n + 1$  units above the upper right corner of the grid, it will be visible to all tree vertices by unobstructed lines of sight and we can complete the drawing of any apex-tree consisting of the given tree and one apex. ◀

The construction is depicted in Figure 10.

## 5 Conclusions and open problems

We have found planar 3-regular bipartite graphs of size cubic in  $\ell$  that cannot be drawn on  $\ell$  lines, cubic bipartite series-parallel graphs of size exponential in  $\ell$  that cannot be drawn on  $\ell$  lines, and apex-trees of size exponential in  $\ell$  that cannot be drawn on  $\ell$  lines. For apex-trees the exponential size bound is necessary, although there may still be room for tightening the gap between the exponential upper and lower bounds. For the other two classes of graphs, we do not know whether our results are tight. Stefan Felsner and Alexander Wolff have recently proven that every 4-vertex-connected maximal planar graph of size at most quadratic in  $\ell$  may be drawn on  $\ell$  pseudolines, and that it is NP-hard to find drawings on two lines [3]. Do there exist planar graphs of subcubic size that cannot be drawn on  $\ell$  lines? Do there exist series-parallel graphs of polynomial size that cannot be drawn on  $\ell$  lines? How well can the optimal number of lines be approximated? We leave these problems as open for future research.

## References

- 1 Hans-Jürgen Bandelt, Victor Chepoi, and David Eppstein. Combinatorics and geometry of finite and infinite squaregraphs. *SIAM Journal on Discrete Mathematics*, 24(4):1399–1440, 2010. doi:10.1137/090760301.
- 2 Michael J. Bannister, William E. Devanny, Vida Dujmović, David Eppstein, and David R. Wood. Track layouts, layered path decompositions, and leveled planarity. *Algorithmica*, 2018. doi:10.1007/s00453-018-0487-5.
- 3 Therese Biedl, William Evans, Stefan Felsner, Sylvain Lazard, Henk Meijer, Pavel Valtr, Sue Whitesides, Steve Wismath, and Alexander Wolff. Line and plane cover numbers revisited. Manuscript, 2019.
- 4 Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Alexander Ravsky, Oleg Verbitsky, and Alexander Wolff. The complexity of drawing graphs on few lines and few planes. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures: 15th International Symposium, WADS 2017, St. John's, NL, Canada, July 31 – August 2, 2017, Proceedings*, volume 10389 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 2017. doi:10.1007/978-3-319-62127-2\_23.
- 5 Steven Chaplick, Krzysztof Fleszar, Fabian Lipp, Oleg Verbitsky, and Alexander Wolff. Drawing graphs on few lines and few planes. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization: 24th International Symposium, GD 2016, Athens, Greece, September 19–21, 2016, Revised Selected Papers*, volume 9801 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2016. doi:10.1007/978-3-319-50106-2\_14.
- 6 Danny Dolev, Tom Leighton, and Howard Trickey. Planar embedding of planar graphs. *Advances in Computing Research*, 2:147–161, 1984. URL: <https://noodle.cs.huji.ac.il/~dolev/pubs/planar-embed.pdf>.
- 7 Vida Dujmović, David Eppstein, Gwenaël Joret, Pat Morin, and David R. Wood. Minor-closed graph classes with bounded layered pathwidth. in preparation, 2018.
- 8 Vida Dujmović, Michael R. Fellows, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances Rosamond, Sue Whitesides, and David R. Wood. On the parameterized complexity of layered graph drawing. *Algorithmica*, 52(2):267–292, 2008. doi:10.1007/s00453-007-9151-1.
- 9 Vida Dujmović, Pat Morin, and David R. Wood. Path-width and three-dimensional straight-line grid drawings of graphs. In Michael T. Goodrich and Stephen G. Kobourov, editors, *Graph Drawing: 10th International Symposium, GD 2002, Irvine, CA, USA, August 26–28, 2002, Revised Papers*, volume 2528 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2002. doi:10.1007/3-540-36151-0\_5.
- 10 Vida Dujmović, Attila Pór, and David R. Wood. Track layouts of graphs. *Discrete Math. Theor. Comput. Sci.*, 6(2):497–521, 2004.
- 11 Vida Dujmović and David R. Wood. Stacks, queues and tracks: layouts of graph subdivisions. *Discrete Math. Theor. Comput. Sci.*, 7(1):155–201, 2005.
- 12 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.
- 13 Peter Eades and Nicholas C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994. doi:10.1007/BF01187020.
- 14 David Eppstein. *Forbidden Configurations in Discrete Geometry*. Cambridge University Press, 2018. Theorem 16.13.
- 15 Stefan Felsner, Giuseppe Liotta, and Stephen Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *Journal of Graph Algorithms and Applications*, 7(4):363–398, 2003. doi:10.7155/jgaa.00075.
- 16 Oksana Firman, Fabian Lipp, Laura Straube, and Alexander Wolff. Examining weak line covers with two lines in the plane (poster abstract). In *Graph Drawing and Network Visualization: 26th International Symposium, GD 2018, Barcelona, Spain, September 26–28, 2018, Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2018. to appear.

- 17 Fabrizio Frati and Maurizio Patrignani. A note on minimum-area straight-line drawings of planar graphs. In *Graph Drawing: 15th International Symposium, GD 2007, Sydney, Australia, September 24-26, 2007, Revised Papers*, volume 4875 of *Lecture Notes in Computer Science*, pages 339–344. Springer, 2008. doi:10.1007/978-3-540-77537-9\_33.
- 18 Saunders Mac Lane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3(3):460–472, 1937. doi:10.1215/S0012-7094-37-00336-3.
- 19 Alexander Ravsky and Oleg Verbitsky. On collinear sets in straight-line drawings. In Petr Kolman and Jan Kratochvíl, editors, *Graph-Theoretic Concepts in Computer Science: 37th International Workshop, WG 2011, Teplá Monastery, Czech Republic, June 21–24, 2011, Revised Papers*, volume 6986 of *Lecture Notes in Computer Science*, pages 295–306. Springer, 2011. doi:10.1007/978-3-642-25870-1\_27.
- 20 Daniel D. Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.



# Counting Polygon Triangulations is Hard

David Eppstein

Computer Science Department, University of California, Irvine, USA  
eppstein@uci.edu

---

## Abstract

We prove that it is  $\#P$ -complete to count the triangulations of a (non-simple) polygon.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Problems, reductions and completeness

**Keywords and phrases** counting complexity,  $\#P$ -completeness, triangulation, polygons

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.33

**Funding** *David Eppstein*: Supported in part by NSF grants CCF-1618301 and CCF-1616248.

## 1 Introduction

In 1979, Leslie Valiant published his proof that it is  $\#P$ -complete to compute the permanent of a 0-1 matrix, or equivalently to count the perfect matchings of a bipartite graph [41]. This result was significant in two ways: It was surprising at the time that easy (polynomial time) existence problems could lead to intractable counting problems, and it opened the door to hardness proofs for many other counting problems. Beyond individual problems, several broad classifications of hard graph counting problems are now known. For instance, it is  $\#P$ -complete to count  $k$ -colorings or determine the value of the Tutte polynomial at any value outside of a small finite set of exceptions [23], and all problems of counting homomorphisms to a fixed directed acyclic graph are either  $\#P$ -complete or polynomial [18].

In computational geometry, it has been shown that counting the vertices or facets of high-dimensional convex polytopes is  $\#P$ -complete [28], and that computing the expected total length of the minimum spanning tree of a stochastic subset of three-dimensional points is  $\#P$ -hard [25]. Additionally, when testing existence of a geometric structure is hard [5, 9, 31, 29, 26] it is just hard to determine whether its count is nonzero. However, we know of no past hardness proofs for counting easy-to-construct two-dimensional structures. Although there has been significant research on counting non-crossing configurations in the plane, including matchings, simple polygons, spanning trees, triangulations, and pseudotriangulations, the complexity of these problems has remained undetermined. Research on these problems has instead focused on determining the number of configurations for special classes of point sets [20, 8, 24], bounding the number of configurations as a function of the number of points [2, 4, 39, 37, 17, 38, 1, 33, 3, 36, 21, 10], developing exponential- or subexponential-time counting algorithms [12, 7, 42, 5, 30, 13], or finding faster approximations [6, 27].

In this paper we bring the two worlds of  $\#P$ -completeness and counting planar structures together by proving the following theorem:

► **Theorem 1.** *It is  $\#P$ -complete to count the number of triangulations of a given polygon.*

Necessarily, the polygons of our hardness construction have holes, as it is straightforward to count the triangulations of a simple polygon in polynomial time by dynamic programming [19, 32, 16]. The polygons resulting from the construction can be drawn with all vertices on a grid of polynomial size. Our proof strategy is to develop a polynomial-time counting reduction from the problem of counting independent sets in 3-regular bipartite planar graphs (here the independent sets are not necessarily maximum nor maximal), which was proved

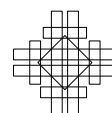


© David Eppstein;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 33; pp. 33:1–33:17  
Leibniz International Proceedings in Informatics



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





#P-complete by Xia, Zhang, and Zhao [43]. We reduce counting independent sets in 3-regular planar graphs to counting maximum non-crossing subsets of a special class of line segment arrangements, which we in turn reduce to counting triangulations.

## 2 Preliminaries

### 2.1 Polygons and triangulation

A *planar straight-line graph* consists of finitely many closed line segments in the Euclidean plane, disjoint except for shared endpoints. The endpoints of these segments can be interpreted as the vertices, and the segments as the edges, of an undirected graph drawn with straight edges and no crossings in the plane. The *faces* of the planar straight-line graph are the connected components of its complement (that is, maximal connected subsets of the plane that are disjoint from the segments of the graph). In any planar straight-line graph, exactly one unbounded face extends beyond the bounding box of the segments; all other faces are bounded. A segment of the graph forms a *side* of a face if the interior of the segment intersects the topological closure of the face. As with any graph, a planar straight-line graph is  $d$ -regular if each of its vertices is incident to exactly  $d$  line segments.

For the purposes of this paper, we define a *polygon*  $P$  to be a 2-regular planar straight-line graph in which there is a bounded face  $\phi$  whose sides are all the segments of  $P$ . If  $\phi$  exists, it is uniquely determined from  $P$ . We call  $\phi$  the *interior* of  $P$ . The connected components of the graph are necessarily simple cycles of line segments, exactly one of which separates  $\phi$  from the unbounded face. If there is more than one connected component of the graph, we call the other components *holes*, and if there are no holes we call  $P$  a *simple polygon*.

We define a *triangulation* of a polygon  $P$  to be a planar straight-line graph consisting of the edges of  $P$  and added segments interior to  $P$ , all of whose vertices are vertices of  $P$ , partitioning the interior of  $P$  into three-sided faces. As is well known, every polygon has a triangulation. A triangulation of an  $n$ -vertex polygon can be found in  $O(n \log n)$  time, for instance by constrained Delaunay triangulation, and this can be improved to  $O(n \log h)$  for polygons with  $h$  holes [11]. The known exponential or sub-exponential algorithms for counting triangulations of point sets [7, 30] can be adapted to count triangulations of polygons in the same time bounds.

### 2.2 Counting complexity

The complexity class #P is defined as the class of functional algorithmic problems for which the desired output counts the accepting paths of some nondeterministic polynomial-time Turing machine.

► **Lemma 2.** *Computing the number of triangulations of a polygon is in #P.*

**Proof.** The output is the number of accepting paths of a nondeterministic polynomial-time Turing machine that guesses the set of edges in the triangulation, and verifies that these edges form a triangulation of the input. ◀

#P-hardness and #P-completeness are defined using reductions, polynomial-time transformations from one problem  $X$  (typically already known to be hard) to another problem  $Y$  that we wish to prove hard. Three types of reduction are in common use for this purpose:

- *Turing reductions* consist of an algorithm for solving problem  $X$  in polynomial time given access to an oracle for solving problem  $Y$ .

- *Polynomial-time counting reductions* consist of two polynomial-time transformations: a transformation  $\sigma$  that transforms inputs to  $X$  into inputs to  $Y$ , and a second transformation  $\tau$  that transforms outputs of  $Y$  back to outputs of  $X$ . The reduction is valid if, for every input  $\chi$  to problem  $X$ ,  $\tau(Y(\sigma(\chi))) = X(\chi)$ .
- *Parsimonious reductions* consist of a polynomial-time transformation from inputs of  $X$  to inputs of  $Y$  that preserve the exact solution value.

A problem  $Y$  is defined to be  $\#P$ -hard for a given class of reductions if every problem  $X$  in  $\#P$  has a reduction to  $Y$ .  $Y$  is  $\#P$ -complete if, in addition,  $Y$  is itself in  $\#P$ . Composing two reductions of the same type produces another reduction, so we will generally prove  $\#P$ -hardness or  $\#P$ -completeness by finding a single reduction from a known-hard problem and composing it with the reductions from everything else in  $\#P$  to that known-hard problem.

The reductions that we construct in this work will be polynomial-time counting reductions. However, we rely on earlier work on  $\#P$ -completeness of graph problems that uses the weaker notion of Turing reductions. Therefore, we will prove that our geometric problems are  $\#P$ -complete under Turing reductions. If the graph-theoretic results are strengthened to use counting reductions (and in particular if counting maximum independent sets in regular planar graphs is  $\#P$ -complete under counting reductions) then the same strengthening will apply as well to counting triangulations. For the remainder of this paper, however, whenever we refer to  $\#P$ -hardness or  $\#P$ -completeness, it will be under Turing reductions.

### 2.3 Counting independent sets

Xia, Zhang, and Zhao [43] proved that it is  $\#P$ -complete to count the vertex covers in a connected 3-regular bipartite planar graph (subsets of vertices that touch all edges). A set of vertices is a vertex cover if and only if its complement is an independent set, so it immediately follows that it is also  $\#P$ -complete to count independent sets in the same graphs (the counts are the same). It is also  $\#P$ -complete to find the number of *maximum* independent sets in a planar bipartite graph of maximum degree three [40], but for our purposes the regularity of the graph is more important than the maximality of the independent set.

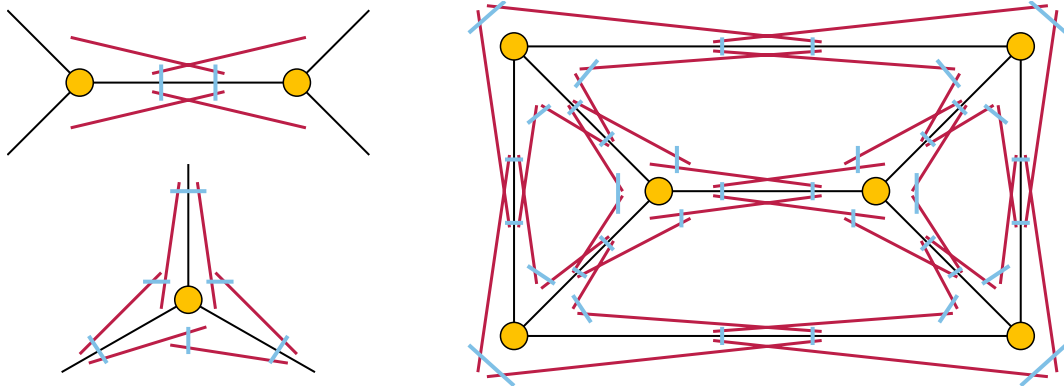
## 3 Red–blue arrangements

In this section we define and prove hard a counting problem that will be an intermediate step in our hardness proof for triangulations. It involves counting maximum non-crossing subsets of certain special line segment arrangements. Counting maximum non-crossing subsets of arbitrary line segment arrangements can easily be shown to be hard: It follows from the hardness of counting maximum independent sets in planar graphs [40] and from the proof of Scheinerman’s conjecture that every planar graph can be represented as an intersection graph of line segments [34, 14]. So the significance of the reduction that we describe in this section is that it provides arrangements with a highly constrained form, a form that will be useful in our eventual reduction to counting triangulations.

► **Definition 3.** *We define a red–blue arrangement to be a collection of finitely many line segments in the plane (specified by the Cartesian coordinates of their endpoints) with the following properties:*

- *Each line segment is assigned a color, either red or blue.*
- *Each intersection point of two segments is a proper crossing point of exactly two segments.*
- *Each blue segment is crossed by exactly two other segments, both red.*
- *Each red segment is crossed by exactly three other segments, in the order blue–red–blue.*
- *The union of the segments forms a connected subset of the plane.*

33:4 Counting Polygon Triangulations is Hard



■ **Figure 1** Transformation from a 3-regular planar graph  $G$  to a red–blue arrangement. Left: the gadgets for edges and vertices of the graph. Right: an example of the whole arrangement for the graph of a triangular prism.

See the right side of Figure 1 for an example. We will be interested in the maximum non-crossing subsets of such an arrangement: sets of as many segments as possible, no two of which cross.

In a red–blue arrangement, consider the graph whose vertices represent line segments and whose edges represent bichromatic crossings (crossings between a red segment and a blue segment). Then this graph is a disjoint union of cycles, each of which has even length with vertices alternating between red and blue. We call the cycles of this graph *alternating cycles* of the arrangement.

► **Lemma 4.** *Every red–blue arrangement has equal numbers of red and blue segments. If there are  $n$  red and  $n$  blue segments, the maximum non-crossing subsets of the arrangement all have exactly  $n$  segments. Within each alternating cycle of the arrangement, a maximum non-crossing subset must use a monochromatic subset of the cycle (either all the red segments of the cycle, or all the blue segments of the cycle).*

**Proof.** The equal numbers of red and blue segments in the whole arrangement follow from the decomposition of the arrangement into alternating cycles and the equal numbers within each cycle. Within a single cycle, there are exactly two maximum non-crossing subsets, the subsets of red and of blue segments, each of which uses exactly half of the segments of the cycle. Therefore, no non-crossing subset of the whole arrangement can include more than  $n$  segments (half of the total), and a non-crossing subset that uses exactly  $n$  segments must be monochromatic within each alternating cycle. There exists at least one non-crossing subset of exactly  $n$  segments, namely the set of blue segments. ◀

Given a connected 3-regular planar graph  $G$ , we will construct a red–blue arrangement  $A_G$  from it, as follows.

We begin by finding a straight-line drawing of  $G$ . We then replace each vertex  $v$  of  $G$  by a twelve-segment alternating cycle (Figure 1, lower left). Each edge of  $G$  incident to  $v$  will have two red segments on either side of it, within the two faces bounded by that edge, and these segments form the six red segments of the alternating cycle. Three blue segments, drawn near  $v$  within the three faces incident to  $v$ , connect pairs of red segments within each face. Another three blue segments cross the three edges incident to  $v$  (which are not themselves part of the arrangement) and connect the two red segments on either side of the edge.

In this way, each edge  $uv$  of  $G$  will have four red segments near it (two on each side from each of its two endpoints) and two blue segments crossing it (one from each endpoint). We arrange these segments so that, on each side of  $uv$ , the two red segments cross, and there are no other crossings except those in the alternating cycles (Figure 1, upper left). An example of the whole red–blue arrangement produced by these rules is shown in Figure 1, right.

By using an initial drawing of  $G$  within an  $n \times n$  unit grid, where  $n$  is the number of vertices of  $G$  [35, 15], and then scaling the grid by a polynomial factor, we can create room to place each segment of the corresponding red–blue arrangement near its corresponding edge or vertex in the drawing of  $G$ , with integer coordinates for its endpoints. In this way, the entire construction can be performed in polynomial time, using segments whose endpoints have integer coordinates of polynomial magnitude. Because the coordinates are small integers, the angles between any two crossing segments will be at least inverse-polynomial. We omit the details.

► **Lemma 5.** *Let  $G$  be a connected 3-regular planar graph, and let  $A_G$  be constructed from  $G$  as above. Then the independent sets of  $G$  correspond one-for-one with the maximum non-crossing subsets of  $A_G$ , with an independent set having size  $k$  if and only if the corresponding non-crossing subset has exactly  $6k$  red segments.*

**Proof.** Given an independent set  $Z$  in  $G$ , one can construct a non-crossing subset of  $A_G$  of size  $n$ , with  $6|Z|$  red segments, by choosing all the red segments in the alternating cycles around members of  $Z$ , and all the blue segments in the remaining alternating cycles. Because  $Z$  includes no two adjacent vertices, the subsets chosen in this way will be non-crossing.  $Z$  can be recovered as the set of vertices from which we used red segments of alternating cycles, so the correspondence from independent sets to non-crossing subset constructed in this way is one-to-one.

By Lemma 4, these non-crossing subsets are maximum. By the same lemma, every maximum non-crossing subset consists of red segments in the alternating cycles of some vertices of  $G$  and blue segments in the remaining alternating cycles. No two adjacent vertices of  $G$  can have their red segments chosen as that would produce a crossing, so there are no maximum non-crossing subsets other than the ones coming from independent sets of  $G$ . ◀

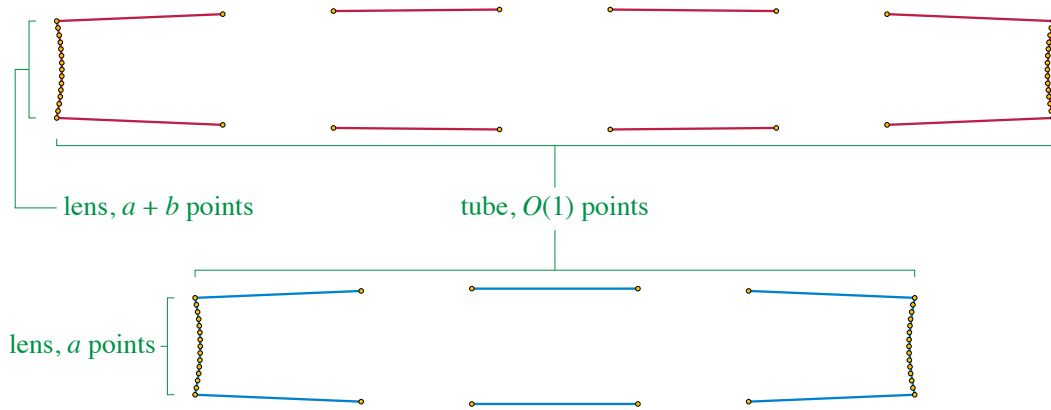
This leads to the main result of this section:

► **Lemma 6.** *It is #P-complete to compute the number of maximum non-crossing subsets of a red–blue arrangement  $A$ .*

**Proof.** The problem is clearly in #P. By Lemma 5, the construction of the red–blue arrangement  $A_G$  from a 3-regular planar graph  $G$  is a parsimonious reduction from the known #P-complete problem of counting independent sets in 3-regular planar graphs. ◀

## 4 Reduction to triangulation

In this section we describe our reduction from line segment arrangements to polygons. The rough idea is to thicken each segment of the given red–blue arrangement to a rectangle with rounded ends, and form the union of these rectangles. In the part of the polygon formed from each segment, many more triangulations will use diagonals running end-to-end along the segment than triangulations that do not, causing most triangulations to correspond to sets of diagonals from a maximum non-crossing subset of the arrangement. With a careful choice of the shapes of the thickened segments in this construction, we can recover the number of maximum non-crossing subsets of the arrangement from the number of triangulations.



■ **Figure 2** Polygonal gadgets for replacing the red segments (top) and blue segments (bottom) of a red–blue arrangement.

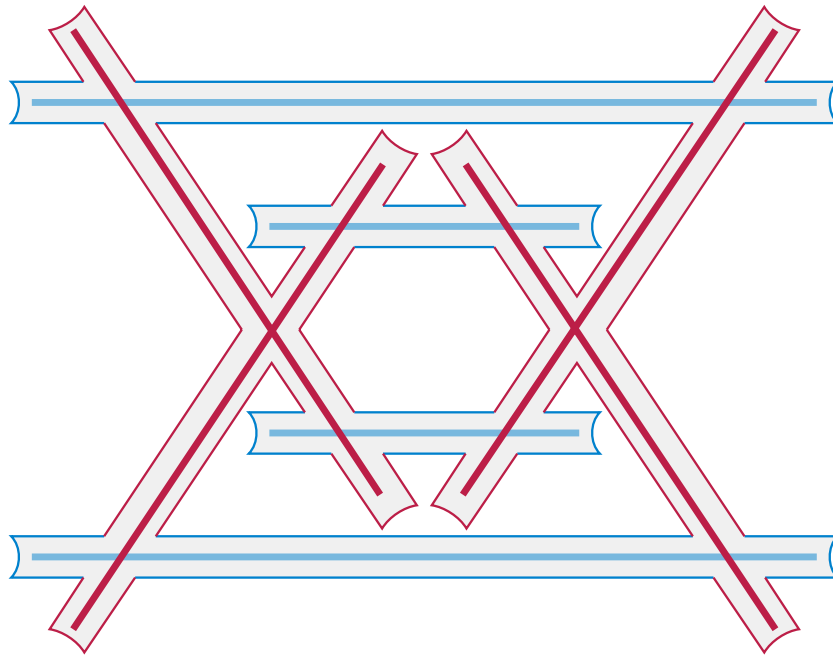
#### 4.1 From segments to polygons

The gadgets that we use to replace the red and blue segments of a red–blue arrangement are shown (not to scale) in Figure 2. They resemble the cross-section of a telescope, and we have named their parts accordingly. Each gadget consists of  $O(1)$  vertices spaced along two convex curves, which we call the tube. These two curves bend slightly outwards from each other, but both remain close to the center line of the gadget. Consecutive pairs of points are connected to each other along the tube, leaving either three gaps (for red segments) or two gaps (for blue segments) where other segments of the arrangement cross the given segment. The two vertices on each side of each gap are shared with the gadget for the crossing segment.

At the ends of the tubes, the two convex curves are connected to each other by two concave curves (the lenses of the gadget), each containing larger numbers of vertices that are connected consecutively to each other without gaps. Each blue lens has some number  $a$  of vertices, and each red lens has some larger number  $a + b$  of vertices, where  $a$  and  $b$  are both positive integers to be determined later as a function of the number of segments in the arrangement. The key geometric properties of these gadgets are:

- Each gadget forms a collection of polygonal chains whose internal vertices (the vertices that are not the end-point of any of the chains) are not part of any other gadget and whose endpoints are part of exactly one other gadget. The union of all the gadgets of the arrangement forms a single polygon (with holes).
- Each vertex of the lens at one end of the gadget is visible within the polygon to each vertex of the lens at the other end of the gadget, and to each vertex of the tube of the gadget, but not to any vertices of its own lens that it is non-adjacent to.
- All of the vertices that are visible to a vertex of the lens of a gadget belong to the same gadget. (This is not an automatic property of the gadgets as drawn, but can be achieved by making the gadgets sufficiently narrow relative to the spacing of points along their tubes, depending on the angles at which different segments cross each other.)
- Each pair of points that are visible to each other are both part of one gadget. (Again, this is not an automatic property, but can be achieved by using narrow-enough gadgets.)

We denote by  $P_A$  the polygon obtained by replacing each segment of a red–blue arrangement  $A$  by a gadget. Figure 3 gives an example of a red–blue arrangement  $A$  and its corresponding polygon  $P_A$ , drawn approximately and schematically (not to scale).



■ **Figure 3** A red–blue arrangement  $A$  and the polygon  $P_A$  obtained from it by replacing its segments by gadgets (schematic view, not to scale).

As long as the segments of  $A$  have endpoints whose coordinates are integers of polynomial magnitude, and the parameters  $a$  and  $b$  are chosen to also be of polynomial magnitude, it will be possible to carry out the construction of  $P_A$  from  $A$  in polynomial time and for the coordinates of the resulting polygon to be integers of (larger) polynomial magnitude. We omit the details.

## 4.2 Triangulation within a gadget

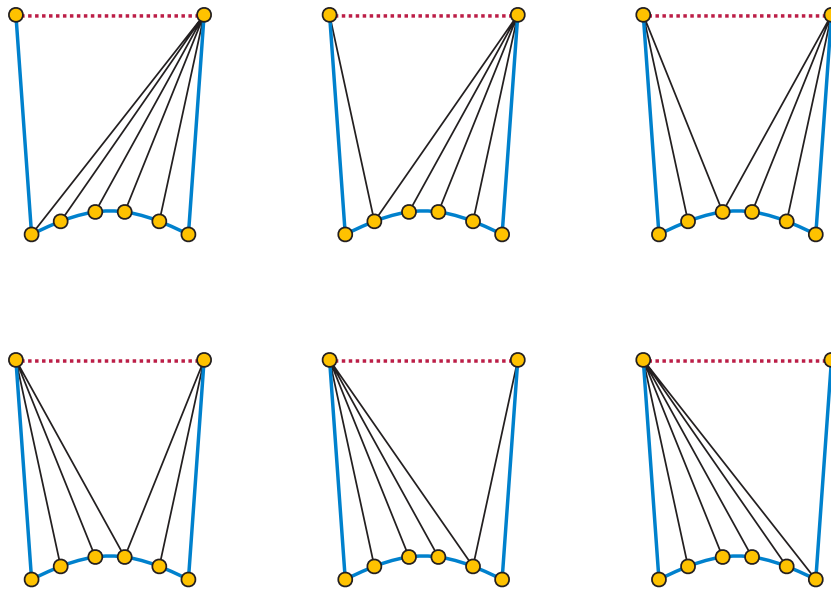
For a triangulation  $T$  of a polygon  $P_A$ , and a gadget  $S$  of  $P_A$  corresponding to a segment  $s$  of  $A$ , we define the *local part* of  $T$  in  $S$  to be the set of triangles that have a side on one of the two lenses of  $S$ . We say that segment  $s$  is *active* in  $T$  if its local part includes at least one triangle that touches both lenses of  $S$ , and that  $s$  is *passive* otherwise.

► **Lemma 7.** *The active segments of a triangulation form a non-crossing set.*

**Proof.** If two segments cross, any lens-to-lens triangle within one segment crosses any lens-to-lens triangle of the other. But in a triangulation, no two triangles can cross each other. ◀

► **Lemma 8.** *Let  $T$  be a triangulation of  $P_A$ , let  $s$  be an active segment of  $T$ , let  $S$  be the corresponding gadget, and let  $\Pi_S$  be the polygon formed by intersecting  $P_A$  with the convex hull of  $S$ . Then the intersection of  $T$  and  $\Pi_S$  is a triangulation of  $\Pi_S$ .*

**Proof.** By the construction of  $P_A$ , every edge of  $T$  must belong to a single gadget (as there are no other visibilities between pairs of vertices in  $P_A$ ). By the same argument as in Lemma 7, no segment with its endpoints in a single other gadget than  $S$  can cross  $\Pi_S$ . Therefore, all edges of  $T$  that include an interior point of  $\Pi_S$  must connect two vertices of  $S$ , and lie within  $\Pi_S$ . It follows that all triangles of  $T$  that include an interior point of  $\Pi_S$  must also lie within  $\Pi_S$ , so that  $T$  restricts to a triangulation of  $\Pi_S$ . ◀



■ **Figure 4** Illustration for Lemma 10: Closing off a lens by connecting the endpoints of its path (red dashed edge) forms a polygon with as many distinct triangulations as lens vertices.

► **Corollary 9.** *Let  $T$  be a triangulation of  $P_A$ , and let  $L$  be a lens in polygonal chain  $C_L$  of a gadget  $S$ . Suppose that some active segment of  $T$  separates  $C_L$  from the rest of  $S$ . Let  $\Pi_L$  be the polygon formed from  $C$  by adding one more edge connecting the two endpoints of  $C$ . Then  $T$  intersects  $\Pi_L$  in a triangulation of  $\Pi_L$ .*

**Proof.** By Lemma 8,  $T$  contains the edge connecting the endpoints of  $C$ . This edge separates  $\Pi_L$  from the rest of the polygon, so each triangle of  $T$  must lie either entirely within or entirely outside  $\Pi_L$ . But the triangles within  $\Pi_L$  cover  $\Pi_L$  (as the triangles of  $T$  cover all of  $P_A$ ) so they form a triangulation of  $\Pi_L$ . ◀

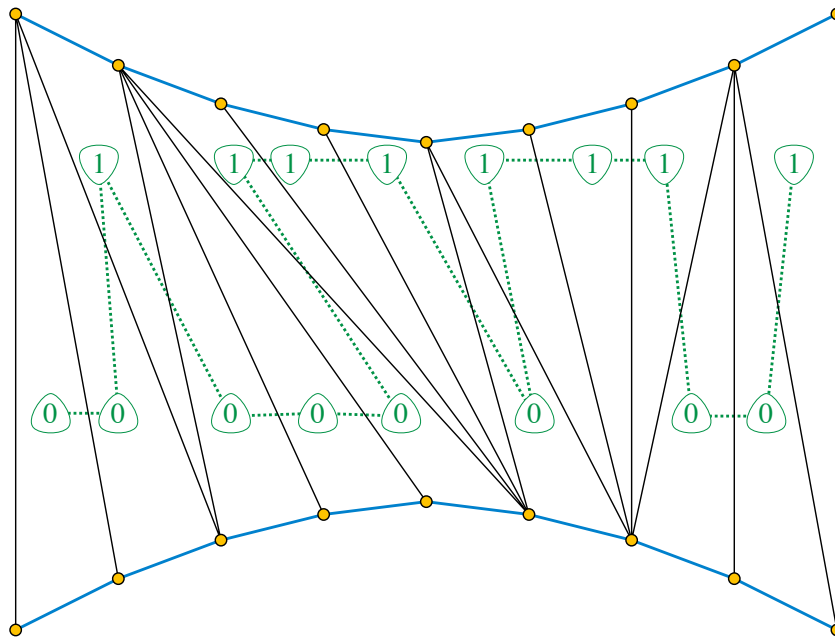
These claims already allow us to produce a precise count of the triangulations whose active segments form a maximum non-crossing subset, as a function of the number of red segments in the subset, which we will do in the next section. However, we also need to bound the number of triangulations with smaller sets of active segments, and show that (with an appropriate choice of the parameters  $a$  and  $b$ ) they form a negligible number of triangulations compared to the ones coming from maximum non-crossing subsets.

### 4.3 Counting within a gadget

It is straightforward to count triangulations of the polygon formed by a single lens:

► **Lemma 10.** *Let  $\Pi_L$  be the polygon formed from a lens by closing its path off by a single edge  $uv$ , as in Corollary 9. Then the number of distinct triangulations of  $\Pi_L$  equals the number of vertices of the lens ( $a$  for a blue lens,  $a + b$  for a red lens).*

**Proof.** Any triangulation of  $\Pi_L$  is completely determined by the choice of apex of the triangle that has  $uv$  as one of its sides. All the remaining edges of the triangulation must each connect one vertex of the lens to the remaining visible endpoint of  $uv$ , the only vertex visible to that lens vertex (Figure 4). The number of choices for the apex equals the number of lens vertices. ◀



■ **Figure 5** Encoding a triangulation of the convex hull of two lenses by a sequence of bits.

More generally, for any passive segment, we have:

► **Lemma 11.** *Let  $S$  be the gadget of segment  $s$ . Then the number of combinatorially distinct ways to choose the local part in  $S$  of a triangulation of  $P_A$  for which  $s$  is passive is polynomial in the number of lens vertices of  $S$ .*

**Proof.** Recall that the triangles of the local part each have one lens edge as one of their sides. Within each lens, the only choice is where to place the apex of each of these triangles. There are  $O(1)$  choices of apex vertex, and within each lens the edges whose triangles have a given apex must form a contiguous subsequence. There are only polynomially many ways of partitioning the sequences of lens edges into a constant number of contiguous subsequences. ◀

A similar argument also shows:

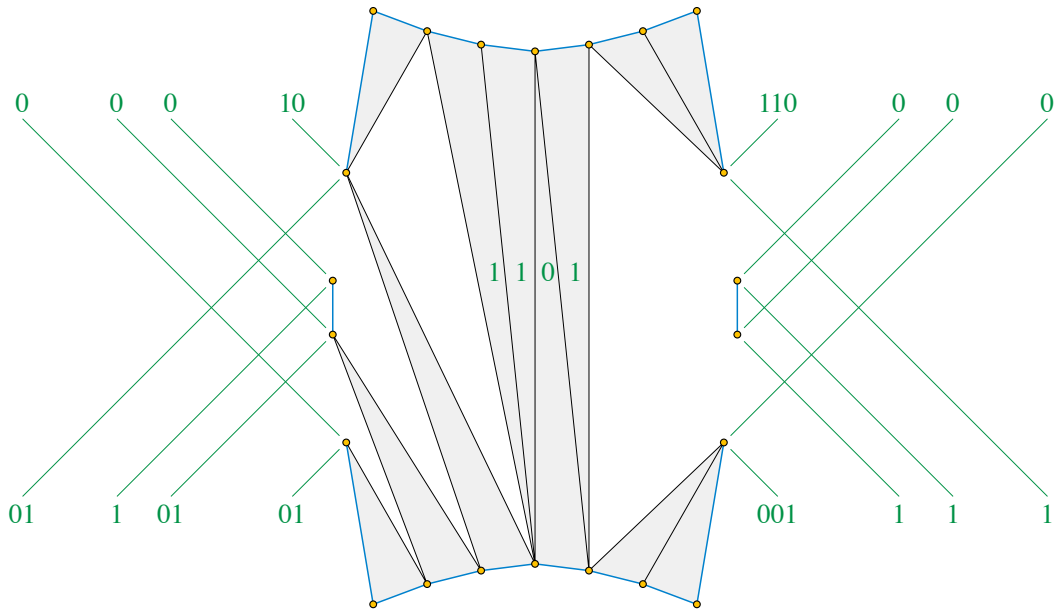
► **Lemma 12.** *Let  $T$  be a triangulation of  $P_A$  and let  $S$  be a gadget of  $P_A$ . Then only  $O(1)$  vertices of  $S$  can participate in triangles of  $T$  which are not part of the local part of  $T$  in  $S$ .*

**Proof.**  $S$  has  $O(1)$  vertices outside of its two lenses, so it has  $O(1)$  non-lens vertices that can participate in non-local triangles. A lens vertex can participate in a non-local triangle in one of two ways: either the triangle has one lens vertex and two non-lens vertices, or it has two vertices from opposite lenses and one non-lens vertex. There are  $O(1)$  edges of  $T$  between non-lens vertices of  $S$ , each of which is part of two triangles in  $T$ , so there are  $O(1)$  triangles in  $S \cap T$  with only one lens vertex. Each non-lens vertex of  $S$  can participate in only one triangle whose other two vertices are on opposite lenses, so again there are  $O(1)$  triangles of this type. Therefore, there are  $O(1)$  lens vertices in non-local triangles. ◀

It is sufficiently messy to count the triangulations of active gadgets that we provide here only approximate bounds. However, the exact number of triangulations can be found in polynomial time using the algorithm for counting triangulations of simple polygons [19, 32, 16].



33:10 Counting Polygon Triangulations is Hard



■ **Figure 6** Encoding the local part of a triangulation by a sequence of bits.

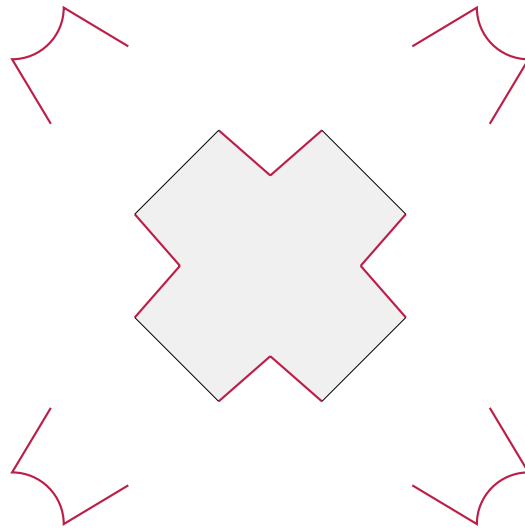
► **Lemma 13.** *Let  $\Pi_S$  be the polygon formed by intersecting polygon  $P_A$  with the convex hull of gadget  $S$ , as in Lemma 8. Let  $\ell$  be the number of lens vertices of  $\Pi_S$  ( $2a$  for a blue gadget,  $2a + 2b$  for a red gadget). Then the number of triangulations of  $\Pi_S$  is  $\Theta(2^\ell/\sqrt{\ell})$ .*

**Proof.** One can obtain  $\Omega(2^\ell/\sqrt{\ell})$  triangulations by choosing a maximal set of non-crossing lens-to-lens diagonals in  $\Pi_S$ , which form the edges of a triangulation of the convex hull of the two lenses, and then choosing arbitrarily a triangulation of the remaining polygons of  $O(1)$  vertices on either side of the convex hull of the two lenses. The triangles within any triangulation of the convex hull of the two lenses have a path as their dual graph, and if they are ordered along the path then the triangulation itself is determined by a sequence of bits (one bit per triangle) that denote whether each triangle includes an edge of one lens or of the other lens [22] (Figure 5). There are  $\ell - 2$  bits in the sequence, exactly half of which must be zeros and half of which must be ones, so the number of these triangulations is

$$\binom{\ell - 2}{(\ell - 2)/2} = \Omega\left(\frac{2^\ell}{\sqrt{\ell}}\right).$$

In the other direction, we can produce an overestimate of the number of distinct local parts of triangulations of  $\Pi_S$  by a similar method of counting balanced binary strings of slightly greater length. As in Figure 5, consider one of the two lenses as being “upper” (labeled by 1’s in the binary string) and the other as being “lower” (labeled by 0’s). Similarly, we can describe the two sides of the tube of the gadget as being left or right. Suppose also that each side of a tube of the given gadget has  $t$  internal vertices (vertices that do not belong to either lens). We describe the local part of a triangulation by a sequence of bits, as follows (Figure 6):

- $t$  blocks of 0-bits, each terminated by a single 1-bit, specifying how many edges of the lower lens form triangles whose apex is at each of the internal vertices of the left tube.
- $t$  blocks of 1-bits, each terminated by a single 0-bit, specifying how many edges of the upper lens form triangles whose apex is at each of the internal vertices of the left tube.



■ **Figure 7** The twelve-sided polygon in the center of two crossing red gadgets.

- A sequence of bits as before describing the left-to-right sequence of triangles that have all three vertices on the two lenses, with a 0-bit for a triangle with a side on the lower lens and a 1-bit for a triangle with a side on the upper lens.
- $t$  blocks of 0-bits, each terminated by a single 1-bit, specifying how many edges of the lower lens form triangles whose apex is at each of the internal vertices of the right tube.
- $t$  blocks of 1-bits, each terminated by a single 0-bit, specifying how many edges of the upper lens form triangles whose apex is at each of the internal vertices of the right tube.

The resulting sequence of bits uniquely describes each distinct local part, has  $2\ell - 2 + 4t$  bits, and has equal numbers of 0- and 1-bits. Not all such sequences of bits describe a valid local part (they may specify triangles connecting the lower and upper lenses to the tubes that cross each other) but this is non-problematic. There are  $O(2^\ell/\sqrt{\ell})$  possible sequences of bits of this type, so there are  $O(2^\ell/\sqrt{\ell})$  local parts of triangulations of  $\Pi_S$ . Each local part leaves remaining untriangulated regions on the left and right sides of  $\Pi_S$  with a bounded total number of vertices (as in the proof of Lemma 12) so it can be extended to a complete triangulation in  $O(1)$  distinct ways. Therefore, the total number of triangulations of  $\Pi_S$  is  $O(2^\ell/\sqrt{\ell})$ . ◀

#### 4.4 Global counting

The key properties of the number of triangulations of  $P_A$  (with an appropriate choice of  $a$  and  $b$ ) that allow us to prove our counting reduction are that

- the number of triangulations coming from any particular maximum non-crossing subset of  $A$  can be determined only from  $n$  and the number of red segments in the subset, and
- non-crossing subsets that use fewer segments than the maximum, or that use the maximum total number of segments but fewer than the maximum possible number of red segments, contribute a negligible fraction of the total number of triangulations.

In this section we make these notions precise.

▶ **Definition 14.** We define the following numerical parameters:

- $\alpha$  is the number of triangulations of  $\Pi_S$  for a blue gadget  $S$ , as bounded asymptotically as a function of parameter  $a$  by Lemma 13.

### 33:12 Counting Polygon Triangulations is Hard

- $\beta$  is the number of triangulations of  $\Pi_S$  for a red gadget  $S$ , as bounded asymptotically as a function of parameters  $a$  and  $b$  by Lemma 13.
- $\gamma$  is the number of triangulations of the twelve-sided polygon formed in the center of two crossing red gadgets by the four shared vertices of the two gadgets and their four neighbors in each gadget (Figure 7).
- $q_{n,r}$ , for any  $n \geq 0$  and  $0 \leq r \leq n/2$ , is  $\alpha^{n-r} \beta^r \gamma^{(n-2r)/2} a^{2r} (a+b)^{2(n-r)} 2^{3r}$ .

We do not derive an explicit formula for  $\alpha$  and  $\beta$ . Therefore, it is important for us that they can be calculated in polynomial time by dynamic programming, as (for appropriate choices of  $a$  and  $b$ ) they count triangulations of simple polygons of polynomial size.

► **Lemma 15.** *Let  $A$  be a red–blue arrangement with  $n$  red and  $n$  blue segments. Let  $\Xi$  be a maximum non-crossing subset of  $A$ , with  $r = r(\Xi)$  red segments. Then the number of triangulations of  $P_A$  in which  $\Xi$  forms the active set is  $q_{n,r}$ .*

**Proof.** By Lemma 8, the triangulations with  $\Xi$  active can be partitioned into the polygons  $\Pi_S$  for the active segments of  $\Xi$  and the remaining sub-polygons left by the removal of these polygons, each of which can be triangulated independently and each of which contributes a term to the product in the definition of  $q_{n,r}$ . There are  $r$  active red segments and  $n - r$  active blue segments, each of which contributes a factor of  $\beta$  or  $\alpha$  (respectively) to the product by which  $q_{n,r}$  was defined. There are  $2r$  lenses of passive blue segments and  $2(n - r)$  lenses of passive red segments, each of which contributes a term of  $a$  or  $(a + b)$  to the product, respectively, by Lemma 10.

The  $r$  active red segments and  $r$  passive red segments that they cross leave  $n - 2r$  red segments that are passive but not crossed by another red segment. These  $n - 2r$  red segments form  $(n - 2r)/2$  twelve-sided polygons where their gadgets cross each other in pairs, with each pair contributing a factor of  $\gamma$  to the product.

The remaining factor of  $2^{3r}$  in the product comes from the  $r$  passive blue gadgets, each of which has a central quadrilateral between the two red active segments that cross it, and from the  $r$  passive red gadgets that are crossed by an active red gadget, each of which has two central quadrilaterals between the consecutive pairs of the three active segments that cross it. These  $3r$  quadrilaterals each can be triangulated in two ways. ◀

► **Corollary 16.** *Let  $a \geq b$ . Then  $q_{n,r}/q_{n,r-1} = \Theta(2^{2b})$ .*

**Proof.** Increasing the exponent of  $\beta$  by one and decreasing the exponent of  $\alpha$  by one leads to the  $2^{2b}$  change in the total. All of the other changes to the formula for  $q_{n,r}$  are bounded either by a constant (independent of  $a$  and  $b$ ) or by a power of  $a/(a + b)$ , which is at least  $1/2$  by the assumption that  $a \geq b$ . ◀

► **Lemma 17.** *Suppose the parameters  $a$  and  $b$  are both upper-bounded by polynomials of  $n$ . Then*

$$\log_2 q_{n,r} = 2an + 2rb \pm O(n \log n),$$

where the constant in the  $O$ -notation depends on the bounds on  $a$  and  $b$ .

**Proof.** In the formula for  $q_{n,r}$ , each factor of  $\alpha$  contributes  $2a - O(\log n)$  to the logarithm, each factor of  $\beta$  contributes  $2a + 2b - O(\log n)$ , each factor of  $a$  or  $b$  contributes  $O(\log n)$ , and each factor of  $\gamma$  or  $2$  contributes  $O(1)$ . The result follows by adding these contributions according to their exponents. ◀

► **Lemma 18.** *Suppose the parameters  $a$  and  $b$  are both upper-bounded by polynomials of  $n$ . Let  $p$  denote the number of triangulations of  $P_A$  that do not have a maximum non-crossing subset of  $A$  as their active set. Then*

$$\log_2 p \leq 2a(n-1) + 2b(n-1) + O(n \log n).$$

**Proof.** There are  $2^{2n}$  subsets of the  $2n$  segments of  $A$ , and a smaller number of these subsets that can be a non-maximum active set of a triangulation. The choice of this subset adds  $O(n)$  to the logarithm. For each non-maximum active set, there can be at most  $n-1$  active segments, and the triangulations of the corresponding polygons adds at most  $2a(n-1) + 2b(n-1) - O(n \log n)$  to the logarithm. The local parts of the passive gadgets can be chosen in a number of ways per gadget that is polynomial in  $a$  and  $b$ , adding another  $O(n \log n)$  term to the logarithm. Once the active gadgets have been triangulated and the local parts of the passive gadgets have been chosen, the regions that remain to be triangulated have a total of  $O(n)$  vertices, so the number of ways to triangulate them contributes another  $O(n)$  to the logarithm. ◀

## 4.5 Completing the reduction

We have already described how to transform an arrangement  $A$  into a polygon  $P_A$ , modulo the choice of the parameters  $a$  and  $b$ . To complete the description of our reduction, we need to set  $a$  and  $b$  and we need to describe how to recover the number of maximum non-crossing subsets of  $A$  from the number  $N$  of triangulations of  $P_A$ .

Given a red–blue arrangement  $A$  with  $n$  red and  $n$  blue segments, set  $b = 2n$ . By Corollary 16,  $q_{n,r-1}$  and  $q_{n,r}$  differ by a factor of approximately  $2^{4n}$ , much larger than the  $2^{2n}$  bound on the total number of non-crossing sets of segments. Therefore, for any given  $r$  and for all sufficiently large  $n$ , the number of triangulations with a maximum non-crossing set of active segments that includes fewer than  $r$  red segments is strictly less than  $q_{n,r}$ . Next, set  $a = 3n^2$ . This is large enough that, comparing the bounds of Lemma 17 and Lemma 18, the number  $p$  of triangulations whose active set is non-maximum is strictly smaller than  $q_{n,0}$ , as the  $O(n \log n)$  term and larger multiple of  $b$  in the formula for  $\log_2 p$  are not large enough to make up for the smaller multiple of  $a$  in the same formula.

With these choices of  $a$  and  $b$ , the same reasoning also shows that (for all sufficiently large  $n$ ) the total number of triangulations of any  $P_{A'}$  for any arrangement  $A'$  with fewer red and blue segments than  $A$  is strictly smaller than  $q_{n,0}$ . Therefore, from the total number  $N$  of triangulations of  $P_A$  we can unambiguously determine the size of  $A$ .

With these considerations, we are ready to prove the correctness of our reduction:

**Proof of Theorem 1.** Counting triangulations of a given polygon  $P$  is in #P by Lemma 2, so we can complete the proof by describing a polynomial-time counting reduction from the number of maximum non-crossing subsets of a red–blue arrangement  $A$  (proved #P-hard in Lemma 6). To transform  $A$  into a polygon, let  $n$  be the number of red segments in  $A$ . Our reduction will work for all sufficiently large  $n$ ; for the bounded values of  $n$  that are not sufficiently large, directly compute the number of maximum non-crossing subsets of  $A$  and construct a polygon with the same number of triangulations using the formula for numbers of triangulations of lens polygons of Lemma 10. Otherwise, choose  $a = 4n^2$  and  $b = 3n$  as above, and construct the polygon  $P_A$ . To transform the number  $N$  of triangulations of the given polygon into the number of maximum non-crossing subsets of  $A$ , first check whether this number is sufficiently small that it comes from our special-case construction for bounded values of  $n$ . If so, directly decode it to the number of maximum non-crossing subsets.

In the remaining case, recover  $n$  as the unique value that could have produced a polygon  $P_A$  with  $N$  triangulations. For each choice of  $r$  from  $n$  down to 0 (in decreasing order) compute the number of maximum non-crossing subsets with  $r$  red segments as  $\lfloor N'/q_{n,r} \rfloor$  (where  $N'$  starts with the value  $N$  and is reduced as the algorithm progresses) and then replace  $N'$  by  $N' \bmod q_{n,r}$  before proceeding to the next value of  $r$ . Sum the numbers of maximum-non-crossing subsets obtained for each value of  $r$  to obtain the total number of maximum non-crossing subsets.

When computing the number of non-crossing subsets for each value of  $r$ , the contributions from triangulations whose active segments include more red segments than  $r$  will already have been subtracted off, by induction. The contributions from triangulations whose active segments include fewer red segments than  $r$ , or from triangulations that do not have maximum non-crossing sets of active segments, will sum to less than a single multiple of the number of triangulations for each non-crossing set with the given number of red segments, as discussed above. Therefore, each number of non-crossing subsets is computed correctly. ◀

## 5 Conclusions and open problems

We have shown that counting triangulations of polygons (with holes) is #P-complete under Turing reductions. It would be of interest to tighten this result to show completeness under counting reductions, or even under parsimonious reductions. Can this be done, either by strengthening the type of reduction used for the underlying graph problem that we reduce from (independent sets in regular planar graphs) or by finding a different reduction for triangulations that bypasses the Turing reductions used for this graph problem?

In a triangulation of a polygon with holes, every hole has a diagonal connecting its leftmost vertex to a vertex to the left of it in another boundary component. By testing all combinations of these left diagonals, and using dynamic programming to count triangulations of the simple polygon formed by cutting the input along one of these sets of diagonals (avoiding triangulations that use previously-tested diagonals) it is possible to count triangulations of an  $n$ -vertex polygon with  $h$  holes in time  $O(n^{h+3})$ . Is the dependence on  $h$  in the exponent of  $n$  necessary, or is there a fixed-parameter tractable algorithm for this problem?

More generally, there are many other counting problems in discrete geometry for which we neither know a polynomial time algorithm nor a hardness proof. For instance, we do not know the complexity of counting triangulations, planar graphs, non-crossing Hamiltonian cycles, non-crossing spanning trees, or non-crossing matchings of sets of  $n$  points in the plane. Are these problems hard?

---

## References

- 1 Oswin Aichholzer, Victor Alvarez, Thomas Hackl, Alexander Pilz, Bettina Speckmann, and Birgit Vogtenhuber. An improved lower bound on the minimum number of triangulations. In Sándor Fekete and Anna Lubiw, editors, *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages A7:1–A7:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.7.
- 2 Oswin Aichholzer, Thomas Hackl, Clemens Huemer, Ferran Hurtado, Hannes Krasser, and Birgit Vogtenhuber. On the number of plane geometric graphs. *Graphs and Combinatorics*, 23(suppl. 1):67–84, 2007. doi:10.1007/s00373-007-0704-5.
- 3 Oswin Aichholzer, Ferran Hurtado, and Marc Noy. A lower bound on the number of triangulations of planar point sets. *Comput. Geom. Theory and Applications*, 29(2):135–145, 2004. doi:10.1016/j.comgeo.2004.02.003.

- 4 Oswin Aichholzer, David Orden, Francisco Santos, and Bettina Speckmann. On the number of pseudo-triangulations of certain point sets. *J. Combin. Theory Ser. A*, 115(2):254–278, 2008. doi:10.1016/j.jcta.2007.06.002.
- 5 Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting triangulations and other crossing-free structures via onion layers. *Discrete Comput. Geom.*, 53(4):675–690, 2015. doi:10.1007/s00454-015-9672-3.
- 6 Victor Alvarez, Karl Bringmann, Saurabh Ray, and Raimund Seidel. Counting triangulations and other crossing-free structures approximately. *Comput. Geom. Theory and Applications*, 48(5):386–397, 2015. doi:10.1016/j.comgeo.2014.12.006.
- 7 Victor Alvarez and Raimund Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In Timothy Chan and Rolf Klein, editors, *Proceedings of the 29th Annual Symposium on Computational Geometry (SoCG'13)*, pages 1–8, New York, 2013. ACM. doi:10.1145/2462356.2462392.
- 8 Emile E. Anclin. An upper bound for the number of planar lattice triangulations. *J. Combin. Theory Ser. A*, 103(2):383–386, 2003. doi:10.1016/S0097-3165(03)00097-9.
- 9 Takao Asano, Tetsuo Asano, and Hiroshi Imai. Partitioning a polygonal region into trapezoids. *J. ACM*, 33(2):290–312, 1986. doi:10.1145/5383.5387.
- 10 Andrei Asinowski and Günter Rote. Point sets with many non-crossing perfect matchings. *Comput. Geom. Theory and Applications*, 68:7–33, 2018. doi:10.1016/j.comgeo.2017.05.006.
- 11 Marshall W. Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, 2nd edition, 1995.
- 12 Sergei Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Comput. Geom. Theory and Applications*, 23(3):271–279, 2002. doi:10.1016/S0925-7721(02)00111-6.
- 13 Hervé Brönnimann, Lutz Kettner, Michel Pocchiola, and Jack Snoeyink. Counting and enumerating pointed pseudotriangulations with the greedy flip algorithm. *SIAM J. Comput.*, 36(3):721–739, 2006. doi:10.1137/050631008.
- 14 Jérémie Chalopin and Daniel Gonçalves. Every planar graph is the intersection graph of segments in the plane: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 631–638, 2009. doi:10.1145/1536414.1536500.
- 15 Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990. doi:10.1007/BF02122694.
- 16 Q. Ding, J. Qian, W. Tsang, and C. Wang. Randomly generating triangulations of a simple polygon. In Lusheng Wang, editor, *Computing and Combinatorics: 11th Annual International Conference, COCOON 2005, Kunming, China, August 16–19, 2005, Proceedings*, volume 3595 of *Lecture Notes in Computer Science*, pages 471–480. Springer, Berlin, 2005. doi:10.1007/11533719\_48.
- 17 Adrian Dumitrescu, André Schulz, Adam Sheffer, and Csaba D. Tóth. Bounds on the maximum multiplicity of some common geometric graphs. *SIAM J. Discrete Math.*, 27(2):802–826, 2013. doi:10.1137/110849407.
- 18 Martin Dyer, Leslie Ann Goldberg, and Mike Paterson. On counting homomorphisms to directed acyclic graphs. *J. ACM*, 54(6):A27:1–A27:23, 2007. doi:10.1145/1314690.1314691.
- 19 Peter Epstein and Jörg-Rüdiger Sack. Generating triangulations at random. *ACM Trans. Model. Comput. Simul.*, 4(3):267–278, 1994. doi:10.1145/189443.189446.
- 20 Philippe Flajolet and Marc Noy. Analytic combinatorics of non-crossing configurations. *Discrete Math.*, 204(1-3):203–229, 1999. doi:10.1016/S0012-365X(98)00372-0.
- 21 Alfredo García, Marc Noy, and Javier Tejel. Lower bounds on the number of crossing-free subgraphs of  $K_N$ . *Comput. Geom. Theory and Applications*, 16(4):211–221, 2000. doi:10.1016/S0925-7721(00)00010-9.
- 22 Ferran Hurtado, Marc Noy, and Jorge Urrutia. Flipping edges in triangulations. *Discrete Comput. Geom.*, 22(3):333–346, 1999. doi:10.1007/PL00009464.



- 23 F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Cambridge Philos. Soc.*, 108(1):35–53, 1990. doi:10.1017/S0305004100068936.
- 24 Volker Kaibel and Günter M. Ziegler. Counting lattice triangulations. In C. D. Wensley, editor, *Surveys in Combinatorics 2003: Papers from the 19th British Combinatorial Conference held at the University of Wales, Bangor, June 29–July 4, 2003*, volume 307 of *London Math. Soc. Lecture Note Ser.*, pages 277–307. Cambridge Univ. Press, Cambridge, UK, 2003. arXiv:math/0211268.
- 25 Pegah Kamousi and Subhash Suri. Stochastic minimum spanning trees and related problems. In Philippe Flajolet and Daniel Panario, editors, *Proceedings of the Eighth Workshop on Analytic Algorithmics and Combinatorics, ANALCO 2011, San Francisco, California, USA, January 22, 2011*, pages 107–116. SIAM, 2011. doi:10.1137/1.9781611973013.12.
- 26 Goos Kant and Hans L. Bodlaender. Triangulating planar graphs while minimizing the maximum degree. *Inform. and Comput.*, 135(1):1–14, 1997. doi:10.1006/inco.1997.2635.
- 27 Marek Karpinski, Andrzej Lingas, and Dzmityry Sledneu. A QPTAS for the base of the number of crossing-free structures on a planar point set. *Theoretical Computer Science*, 711:56–65, 2018. doi:10.1016/j.tcs.2017.11.003.
- 28 Nathan Linial. Hard enumeration problems in geometry and combinatorics. *SIAM J. Algebraic Discrete Methods*, 7(2):331–335, 1986. doi:10.1137/0607036.
- 29 Anna Lubiw. Decomposing polygonal regions into convex quadrilaterals. In Joseph O’Rourke, editor, *Proceedings of the 1st Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 97–106, New York, 1985. ACM. doi:10.1145/323233.323247.
- 30 Dániel Marx and Tillmann Miltzow. Peeling and nibbling the cactus: subexponential-time algorithms for counting triangulations and related problems. In Sándor Fekete and Anna Lubiw, editors, *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages A52:1–A52:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.52.
- 31 Alexander Pilz and Carlos Seara. Convex quadrangulations of bichromatic point sets. In *Proceedings of the 33rd European Workshop on Computational Geometry (EuroCG 2017)*, 2017. URL: [https://mat-web.upc.edu/people/carlos.seara/data/publications/internationalConferences/EuroCG-17\\_paper\\_23.pdf](https://mat-web.upc.edu/people/carlos.seara/data/publications/internationalConferences/EuroCG-17_paper_23.pdf).
- 32 Saurabh Ray and Raimund Seidel. A simple and less slow method for counting triangulations and for related problems. In *Proceedings of the 20th European Workshop on Computational Geometry (EuroCG 2004)*, 2004. URL: <https://hdl.handle.net/11441/55368>.
- 33 Francisco Santos and Raimund Seidel. A better upper bound on the number of triangulations of a planar point set. *J. Combin. Theory Ser. A*, 102(1):186–193, 2003. doi:10.1016/S0097-3165(03)00002-5.
- 34 Edward R. Scheinerman. *Intersection Classes and Multiple Intersection Parameters of Graphs*. PhD thesis, Princeton University, 1984.
- 35 Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proceedings of the First Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 1990, 22-24 January 1990, San Francisco, California, USA*, pages 138–148, 1990. URL: <https://dl.acm.org/citation.cfm?id=320176.320191>.
- 36 Raimund Seidel. On the number of triangulations of planar point sets. *Combinatorica*, 18(2):297–299, 1998. doi:10.1007/PL00009823.
- 37 Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electron. J. Combin.*, 18(1):P70:1–P70:74, 2011. URL: [https://emis.ams.org/journals/EJC/Volume\\_18/PDF/v18i1p70.pdf](https://emis.ams.org/journals/EJC/Volume_18/PDF/v18i1p70.pdf).
- 38 Micha Sharir, Adam Sheffer, and Emo Welzl. Counting plane graphs: perfect matchings, spanning cycles, and Kasteleyn’s technique. *J. Combin. Theory Ser. A*, 120(4):777–794, 2013. doi:10.1016/j.jcta.2013.01.002.

- 39 Micha Sharir and Emo Welzl. On the number of crossing-free matchings, cycles, and partitions. *SIAM J. Comput.*, 36(3):695–720, 2006. doi:10.1137/050636036.
- 40 Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2):398–427, 2001. doi:10.1137/S0097539797321602.
- 41 L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979. doi:10.1016/0304-3975(79)90044-6.
- 42 Manuel Wettstein. Counting and enumerating crossing-free geometric graphs. *J. Comput. Geom.*, 8(1):47–77, 2017. URL: <https://jocg.org/index.php/jocg/article/view/280>.
- 43 Mingji Xia, Peng Zhang, and Wenbo Zhao. Computational complexity of counting problems on 3-regular planar graphs. *Theoretical Computer Science*, 384(1):111–125, 2007. doi:10.1016/j.tcs.2007.05.023.





# Topologically Trivial Closed Walks in Directed Surface Graphs

Jeff Erickson 

University of Illinois at Urbana-Champaign, Urbana, IL, USA  
jeffe@illinois.edu

Yipu Wang

University of Illinois at Urbana-Champaign, Urbana, IL, USA  
ywang298@illinois.edu

---

## Abstract

Let  $G$  be a directed graph with  $n$  vertices and  $m$  edges, embedded on a surface  $S$ , possibly with boundary, with first Betti number  $\beta$ . We consider the complexity of finding closed directed walks in  $G$  that are either contractible (trivial in homotopy) or bounding (trivial in integer homology) in  $S$ . Specifically, we describe algorithms to determine whether  $G$  contains a simple contractible cycle in  $O(n + m)$  time, or a contractible closed walk in  $O(n + m)$  time, or a bounding closed walk in  $O(\beta(n + m))$  time. Our algorithms rely on subtle relationships between strong connectivity in  $G$  and in the dual graph  $G^*$ ; our contractible-closed-walk algorithm also relies on a seminal topological result of Hass and Scott. We also prove that detecting simple bounding cycles is NP-hard.

We also describe three polynomial-time algorithms to compute shortest contractible closed walks, depending on whether the fundamental group of the surface is free, abelian, or hyperbolic. A key step in our algorithm for hyperbolic surfaces is the construction of a context-free grammar with  $O(g^2 L^2)$  non-terminals that generates all contractible closed walks of length at most  $L$ , and only contractible closed walks, in a system of quads of genus  $g \geq 2$ . Finally, we show that computing shortest simple contractible cycles, shortest simple bounding cycles, and shortest bounding closed walks are all NP-hard.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** computational topology, surface-embedded graphs, homotopy, homology, strong connectivity, hyperbolic geometry, medial axes, context-free grammars

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.34

**Category** Regular Paper

**Related Version** A full version of this paper is available at [arXiv:1812.01564](https://arxiv.org/abs/1812.01564) [30].

**Funding** Research on this paper was partially supported by NSF grant CCF-1408763.

**Acknowledgements** The first author would like to thank Tillmann Miltzow for asking an annoying question that led to this work, and to apologize for still being unable to answer it.

## 1 Introduction

A key step in several algorithms for surface-embedded graphs is finding a shortest closed walk and/or simple cycle in the input graph with some interesting topological property. There is a large body of work on finding short interesting walks and cycles in undirected surface graphs, starting with Thomassen’s seminal *3-path condition* [49, 55]. For example, efficient algorithms are known for computing shortest non-contractible and non-separating cycles [7, 12, 27, 28, 46], shortest contractible closed walks [10], simple cycles that are shortest in their own homotopy class [11], and shortest closed walks in a *given* homotopy [18] or homology class [29], and for detecting simple cycles that are either contractible, non-contractible, or non-separating [9]. On the other hand, several related problems are known to be NP-hard, including computing



© Jeff Erickson and Yipu Wang;

licensed under Creative Commons License CC-BY

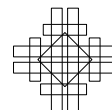
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 34; pp. 34:1–34:17

Leibniz International Proceedings in Informatics



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



shortest splitting closed walks [13], computing shortest separating cycles [6], computing shortest closed walks in a given homology class [14], and deciding whether a surface graph contains a simple separating or splitting cycle [9].

Directed surface graphs are much less understood, in part because they do not share convenient properties of undirected graphs, such as Thomassen’s 3-path condition [49, 55], or the assumption that if shortest paths are unique, then two shortest paths cross at most once [26]. The first progress in this direction was a pair of algorithms by Cabello, Colin de Verdière, and Lazarus [8], which compute shortest non-contractible and non-separating cycles in directed surface graphs, with running times  $O(n^2 \log n)$  and  $O(g^{1/2} n^{3/2} \log n)$ . Erickson and Nayyeri described an algorithm to compute the shortest non-separating cycles in  $2^{O(g)} n \log n$  time [29]. Later Fox [33] described algorithms to compute shortest non-contractible cycles in  $O(\beta^3 n \log n)$  time and shortest non-separating cycles in  $O(\beta^2 n \log n)$  time on surfaces with first Betti number  $\beta$ . (As in previous papers, the input size  $n$  is the total number of vertices and edges in the input graph.)

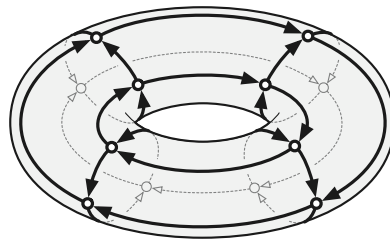
This paper describes the first algorithms and hardness results for finding topologically *trivial* closed walks in directed surface graphs. Our results extend similar results of Cabello, Colin de Verdière, and Lazarus [6, 9] for undirected surface graphs; however, our algorithms require several new techniques, both in design and analysis. (On the other hand, our NP-hardness proofs are actually simpler than the corresponding proofs for undirected graphs!)

We present results for eight different problems, determined by three independent design choices. First, we consider two types of “trivial” closed walks: *contractible* walks, which can be continuously deformed to a point, and *bounding* walks, which are weighted sums of face boundaries. (See Section 2 for more detailed definitions.) Second, like Cabello et al. [6, 8, 9], we carefully distinguish between closed walks and simple cycles throughout the paper. Finally, we consider two different goals: deciding whether a given directed graph contains a trivial cycle or closed walk, and finding the shortest trivial cycle or closed walk in a given directed graph (possibly with weighted edges). Crucially, our algorithms do *not* assume that the faces of the input embedding are open disks. Our results are summarized in Table 1.

■ **Table 1** Our results;  $\beta$  is the first Betti number of the underlying surface.

| Structure                 | Surface  | Any          | Shortest             |
|---------------------------|----------|--------------|----------------------|
| Simple contractible cycle | any      | $O(n)$       | NP-hard              |
| Contractible closed walk  | annulus  | $O(n)$       | $O(n^2 \log \log n)$ |
|                           | torus    | $O(n)$       | $O(n^3 \log \log n)$ |
|                           | boundary | $O(n)$       | $O(\beta^5 n^3)$     |
|                           | other    | $O(n)$       | $O(\beta^6 n^9)$     |
| Simple bounding cycle     | any      | NP-hard      | NP-hard              |
| Bounding closed walk      | any      | $O(\beta n)$ | NP-hard              |

In Section 3, we describe linear-time algorithms to determine whether a directed surface graph contains a simple contractible cycle or a contractible closed walk, matching similar algorithms for undirected graphs by Cabello et al. [9]. Our algorithms are elementary: After removing some obviously useless edges, we report success if and only if some face of the input embedding has a (simple) contractible boundary. However, the proofs of correctness require careful analysis of the dual graphs, and the correctness proof for contractible closed walks relies on a subtle topological lemma of Hass and Scott [38]. These two problems are nontrivial for directed graphs *even if* every face of the input embedding is a disk; see Figure 1.



■ **Figure 1** A cellularly embedded directed graph with no contractible or bounding closed walks.

In Section 4, we describe an algorithm to determine whether a directed surface graph contains a *bounding* closed walk in  $O(\beta n)$  time. We exploit a careful analysis of the interplay between strong connectivity in the input graph  $G$  and its dual graph  $G^*$ . With some additional effort, our algorithm can return an explicit description of a bounding closed walk in  $O(n^2)$  time if one exists; in the full paper [30, Section 4.2] we prove that this quadratic upper bound is optimal. This problem can be reduced to finding zero cycles in periodic (or “dynamic”) graphs [17, 40, 45]; a periodic graph is a graph whose edges are labeled with integer vectors; a zero cycle is a closed walk whose edge labels sum to the zero vector. However, all algorithms known for finding zero cycles rely on linear programming, and thus are much more complex and much less efficient than the specialized algorithm we present.

Next, we propose three polynomial-time algorithms to compute shortest contractible closed walks. Each of our algorithms is designed for a different class of surfaces, depending whether the surface’s fundamental group is abelian (the annulus and the torus), free (any surface with boundary), or hyperbolic (everything else). Our algorithm for the annulus and torus uses a standard covering-space construction, together with a recent algorithm of Mozes et al. [50]. For graphs on surfaces with boundary, we exploit the fact that the set of trivial words for any finitely-generated free group is a context-free language [51, 54]; this observation allows us to reduce to a small instance of the *CFG shortest path* problem [3, 4, 56]. We describe these two algorithms in detail in the full paper [30, Section 6]. In Section 5, we describe our algorithm for hyperbolic surfaces, which also reduces to CFG shortest paths; the main technical hurdle is the construction of an appropriate context-free grammar. Specifically, for any integers  $g \geq 2$  and  $L \geq 0$ , we construct a context-free grammar with  $O(g^2 L^2)$  nonterminals, in Chomsky normal form, that generates all contractible closed walks of length  $L$ , and only contractible closed walks, in a canonical genus- $g$  surface map called a *system of quads* [31, 47]. Our construction exploits well-known geometric properties of hyperbolic tilings.

Finally, in the full paper [30, Section 5], we prove that detecting simple bounding cycles, finding shortest simple contractible cycles, and finding shortest bounding closed walks are all NP-hard. Cabello [6] described a polynomial-time algorithm to compute the shortest simple contractible cycle in an *undirected* surface graph; thus, our reduction for that problem makes essential use of the fact that the input graph is directed. Cabello [6] and Cabello et al. [9] proved that the other two problems are NP-hard in undirected surface graphs. Our NP-hardness proofs closely follow theirs but are slightly simpler.

## 2 Background

**Directed graphs.** Let  $G$  be an arbitrary directed graph, possibly with loops and parallel edges. Each edge of  $G$  is directed from one endpoint, called its *tail*, to the other endpoint, called its *head*. An edge is a *loop* if its head and tail coincide. At the risk of confusing the reader, we sometimes write  $u \rightarrow v$  to denote an edge with tail  $u$  and head  $v$ .

## 34:4 Topologically Trivial Closed Walks in Directed Surface Graphs

A *walk* in  $G$  is an alternating sequence of vertices and edges  $v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_\ell$ , where  $v_i \rightarrow v_{i+1}$  is an edge in  $G$  for each index  $i$ ; this walk is *closed* if  $v_0 = v_\ell$ . A *simple cycle* is a closed walk that visits each vertex at most once. An *edge cut* in a directed graph  $G$  is a nonempty subset  $X$  of edges such that  $G \setminus X$  has two components, one containing the tails of edges in  $X$ , and other containing the heads of edges in  $X$ . A directed graph is *strongly connected* if it contains a walk from any vertex to any other vertex, or equivalently, if it contains no edge cuts.

An (integer) *circulation* in a directed graph  $G$  is a function  $\phi: E(G) \rightarrow \mathbb{N}$  that satisfies the balance constraint  $\sum_{u \rightarrow v} \phi(u \rightarrow v) = \sum_{v \rightarrow w} \phi(v \rightarrow w)$  for every vertex  $v$ . The *support* of  $\phi$  is the subset of edges  $e$  such that  $\phi(e) > 0$ . An *Euler tour* of  $\phi$  is a closed walk that traverses each edge  $e$  exactly  $\phi(e)$  times; such a walk exists if and only if the support of  $\phi$  is connected.

**Surfaces, embeddings, and duality.** A *surface* is a 2-manifold, possibly with boundary. A surface is *orientable* if it does not contain a Möbius band; we explicitly consider only orientable surfaces in this paper. A *closed curve* on a surface  $S$  is (the image of) a continuous map  $\gamma: S^1 \hookrightarrow S$ ; a closed curve is *simple* if this map is injective. The *boundary*  $\partial S$  of  $S$  consists of disjoint simple closed curves; the *interior* of  $S$  is the complement  $S^\circ = S \setminus \partial S$ . The *genus* of  $S$  is the maximum number of disjoint simple closed curves in  $S^\circ$  whose deletion leaves the surface connected. Up to homeomorphism, there is exactly one orientable surface with genus  $g$  and  $b$  boundary cycles, for any non-negative integers  $g$  and  $b$ . The *first Betti number* of an orientable surface is either  $2g$  if  $b = 0$ , or  $2g + b - 1$  if  $b > 0$ .

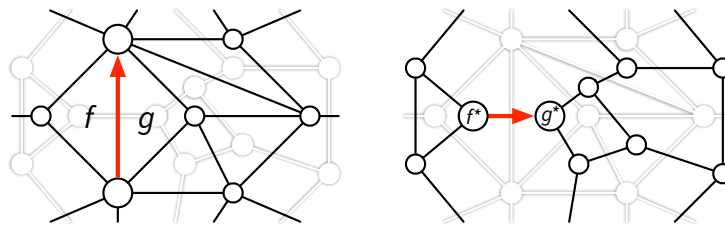
An *embedding* of  $G$  on a surface  $S$  is a continuous map that sends vertices of  $G$  to distinct points in  $S^\circ$ , and sends edges to interior-disjoint simple paths in  $S^\circ$  from their tails to their heads. In particular, if  $G$  contains two anti-parallel edges  $u \rightarrow v$  and its reversal  $v \rightarrow u$ , those edges are embedded as interior-disjoint paths. The embedding of  $G$  maps every (simple) closed walk in  $G$  to a (simple) closed curve in  $S^\circ$ ; we usually do not distinguish between a closed walk in  $G$  and its image in  $S$ .

We explicitly consider graphs with loops and parallel edges; however, without loss of generality, we assume that no loop edge is the boundary of a disk, and no two parallel edges are the boundary of a disk. With this assumption, Euler's formula implies that a graph with  $n$  vertices on a surface with first Betti number  $\beta$  has at most  $O(n + \beta)$  edges.

A *face* of an embedding is a component of the complement of the image of the graph. An embedding is *cellular* if every face is homeomorphic to an open disk. *Unlike most previous papers, we explicitly consider non-cellular graph embeddings*; a single face can have disconnected boundary and/or positive genus. Each directed edge  $e$  in a surface graph lies on the boundary of two (possibly equal) faces, called the *left shore* and *right shore* of  $e$ . We sometimes write  $f \uparrow f'$  to denote a directed edge whose left shore is  $f$  and whose right shore is  $f'$ . A *boundary face* of the embedding is any face that intersects the boundary of  $S$ .

A *dual walk* is an alternating sequence of faces and edges  $f_0 \uparrow f_1 \uparrow \cdots \uparrow f_\ell$ , where  $f_i \uparrow f_{i+1}$  is an edge in  $G$  for each index  $i$ . A dual walk is *closed* if its initial face  $f_0$  and final face  $f_\ell$  coincide; a dual walk is *simple* if the faces  $f_i$  are distinct (except possibly  $f_0 = f_\ell$ ). A simple closed dual walk is called a *cocycle*. A minimal edge cut in a surface graph is the disjoint union of at most  $g + 1$  cocycles.

Every surface embedding of a directed graph  $G$  defines a directed *dual graph*  $G^*$ , with one vertex  $f^*$  for each face  $f$  of  $G$ , and one edge  $e^*$  for each edge  $e$  of  $G$ . Specifically, for any edge  $e$  in  $G$ , we have  $\text{tail}(e^*) = \text{left}(e)^*$  and  $\text{head}(e^*) = \text{right}(e)^*$ , as shown in Figure 2. We will treat  $G^*$  exclusively as an abstract directed graph. Every dual walk in  $G$  corresponds to a walk in  $G^*$ ; in particular, every cocycle in  $G$  corresponds to a cycle in  $G^*$ .



■ **Figure 2** Duality in directed surface graphs.

**Contractible and bounding.** Let  $\alpha: [0, 1] \rightarrow S$  and  $\beta: [0, 1] \rightarrow S$  be two (not necessarily simple) paths in  $S$  with the same endpoints. A *homotopy* between  $\alpha$  and  $\beta$  is a continuous function  $h: [0, 1]^2 \rightarrow S$  such that  $h(s, 0) = \alpha(0) = \beta(0)$  and  $h(s, 1) = \alpha(1) = \beta(1)$  for all  $s$ , and  $h(0, t) = \alpha(t)$  and  $h(1, t) = \beta(t)$  for all  $t$ . Two paths are *homotopic*, or in the same *homotopy class*, if there is a homotopy between them. A closed curve  $\gamma$  in  $S$  (or a closed walk in a surface graph  $G$ ) is *contractible* if it is homotopic to a constant function. A simple closed curve (or a simple cycle in  $G$ ) is contractible in  $S$  if and only if it is the boundary of a disk in  $S$  [25].

Mirroring classical terminology for curves in the plane [1, 52], an *Alexander numbering* for a surface graph  $G$  is a function  $\alpha: F(G) \rightarrow \mathbb{Z}$  that assigns an integer to each face of  $G$ , such that  $\alpha(f) = 0$  for every boundary face  $f$ . The *boundary*  $\partial\alpha$  of an Alexander numbering  $\alpha$  is a circulation  $\partial\alpha: E(G) \rightarrow \mathbb{Z}$ , defined by setting  $\partial\alpha(e) = \alpha(\text{left}(e)) - \alpha(\text{right}(e))$ . A closed walk is *bounding* if and only if it is an Euler tour of some boundary circulation. Equivalently, a closed walk (or its underlying circulation) is bounding if and only if its *integer* homology class is trivial. (We refer to reader to Hatcher [39], Giblin [35], or Edelsbrunner and Harer [23] for a more through introduction to homology.) On the sphere or the plane, every closed walk is bounding, and every circulation is a boundary circulation; however, these equivalences do not extend to other surfaces. (See Figure 1!) A simple cycle  $\gamma$  in  $G$  is bounding in  $S$  if and only if  $S \setminus \gamma$  is disconnected; simple bounding cycles are often called *separating* cycles.

### 3 Contractible Cycles and Walks

#### 3.1 Simple Cycles

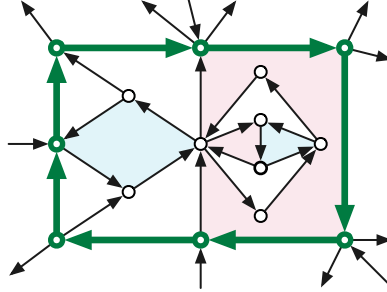
► **Lemma 1.** *Let  $G$  be a directed graph embedded on an orientable surface, and let  $e$  be a directed edge that lies in a directed cycle of  $G$ . No bounding closed walk (and in particular, no simple contractible cycle) in  $G$  traverses  $e$ .*

**Proof.** Let  $\lambda$  be a directed cocycle in  $G$ , and let  $\omega$  be a bounding closed walk. Let  $\phi: E(G) \rightarrow \mathbb{N}$  be the boundary circulation defined by setting  $\phi(e)$  to the number of times that  $\omega$  traverses  $e$ , and let  $\alpha: F(G) \rightarrow \mathbb{N}$  be an Alexander numbering of  $\omega$  (so that  $\phi = \partial\alpha$ ). We immediately have  $\sum_{e \in \lambda} \phi(e) = \sum_{e \in \lambda} (\alpha(\text{right}(e)) - \alpha(\text{left}(e))) = 0$ . Because  $\phi(e) \geq 0$  for every edge  $e$  in  $G$ , it follows that  $\phi(e) = 0$  for every edge  $e \in \lambda$ . ◀

In light of this lemma, we can assume without loss of generality that  $G$  contains no directed cocycles; in particular, no edge of  $G$  has the same face on both sides.

The *boundary* of a face  $f$ , denoted  $\partial f$ , is the set of edges that have  $f$  on one side. The boundary of  $f$  is oriented *clockwise* if  $f$  is the right shore of any edge in  $\partial f$ , and *counterclockwise* if  $f$  is the left shore of any edge in  $\partial f$ . Each face in  $G$  with counterclockwise

boundary appears as a source in the dual graph  $G^*$ ; each face of  $G$  with clockwise boundary appears as a sink in  $G^*$ . We call a face *coherent* if its boundary is oriented either clockwise or counterclockwise, and *incoherent* otherwise. See Figure 3.



■ **Figure 3** A simple clockwise contractible cycle (bold green) in a directed surface graph, enclosing four coherent faces: two counterclockwise (shaded blue) and two clockwise (shaded pink).

► **Lemma 2.** *Let  $G$  be a directed graph with no cocycles, embedded on an orientable surface  $S$ . If  $G$  contains a simple contractible directed cycle, then  $G$  has a face whose boundary is a simple contractible directed cycle.*

**Proof.** Let  $\gamma$  be a simple contractible cycle in  $G$ . This cycle is the boundary of a closed disk  $D$  [25]. Without loss of generality, assume  $\gamma$  is oriented clockwise around  $D$ , so the right shore of every edge in  $\gamma$  is a face in  $D$ .

A dual walk that starts at a face inside  $D$  cannot visit the same face more than once, because  $G$  has no cocycles ( $G^*$  has no cycles), and cannot cross  $\gamma$ , because  $\gamma$  is oriented clockwise (all its dual edges point into  $D$ ). Thus,  $D$  must contain at least one face  $f$  with clockwise boundary (a sink in  $G^*$ ). The closure of  $f$  is a subset of the closed disk  $D$ , so it must be homeomorphic to a closed disk with zero or more interior holes.

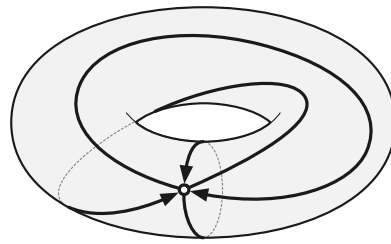
If  $\partial f$  is a simple directed cycle, we are done. Otherwise,  $\partial f$  is the union of two or more edge-disjoint simple directed cycles; consider the non-simple shaded face in Figure 3. Each of these cycles is contractible, because it lies in  $D$ . Any simple cycle in  $\partial f$  that is not the original cycle  $\gamma$  (for example, the boundary of any hole in the closure of  $f$ ) encloses strictly fewer faces than  $\gamma$ . The lemma now follows immediately by induction on the number of enclosed faces. ◀

► **Theorem 3.** *Given an arbitrary directed graph  $G$  embedded on an arbitrary orientable surface, we can determine in  $O(n)$  time whether  $G$  contains a simple contractible cycle.*

**Proof.** The algorithm proceeds as follows. First, we compute the strong components of the dual graph  $G^*$  in  $O(n)$  time. An edge  $e$  of  $G$  lies in a cocycle if and only if both endpoints of the dual edge  $e^*$  lie in the same strong component of  $G^*$ ; we remove all such edges from  $G$  in  $O(n)$  time. Finally, we examine each face of the remaining subgraph by brute force, in  $O(n)$  time. If any face is an open disk whose boundary is a simple directed cycle, we output TRUE; otherwise, we output FALSE. Correctness follows from Lemmas 1 and 2. ◀

### 3.2 Non-simple Closed Walks

It is easy to construct directed surface graphs that contain contractible closed walks but no simple contractible cycles. For example, the one-vertex graph in Figure 4 has two Eulerian circuits, both of which are contractible, but the only simple cycles consist of single edges, none of which are contractible.



■ **Figure 4** A surface graph with contractible closed walks but no simple contractible cycles.

We detect contractible closed walks using essentially the algorithm described in the previous section: After removing all cocycle edges, we look for a face whose boundary is coherent and whose interior is an open disk. However, proving this algorithm correct requires more subtlety. Here we only sketch a proof of correctness, deferring full details to the full version [30, Section 3.2].

First, we observe that some shortest contractible closed walk in  $G$  is *weakly simple*, meaning it can be perturbed in a small neighborhood of the graph into a simple closed curve [15]. Let  $\omega$  be a shortest contractible closed walk in  $G$ , and let  $\tilde{\omega}$  be a generic perturbation of  $\omega$  on the surface. A seminal result of Hass and Scott [38, Theorem 2.7] implies that if  $\tilde{\omega}$  is not already simple, then it contains either a simple contractible closed subpath, which we call a *monogon*, or a pair of simple interior-disjoint subpaths that bound a disk, which we call a *bigon*. We call a bigon *coherent* if one subpath is homotopic to the reverse of the other, and *incoherent* otherwise. See Figure 5. If  $\tilde{\omega}$  contained a monogon or a coherent bigon, smoothing its endpoint(s) would give us shorter contractible closed walk than  $\omega$ , which is impossible. Smoothing the corners of incoherent bigons removes all self-intersections from  $\tilde{\omega}$ ; we can mirror this smoothing by reordering subpaths of the walk  $\omega$ .



■ **Figure 5** Shortening or simplifying a contractible closed curve by smoothing at points of self-intersection. From left to right: A monogon, a coherent bigon, and an incoherent bigon.

A simple winding-number argument implies that any weakly simple *bounding* walk in  $G$  traverses each edge at most once.

Finally, we prove that if a graph  $G$  with no cocycles has a contractible closed walk, then the boundary of some face of  $G$  is contractible, using a similar induction argument as Lemma 2. If some weakly simple contractible closed walk  $\omega$  encloses more than one face, then it must enclose a face  $f$  with coherent boundary (a source or sink in  $G^*$ ). If face  $f$  is a disk, we are done; otherwise,  $f$  is a disk with holes, and the boundary of any hole is another weakly simple contractible closed walk that encloses fewer faces.

► **Theorem 4.** *Given a directed graph  $G$  embedded on any surface, we can determine in  $O(n)$  time whether there is a contractible closed walk in  $G$ .*

If the face  $f$  has an interior hole, then the graph  $G$  *must* be disconnected, because the interior of  $f$  is a subset of the open disk  $A$ . Thus, it may tempting to simplify our argument by assuming “without loss of generality” that the graph  $G$  is connected. Unfortunately, we cannot *simultaneously* assume that  $G$  is connected *and* that  $G$  contains no cocycles. Resolving this tension lies at the core of our algorithm for finding bounding closed walks, which we describe next.



#### 4 Bounding Closed Walks

To simplify our presentation, we assume in this section that our input graphs are embedded on surfaces *without* boundary; this assumption is justified by the following observation. Let  $G$  be a directed graph embedded on a surface  $S$  with  $b > 0$  boundary cycles, and let  $S^\bullet$  be the surface without boundary obtained from  $S$  by attaching a single disk with  $b - 1$  holes to all boundary cycles of  $S$ . All boundary faces of  $G$  on the original surface  $S$  are contained in a single face of  $G$  on  $S^\bullet$ . If  $S$  has first Betti number  $\beta$ , then  $S^\bullet$  has genus  $O(\beta)$ .

► **Lemma 5.** *A closed walk  $\omega$  in  $G$  is bounding in  $S$  if and only if it is bounding in  $S^\bullet$ .*

**Proof.** Any Alexander numbering for  $\omega$  on  $S$  is also an Alexander numbering for  $\omega$  on  $S^\bullet$ .

Let  $\alpha$  be any Alexander numbering for  $\omega$  on  $S^\bullet$ . For any integer  $k$ , the function  $\alpha + k$  defined as  $(\alpha + k)(f) = \alpha(f) + k$  for every face is also an Alexander numbering for  $\omega$  on  $S^\bullet$ . Thus, there is an Alexander numbering  $\alpha^\bullet$  of  $\omega$  such that  $\alpha^\bullet(f) = 0$  for the single face of  $G$  containing  $S^\bullet \setminus S$ . The function  $\alpha^\bullet$  is an Alexander numbering for  $\omega$  on  $S$ . ◀

Our algorithm for detecting bounding closed walks relies on two elementary observations. First, we can assume without loss of generality that  $G$  is strongly connected, because any closed walk in the input graph  $G$  stays within a single strong component of  $G$ . On the other hand, Lemma 1 implies that we can assume without loss of generality that the input graph  $G$  has no cocycles; otherwise, we can remove all cocycles in  $O(n)$  time by contracting each strong component of the dual graph  $G^*$  to a single dual vertex. Indeed, if the input graph satisfies *both* of these conditions, finding bounding closed walks is trivial.

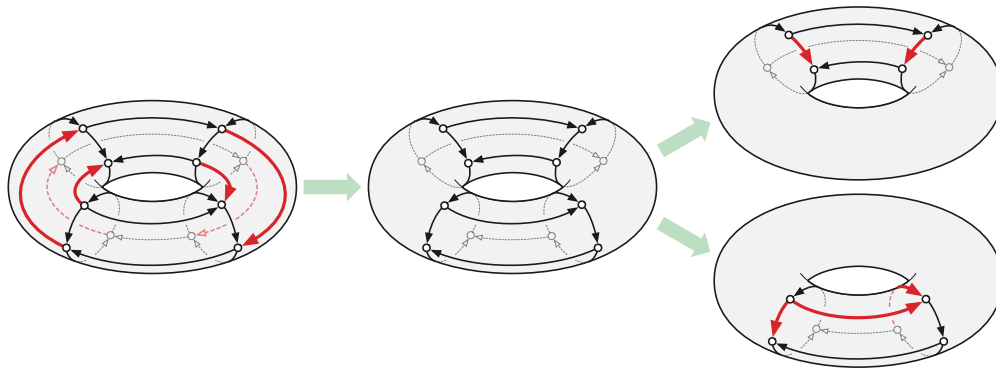
► **Lemma 6.** *Let  $G$  be a strongly connected surface graph with at least one edge, whose dual graph  $G^*$  is acyclic. There is a bounding closed walk in  $G$ .*

**Proof.** For each face  $f$  of  $G$ , let  $\alpha(f)$  be the rank of the dual vertex  $f^*$  in an arbitrary topological sort of the dual graph  $G^*$ . For each edge  $e$  in  $G$ , we have  $\alpha(\text{left}(e)) > \alpha(\text{right}(e))$ . Thus, the function  $\partial\alpha: e \mapsto \alpha(\text{left}(e)) - \alpha(\text{right}(e))$  is a positive integer boundary circulation with connected support. Any Euler tour of this circulation is a bounding closed walk. ◀

Kao and Shannon [41, 43, 44] observed that these two assumptions are actually identical for planar graphs; a planar directed graph  $G$  is strongly connected if and only if its dual graph  $G^*$  is acyclic. (This important observation is the basis of several efficient algorithms for planar directed graphs [2, 41–44, 53].) And indeed, finding bounding closed walks in planar graphs is trivial, because *every* closed walk in a planar graph is bounding.

However, this equivalence does not extend to directed graphs on more complex surfaces; a strongly connected directed surface graph can contain many directed cocycles. Moreover, deleting all the directed cocycles from a strongly connected surface graph can disconnect the graph. More subtly, if  $H$  is a disconnected surface graph without cocycles, the induced embeddings of the individual components of  $H$  *can* contain cocycles. See Figure 6.

Our algorithm repeatedly applies both of these simplifications, alternately removing cocycle edges and separating components, eventually reporting success if it ever finds a component that is both strongly connected and cocycle-free. Each iteration can be performed in  $O(n)$  time, and each iteration removes at least one edge, so conservatively, our algorithm runs in  $O(n^2)$  time. But in fact, our algorithm converges after only  $O(g)$  iterations. We only sketch a proof of this claim in this extended abstract; see the full paper [30, Section 4.1] for complete proofs.



■ **Figure 6** A strongly connected directed graph on the torus with cocycles (bold red). Deleting all cocycles disconnects the graph. The components of the disconnected graph have more cocycles. (Only one cocycle is emphasized in each component.)

► **Lemma 7.** *Let  $H$  be a directed surface graph without cocycles. Every weak component of  $H$  is strongly connected.*

**Proof.** If a weak component of  $H$  is not strongly connected, then  $H$  contains a non-empty directed edge cut. Every non-empty directed edge cut is the disjoint union of cocycles. ◀

We call any face of  $G$  *simple* if it is homeomorphic to an open disk and *non-simple* otherwise. (The boundary of a simple face is *not* necessarily a simple cycle.)

► **Lemma 8.** *Let  $G$  be a strongly connected surface graph, let  $H$  be the subgraph of  $G$  obtained by deleting all cocycles, and let  $G'$  be a (strong) component of  $H$ . Every simple face of  $G'$  is also a simple face of  $G$ .*

► **Corollary 9.** *Let  $G$  be a strongly connected surface graph, let  $H$  be the subgraph of  $G$  obtained by deleting all cocycles, and let  $G'$  be a (strong) component of  $H$ . Every directed cocycle in  $G'$  visits at least one non-simple face of  $G'$ .*

We define the *footprint*  $\llbracket G \rrbracket$  of a directed surface graph  $G$  as the union of the vertices, edges, and *simple* faces of  $G$ . For example, the footprint of the middle graph in Figure 6 is the disjoint union of two annuli. The embedding of  $G$  is cellular if and only if  $\llbracket G \rrbracket$  is the entire surface. Let  $\beta_0(G)$  denote the number of (weak) components of  $G$ , or equivalently, the number of components of its footprint  $\llbracket G \rrbracket$ . Let  $\beta_1(G)$  denote the first Betti number of  $\llbracket G \rrbracket$ , which is the rank of the first homology group of  $\llbracket G \rrbracket$ . If  $G$  has at least one non-simple face, then  $\beta_1(G) := \beta_0(G) - v(G) + e(G) - f_0(G)$ , where  $v(G)$ ,  $e(G)$ , and  $f_0(G)$  denote the number of vertices, edges, and simple faces of  $G$ , respectively. If  $G$  (and therefore  $\llbracket G \rrbracket$ ) is disconnected, then  $\beta_1(G)$  is the sum of the first Betti numbers of its components.

► **Lemma 10.** *Let  $G$  be a strongly connected surface graph, let  $H$  be the subgraph of  $G$  obtained by deleting all cocycles, let  $G'$  be a (strong) component of  $H$ , and let  $H'$  be the subgraph of  $G'$  obtained by deleting all cocycles. If  $H' \neq G'$ , then  $\beta_1(H') < \beta_1(G')$ .*

► **Theorem 11.** *Given a directed graph  $G$  embedded on any surface with genus  $g$ , we can determine in  $O(gn)$  time whether there is a bounding closed walk in  $G$ .*

**Proof.** Assume  $G$  is strongly connected, since otherwise, we can consider each strong component of  $G$  separately. Let  $H$  be the graph obtained by deleting all cocycles of  $G$ . We can construct  $H$  in  $O(n)$  time by computing the strong components of  $G^*$  in linear time, and

then deleting any edge of  $G$  whose incident faces lie in the same strong component of  $G^*$ . Lemma 1 implies that any bounding closed walk in  $G$  is also a bounding closed walk in  $H$ .

If  $H$  has no edges, we return FALSE. If  $H$  is weakly connected and has at least one edge, then, following Lemmas 6 and 7, we immediately return TRUE. Otherwise, we recursively examine each (strong) component of  $H$ . Each vertex of the input graph  $G$  participates in only one subproblem at each level of recursion, so the total time spent at each level is  $O(n)$ . Finally, Lemma 10 implies that the depth of the recursion tree is at most  $O(g)$ . ◀

## 5 Shortest Contractible Closed Walks

Finally, we describe polynomial-time algorithms to compute the shortest contractible closed walk in a directed surface graph with weighted edges. In this extended abstract, we describe only our algorithm for surfaces without boundary and with genus  $g \geq 2$ . These are precisely the surfaces whose fundamental groups are *hyperbolic* in the sense of Gromov [37], but not free; as already observed by Dehn [19], the natural geometry for the universal cover of these surfaces is the hyperbolic plane. (See Figure 8 below.) We describe our (simpler) algorithms for other types of surfaces in the full paper [30, Section 6].

Ultimately, our algorithm reduces the problem of finding the shortest contractible cycle to the *CFG shortest path* problem, introduced by Yannakakis [56]. Given a directed graph  $G = (V, E)$  with weighted edges, a labeling  $\ell: E \rightarrow \Sigma \cup \{\varepsilon\}$  for some finite set  $\Sigma$ , and a context-free grammar  $\mathcal{G}$  over the alphabet  $\Sigma$ , the CFG shortest path problem asks for the shortest walk (if any) in  $G$  whose label is in the language generated by  $\mathcal{G}$ . An algorithm of Barrett et al. [3] solves the CFG shortest path problem in  $O(NPn^3)$  time, where  $N$  and  $P$  are the numbers of nonterminals and productions in  $\mathcal{G}$ , respectively, and  $n$  is the number of vertices in  $G$ , when the grammar  $\mathcal{G}$  is in Chomsky normal form and all edge weights in  $G$  are non-negative. (Their algorithm was extended to graphs with negative edges but no negative cycles by Bradford and Thomas [4].) The main difficulty in our algorithm is the construction of an appropriate context-free grammar.

Fix a directed graph  $G$  with non-negatively weighted edges, embedded on an orientable surface  $S$  with genus  $g \geq 2$  and no boundary. Unlike previous sections, it will prove convenient in this section to treat  $G$  as undirected, but with asymmetric edge weights, as proposed by Cabello et al. [10]. That is, each undirected edge  $uv$  in  $G$  is composed of two *darts*  $u \rightarrow v$  and  $v \rightarrow u$  with independent weights; a walk is alternating series of vertices and *darts*; and the length of a walk is the sum of the weights of its darts, counted with appropriate multiplicity. We also assume without loss of generality that the given embedding of  $G$  is *cellular*, meaning every face is homeomorphic to an open disk. These assumptions can be enforced if necessary by adding at most  $O(m)$  directed edges with (symbolically) infinite weight.

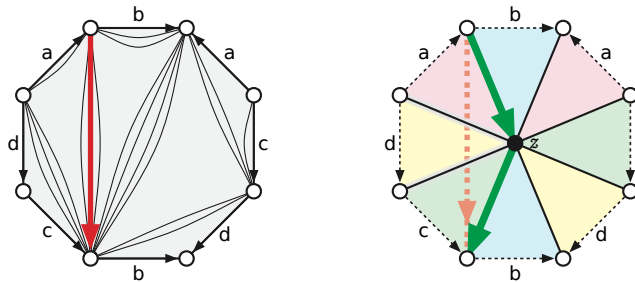
### 5.1 Edge Labeling

The first step in our algorithm is to label the edges of  $G$  so that the labels along any closed walk encode its homotopy type. In principle, we can derive such a labeling by reducing  $G$  to a system of loops [12, 22, 46]; however, this labeling leads to a less efficient algorithm.<sup>1</sup> Instead, we reduce  $G$  to a different canonical surface decomposition called a *system of quads*, as proposed by Lazarus and Rivaud [47] and Erickson and Whittlesey [31].

<sup>1</sup> Specifically, the running time of the resulting algorithm has an extra factor of  $g^{O(g)}$  in the running time.

Let  $(T, L, C)$  be an arbitrary *tree-cotree decomposition* of  $G$ , which partitions the edges of  $G$  into a spanning tree  $T$ , a spanning tree  $C^*$  of the dual graph  $G^*$ , and exactly  $2g$  leftover edges  $L$  [24]. Contracting every edge in  $T$  and deleting every edge in  $C$  reduces  $G$  to a *system of loops*, which has one vertex  $a$ , one face  $f$ , and  $2g$  loops  $L$ . To construct the system of quads  $Q$ , we introduce a new vertex  $z$  in the interior of  $f$ , add edges between  $z$  and every corner of  $f$ , and then delete the edges in  $L$ .

Next we label each directed edge  $e$  in  $G$  with a directed walk  $\langle e \rangle$  in  $Q$  as follows. Every walk  $\langle e \rangle$  starts and ends at  $a$  and thus has even length. If  $e \in T$ , then  $\langle e \rangle$  is the empty walk. Otherwise, after contracting  $T$ , edge  $e$  connects two corners of  $f$ ; we define  $\langle e \rangle$  as the walk of length 2 in  $Q$  from the back corner of  $e$ , to  $z$ , and then to the front corner of  $e$ . (If  $e \in L$ , there are two such walks; choose one arbitrarily.)



■ **Figure 7** Left: A fundamental polygon for a surface of genus 2, obtained by contracting a spanning tree and cutting along a system of loops. Right: Mapping a directed non-tree edge to a path of length 2 in the corresponding system of quads.

Finally, for any closed walk  $\omega$  in  $G$ , let  $\langle \omega \rangle$  denote the closed walk in  $Q$  obtained by concatenating of the labels of edges in  $\omega$  in order. The closed walks  $\omega$  and  $\langle \omega \rangle$  are freely homotopic in  $S$ ; in particular,  $\omega$  is contractible if and only if  $\langle \omega \rangle$  is contractible. Moreover, the number of edges in  $\langle \omega \rangle$  is at most twice the number of edges in  $\omega$ .

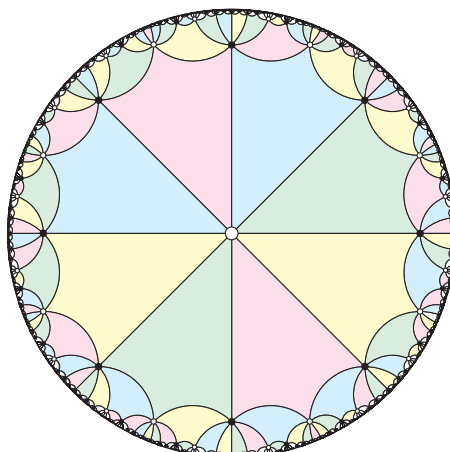
The universal cover  $\widehat{Q}$  of a system of quads  $Q$  is isomorphic to a regular tiling of the hyperbolic plane by quadrilaterals meeting at vertices of degree  $4g$ ; see Figure 8. The covering map from  $\widehat{Q}$  to  $Q$  is a graph homomorphism (mapping vertices to vertices and edges to edges), which (at the risk of confusing the reader) we also denote  $\langle \cdot \rangle: \widehat{Q} \rightarrow Q$ . A closed walk in  $Q$  is contractible if and only if it is the projection of a closed walk in  $\widehat{Q}$ . Thus, a closed walk  $\omega$  in  $G$  is contractible if and only if there is a closed walk  $\widehat{\omega}$  in  $\widehat{Q}$  such that  $\langle \widehat{\omega} \rangle = \langle \omega \rangle$ .

## 5.2 Hyperbolic Geometry

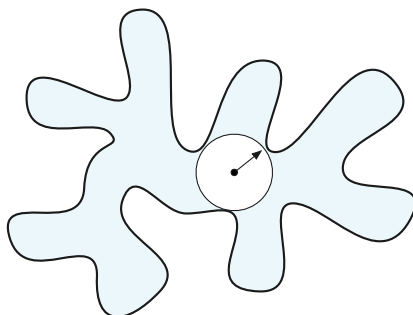
Our algorithm exploits two well-known geometric properties of regular hyperbolic tilings: the area of any ball in a hyperbolic tiling grows exponentially with its radius [36, 48], and the area enclosed by any simple cycle grows at most linearly with the cycle’s length [20, 21]. In particular, we rely on the following immediate corollary of these two properties: The distance from any point in the interior of a simple cycle in a regular hyperbolic tiling is at most *logarithmic* in the length of that cycle; see Figure 9.

Classical results already imply this *qualitative* behavior for *all* regular hyperbolic tilings; indeed, Chepoi et al. [16] have derived tight *asymptotic* bounds. However, we require *precise* upper bounds for growth, isoperimetry, and in-radius in the specific tiling  $\widehat{Q}$ ; the running time of our algorithm ultimately depends *exponentially* on the constants in those bounds.

For any positive integer  $r$ , let  $N(r)$  denote the number of vertices at distance at most  $r$  from an arbitrary *basepoint* vertex  $\widehat{a}$  of  $\widehat{Q}$ ; we derive tight bounds on  $N(r)$  from a recurrence



■ **Figure 8** The universal cover of a genus-2 system of quads. (Compare with Figure 7.)



■ **Figure 9** Simple closed curves in hyperbolic tilings have logarithmic in-radius.

of Floyd and Plotnick [32]. We derive our isoperimetric inequalities using the “spur and bracket” analysis of Erickson and Whittlesey [31, Section 4.2], which is based in turn on results of Gersten and Short [34], along with classical results of Dehn [20, 21]. Our inradius bound then follows immediately. See the full paper [30, Section 7.2] for further details.

► **Lemma 12.**  $N(r) \sim \beta \lambda^r$  for some constants  $1 < \beta < 2$  and  $4g - 3 < \lambda < 4g - 2$ . In particular,  $N(r) \geq \lambda^r$  for all  $r \geq 0$ .

► **Lemma 13.** Any simple cycle of length  $L$  in  $\widehat{Q}$  has less than  $3L/2$  faces of  $\widehat{Q}$  in its interior.

► **Lemma 14.** Any simple cycle of length  $L$  in  $\widehat{Q}$  has less than  $2L/g$  vertices of  $\widehat{Q}$  in its interior.

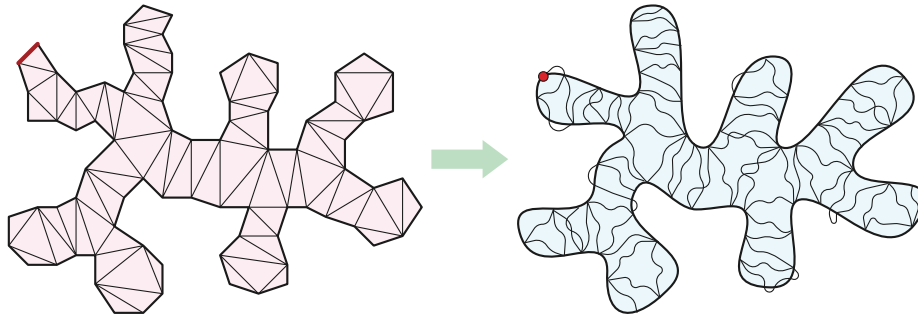
► **Lemma 15.** Let  $\gamma$  be a simple cycle in  $\widehat{Q}$  with length at most  $L$ . Every tiling vertex in the interior of  $\gamma$  has distance at most  $\rho = \lceil \log_\lambda(2L/g) \rceil$  from at least one vertex of  $\gamma$ .

### 5.3 Triangulation

Now let  $\omega$  be a closed walk in  $\widehat{Q}$  with a distinguished vertex  $\hat{a}$  called its *basepoint*. Following Muller and Schupp [51], define a  $k$ -*triangulation* of  $\omega$  is a continuous map  $\mathcal{T}$  from a *reference* triangulation  $T$  of a simple polygon  $P$  to  $\widehat{Q}$  with the following properties:

- $\mathcal{T}$  maps vertices of  $P$  to vertices of  $\omega$ .
- $\mathcal{T}$  maps a distinguished edge of  $P$ , called the *root edge*, to the basepoint  $\hat{a}$ .

- $\mathcal{T}$  maps the boundary of  $P$  continuously onto the walk  $\omega$ ; in particular, every edge of  $P$  except the root edge is mapped to a single edge of  $\omega$ .
  - Finally,  $\mathcal{T}$  maps each diagonal edge in  $T$  to a walk of length at most  $k$  in  $\widehat{Q}$ .
- This map is not necessarily an embedding, even locally. The diagonal walks may intersect each other or  $\omega$ , even if  $\omega$  is a simple cycle; moreover, some diagonal walks may have length zero if  $\omega$  is not simple.

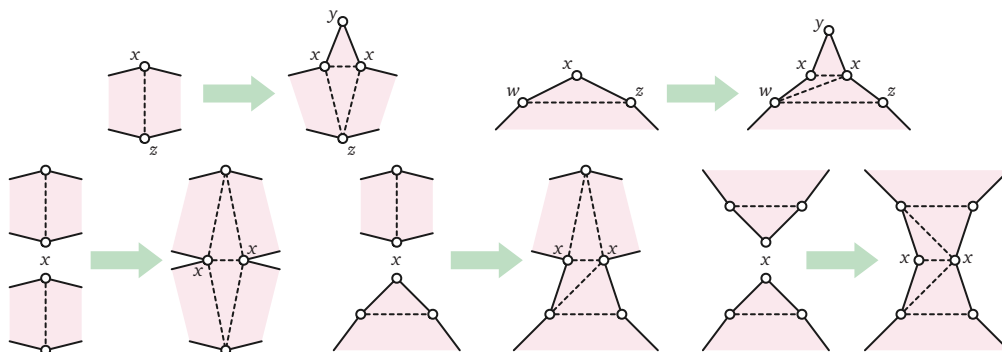


■ **Figure 10** Any closed walk in a hyperbolic tiling can be “triangulated” by paths of logarithmic length.

► **Lemma 16.** *Every simple cycle of length  $L$  in  $\widehat{Q}$  has a  $(2\rho + 2)$ -triangulation, where  $\rho = \lceil \log_\lambda(2L/g) \rceil$ .*

**Proof.** Let  $\gamma$  be a simple cycle of length  $L$  in  $\widehat{Q}$ , and let  $\Gamma$  denote the closed disk bounded by  $\gamma$ . Intuitively, we construct a “Delaunay triangulation” of the vertices of  $\gamma$  inside  $\Gamma$ .

For each tiling vertex  $x$  in  $\Gamma$ , let  $nn(x)$  denote its nearest neighbor on  $\gamma$ , measuring distance by counting edges in  $\widehat{Q}$  and breaking ties arbitrarily. In particular, if  $x \in \gamma$ , then  $nn(x) = x$ . Lemma 15 implies that the distance from any vertex  $x$  to its nearest neighbor  $nn(x)$  is at most  $\rho$ . Arbitrarily triangulate each tile in  $\Gamma$  to obtain a triangulation  $\Delta$ . Let  $M$  denote the set of all edges  $xy$  in  $\Delta$  such that  $nn(x) \neq nn(y)$ . (Intuitively,  $M$  is a “discrete medial axis” of  $\Gamma$  [5].) For each edge  $xy$  in  $M$ , let  $\sigma(x, y)$  denote a shortest path in  $\widehat{Q}$  from  $nn(x)$  to  $nn(y)$ ; in particular, if  $xy$  is an edge of  $\gamma$ , then  $\sigma(x, y) = xy$ . The triangle inequality implies that each shortest path  $\sigma(x, y)$  has length at most  $2\rho + 2$ . The shortest paths  $\{\sigma(x, y) \mid xy \in M\}$  are the paths of a  $(2\rho + 2)$ -triangulation of  $\gamma$ . ◀



■ **Figure 11** Top: Extending a  $k$ -triangulation across a spur. Bottom: Merging two  $k$ -triangulations across a generic repeated vertex.

We extend the previous construction to non-simple closed walks using a straightforward induction argument. Any non-simple closed walk can be decomposed into two shorter closed walks at a repeated vertex. The induction hypothesis gives us two  $(2\rho + 2)$ -triangulations for the shorter closed walks, which we can extend or combine into a  $(2\rho + 2)$ -triangulation for  $\omega$  by case analysis, as suggested by Figure 11. We defer further details to the full paper [30, Section 7.3].

► **Lemma 17.** *Every closed walk of length  $L$  in  $\widehat{Q}$  has a  $(2\rho + 2)$ -triangulation, where  $\rho = \lceil \log_\lambda(2L/g) \rceil$ .*

## 5.4 Context-Free Grammar

Now finally we reach the key component of our algorithm: the construction of a small context-free grammar  $\mathcal{G}$  in Chomsky normal form such that the shortest contractible closed walk in  $G$  is also the shortest walk whose label is in the language generated by  $\mathcal{G}$ .

► **Lemma 18.** *For any positive integer  $L$ , there is a context-free grammar  $\mathcal{G}$  in Chomsky normal form with  $O(g^2L^2)$  non-terminals, such that (1) Every string generated by  $\mathcal{G}$  describes a contractible closed walk in  $Q$ , and (2) every contractible closed walk in  $Q$  of length at most  $L$  is generated by  $\mathcal{G}$ .*

**Proof sketch.** We closely follow a construction of Muller and Schupp [51, Theorem 1]. The alphabet for our grammar consists of all  $8g$  darts in  $Q$ . The variables of  $\mathcal{G}$  correspond to homotopy classes of walks of length at most  $2\rho + 2$  in  $Q$ ; in particular, the starting variable corresponds to the class of contractible walks from vertex  $a$  to vertex  $a$ . The number of nonterminals is

$$2N(2\rho + 2) = O(\lambda^{2\rho+2}) \leq O(\lambda^{2\log_\lambda(2L/g)+4}) = O((2L/g)^2 \lambda^4) = O(g^2L^2)$$

by Lemma 12 and the definition of  $\rho$ . The non-terminal productions in  $\mathcal{G}$  correspond to concatenation of homotopy classes; crudely, there are at most  $O(g^4L^4)$  such productions. Finally, the grammar includes a terminal production for the homotopy class of each dart. For any closed walk  $\omega$  of length  $L$ , and for any  $(2\rho+2)$ -triangulation  $\mathcal{T}$  of  $\omega$ , the dual tree of  $\mathcal{T}$  is a parse tree for  $\omega$  in this grammar. We defer further details to the full paper [30, Section 7.4]. ◀

► **Theorem 19.** *Let  $G$  be a directed graph with  $m$  non-negatively weighted edges, embedded on the orientable surface of genus  $g$  with no boundary. We can compute the shortest contractible closed walk in  $G$  in  $O(g^6m^9)$  time.*

**Proof.** We construct the system of quads  $Q$  and label the edges of  $G$  in  $O(m)$  time. If necessary, subdivide edges of  $G$  so that each edge is labeled with either the empty walk or a single dart in  $Q$ . After subdivision,  $G$  has at most  $2m$  edges, and therefore (because  $G$  is symmetric) at most  $2m$  vertices.

Then we build a context-free grammar  $\mathcal{G}$  that generates only contractible walks in  $Q$  and generates all non-empty contractible walks in  $Q$  of length at most  $2m$ , as described in Lemma 18. This grammar has  $N = O(g^2m^2)$  nonterminals and  $P = O(g^4m^4)$  productions. Recall from Section 3.2 that some shortest contractible walk in  $G$  is weakly simple, and therefore has at most  $2m$  edges. It follows that  $\mathcal{G}$  generates the label of some shortest contractible walk in  $G$ .

Finally, we compute the shortest closed walk in  $G$  whose label is generated by  $\mathcal{G}$ , using the CFG-shortest-path algorithm of Barrett et al. [3], in  $O(NP(n')^3) = O(g^6m^9)$  time. ◀



## References

- 1 James W. Alexander. Topological invariants of knots and links. *Trans. Amer. Math. Soc.*, 30(2):275–306, 1928. doi:10.1090/S0002-9947-1928-1501429-1.
- 2 Lars Arge, Laura Toma, and Norbert Zeh. I/O-efficient topological sorting of planar DAGs. In *Proc. 15th Ann. ACM Symp. Parallel Algorithms Arch.*, pages 85–93, 2003. doi:10.1145/777412.777427.
- 3 Chris Barrett, Riko Jacob, and Madhav Marathe. Formal-language-constrained path problems. *SIAM J. Comput.*, 30(3):809–837, 2000. doi:10.1137/S0097539798337716.
- 4 Phillip G. Bradford and David A. Thomas. Labeled shortest paths in digraphs with negative and positive edge weights. *RAIRO-Theor. Inf. Appl.*, 43(3):567–583, 2009. doi:10.1051/ita/2009011.
- 5 Jonathan W. Brandt. Convergence and continuity criteria for discrete approximations of the continuous planar skeletons. *CVGIP: Image Understanding*, 59(1):116–124, 1994. doi:10.1006/ciun.1994.1007.
- 6 Sergio Cabello. Finding shortest contractible and shortest separating cycles in embedded graphs. *ACM Trans. Algorithms*, 6(2):24:1–24:18, 2010. doi:10.1145/1721837.1721840.
- 7 Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-Source Shortest Paths in Embedded Graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013. doi:10.1137/120864271.
- 8 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proc. 26th Ann. Symp. Comput. Geom.*, pages 156–165, 2010. doi:10.1145/1810959.1810988.
- 9 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding cycles with topological properties in embedded graphs. *SIAM J. Discrete Math.*, 25:1600–1614, 2011. doi:10.1137/100810794.
- 10 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. *J. Comput. Geom.*, 7(1):123–148, 2016. doi:10.20382/jocg.v7i1a7.
- 11 Sergio Cabello, Matt DeVos, Jeff Erickson, and Bojan Mohar. Finding one tight cycle. *ACM Trans. Algorithms*, 6(4):article 61, 2010. doi:10.1145/1824777.1824781.
- 12 Sergio Cabello and Bojan Mohar. Finding Shortest Non-Separating and Non-Contractible Cycles for Topologically Embedded Graphs. *Discrete Comput. Geom.*, 37(2):213–235, 2007. doi:10.1007/s00454-006-1292-5.
- 13 Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom. Theory Appl.*, 41(1–2):94–110, 2008. doi:10.1016/j.comgeo.2007.10.010.
- 14 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In *Proc. 25th Ann. Symp. Comput. Geom.*, pages 377–385, 2009. doi:10.1145/1542362.1542426.
- 15 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proc. 26th ACM-SIAM Symp. Discrete Algorithms*, pages 1655–1670, 2015. doi:10.1137/1.9781611973730.110.
- 16 Victor Chepoi, Feodor Dragan, Bertrand Estellon, Michel Habib, and Yann Vaxès. Diameters, centers, and approximating trees of  $\delta$ -hyperbolic geodesic spaces and graphs. In *Proc. 24th Ann. Symp. Comput. Geom.*, pages 59–68, 2008. doi:10.1145/1377676.1377687.
- 17 Edith Cohen and Nimrod Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in periodic graphs. *J. Assoc. Comput. Mach.*, 40(4):791–830, 1993. doi:10.1145/153724.153727.
- 18 Éric Colin de Verdière and Jeff Erickson. Tightening non-simple paths and cycles on surfaces. *SIAM J. Comput.*, 39(8):3784–3813, 2010. doi:10.1137/090761653.
- 19 Max Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, 71(1):116–144, 1911. URL: <http://eudml.org/doc/158521>.



- 20 Max Dehn. Transformation der Kurven auf zweiseitigen Flächen. *Math. Ann.*, 72(3):413–421, 1912. doi:10.1007/BF01456725.
- 21 Max Dehn. *Papers on Group Theory and Topology*. Springer, 1987. Translated by John Stillwell. doi:10.1007/978-1-4612-4668-8.
- 22 Tamal K. Dey and Sumanta Guha. Transforming Curves on Surfaces. *J. Comput. System Sci.*, 58:297–325, 1999. doi:10.1006/jcss.1998.1619.
- 23 Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., 2010.
- 24 David Eppstein. Dynamic generators of topologically embedded graphs. In *Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 599–608, 2003.
- 25 David B. A. Epstein. Curves on 2-manifolds and isotopies. *Acta Mathematica*, 115:83–107, 1966. doi:10.1007/BF02392203.
- 26 Jeff Erickson. Shortest Non-trivial Cycles in Directed Surface Graphs. In *Proc. 27th Ann. Symp. Comput. Geom.*, pages 236–243, 2011. doi:10.1145/1998196.1998231.
- 27 Jeff Erickson, Kyle Fox, and Luvsandondov Lkhamsuren. Holiest minimum-cost paths and flows in surface graphs. In *Proc. 50th Ann. ACM Symp. Theory Comput.*, pages 1319–1332, 2018. doi:10.1145/3188745.3188904.
- 28 Jeff Erickson and Sarel Har-Peled. Optimally cutting a surface into a disk. *Discrete Comput. Geom.*, 31(1):37–59, 2004. doi:10.1007/s00454-003-2948-z.
- 29 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proc. 22nd Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1166–1176, 2011. doi:10.1137/1.9781611973082.88.
- 30 Jeff Erickson and Yipu Wang. Topologically Trivial Closed Walks in Directed Surface Graphs. Preprint, March 2019. arXiv:1812.01564.
- 31 Jeff Erickson and Kim Whittlesey. Transforming Curves on Surfaces redux. In *Proc. 24th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 1646–1655, 2013. doi:10.1137/1.9781611973105.118.
- 32 William J. Floyd and Steven J. Plotnick. Growth functions on Fuchsian groups and the Euler characteristic. *Invent. Math.*, 88(1):1–29, 1987. doi:10.1007/BF01405088.
- 33 Kyle Fox. Shortest Non-trivial Cycles in Directed and Undirected Surface Graphs. In *Proc. 24th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 352–364, 2013. doi:10.1137/1.9781611973105.26.
- 34 Steve M. Gersten and Hamish B. Short. Small cancellation theory and automatic groups. *Invent. Math.*, 102:305–334, 1990. doi:10.1007/BF01233430.
- 35 Peter Gublin. *Graphs, Surfaces and Homology*. Cambridge Univ. Press, 3rd edition, 2010.
- 36 Rostislav Grigorchuk and Pierre de la Harpe. On problems related to growth, entropy and spectrum in group theory. *J. Dynam. Control Systems*, 3(1):51–89, 1997. doi:10.1007/BF02471762.
- 37 Mikhail Gromov. Hyperbolic Groups. In Steve M. Gersten, editor, *Essays in Group Theory*, number 8 in Math. Sci. Res. Inst. Pub., pages 75–265. Springer-Verlag, 1987.
- 38 Joel Hass and Peter Scott. Intersections of curves on surfaces. *Israel J. Math.*, 51:90–120, 1985. doi:10.1007/BF02772960.
- 39 Allen Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2002. URL: <http://www.math.cornell.edu/~hatcher/AT/ATpage.html>.
- 40 Kazuo Iwano and Kenneth Steiglitz. A semiring on convex polygons and zero-sum cycle problems. *SIAM J. Comput.*, 19(5):883–901, 1990. doi:10.1137/0219061.
- 41 Ming-Yang Kao. Linear-processor NC algorithms for planar directed graphs I: Strongly connected components. *SIAM J. Comput.*, 22(3):431–459, 1993. doi:10.1137/0222032.
- 42 Ming-Yang Kao and Philip N. Klein. Toward overcoming the transitive-closure bottleneck: Efficient parallel algorithms for planar digraphs. *J. Comput. Syst. Sci.*, 47(3):459–500, 1993. doi:10.1016/0022-0000(93)90042-U.

- 43 Ming-Yang Kao and Gregory E. Shannon. Local reorientation, global order, and planar topology. In *Proc. 21st Ann. ACM Symp. Theory Comput.*, pages 286–296, 1989. doi:10.1145/73007.73034.
- 44 Ming-Yang Kao and Gregory E. Shannon. Linear-processor NC algorithms for planar directed graphs II: Directed spanning trees. *SIAM J. Comput.*, 22(3):460–481, 1993. doi:10.1137/0222033.
- 45 S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In *Proc. 20th ACM Symp. Theory. Comput.*, pages 398–406, 1988. doi:10.1145/62212.62251.
- 46 Martin Kutz. Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time. In *Proc. 22nd Ann. Symp. Comput. Geom.*, pages 430–438, 2006. doi:10.1145/1137856.1137919.
- 47 Francis Lazarus and Julien Rivaud. On the homotopy test on surfaces. In *Proc. 53rd Ann. IEEE Symp. Foundations Comput. Sci.*, pages 440–449, 2012. doi:10.1109/FOCS.2012.12.
- 48 John Milnor. A note on curvature and the fundamental group. *J. Diff. Geom.*, 2(1):1–7, 1968. doi:10.4310/jdg/1214501132.
- 49 Bojan Mohar and Carsten Thomassen. *Graphs on Surfaces*. Johns Hopkins Univ. Press, 2001.
- 50 Shay Mozes, Kirill Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum Cut of Directed Planar Graphs in  $O(n \log \log n)$  Time. In *Proc. 29th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 477–494, 2018. doi:10.1137/1.9781611975031.32.
- 51 David E. Muller and Paul E. Schupp. Groups, the theory of ends, and context-free languages. *J. Comput. System Sci.*, 26(3):295–310, 1983. doi:10.1016/0022-0000(83)90003-X.
- 52 August F. Möbius. Über der Bestimmung des Inhaltes eines Polyeders. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl.*, 17:31–68, 1865. *Gesammelte Werke* 2:473–512, Leipzig, 1886.
- 53 Vijaya Ramachandran and Honghua Yang. Finding the closed partition of a planar graph. *Algorithmica*, 11(5):443–468, 1994. doi:10.1007/BF01293266.
- 54 L. B. Smikun. Connection between context-free groups and groups with decidable problems of automata equivalence. *Cybernetics*, 12(5):687–691, 1976. Translated from *Kibernetika (Kiev)* (5):33–37, 1976. doi:10.1007/BF01070252.
- 55 Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *J. Comb. Theory Ser. B*, 48(2):155–177, 1990. doi:10.1016/0095-8956(90)90115-G.
- 56 Mihalis Yannakakis. Graph-theoretic methods in database theory. In *Proc. 9th ACM SIGACT-SIGMOD-SIGART Symp. Principles Database Syst.*, pages 230–242, 1990. doi:10.1145/298514.298576.



# Packing Disks into Disks with Optimal Worst-Case Density

Sándor P. Fekete 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
s.fekete@tu-bs.de

Phillip Keldenich 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
p.keldenich@tu-bs.de

Christian Scheffer 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
c.scheffer@tu-bs.de

---

## Abstract

We provide a tight result for a fundamental problem arising from packing disks into a circular container: The critical density of packing disks in a disk is 0.5. This implies that any set of (not necessarily equal) disks of total area  $\delta \leq 1/2$  can always be packed into a disk of area 1; on the other hand, for any  $\varepsilon > 0$  there are sets of disks of area  $1/2 + \varepsilon$  that cannot be packed. The proof uses a careful manual analysis, complemented by a minor automatic part that is based on interval arithmetic. Beyond the basic mathematical importance, our result is also useful as a blackbox lemma for the analysis of recursive packing algorithms.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems; Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Disk packing, packing density, tight worst-case bound, interval arithmetic, approximation

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.35

**Related Version** A full version of this paper can be found at <http://arxiv.org/abs/1903.07908> [3].

**Supplement Material** <https://github.com/phillip-keldenich/circlepacking>

**Funding** *Phillip Keldenich*: Supported by the German Research Foundation under Grant No. FE 407/17-2.

**Acknowledgements** We thank Sebastian Morr for joint previous work.

## 1 Introduction

Deciding whether a set of disks can be packed into a given container is a fundamental geometric optimization problem that has attracted considerable attention. Disk packing also has numerous applications in engineering, science, operational research and everyday life, e.g., for the design of digital modulation schemes [24], packaging cylinders [1, 10], bundling tubes or cables [29, 27], the cutting industry [28], or the layout of control panels [1], or radio tower placement [28]. Further applications stem from chemistry [30], forestry [28], and origami design [16].

Like many other packing problems, disk packing is typically quite difficult; what is more, the combinatorial hardness is compounded by the geometric complications of dealing with irrational coordinates that arise when packing circular objects. This is reflected by the limitations of provably optimal results for the optimal value for the smallest sufficient disk container (and hence, the densest such disk packing in a disk container), a problem that was



© Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer;  
licensed under Creative Commons License CC-BY

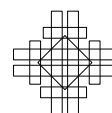
35th International Symposium on Computational Geometry (SoCG 2019).

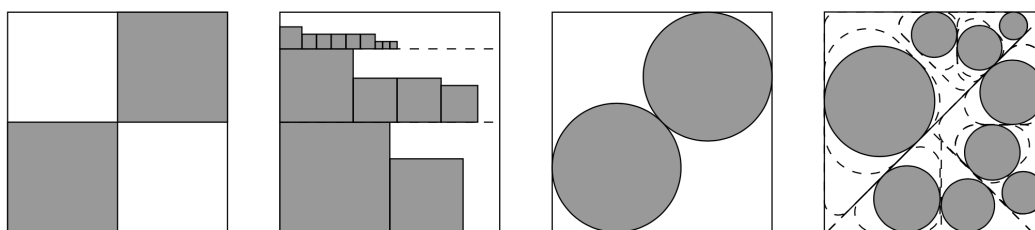
Editors: Gill Barequet and Yusu Wang; Article No. 35; pp. 35:1–35:19

Leibniz International Proceedings in Informatics



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





■ **Figure 1** (1) An instance of critical density for packing squares into a square. (2) An example packing produced by Moon and Moser’s shelf-packing. (3) An instance of critical density for packing disks into a square. (4) An example packing produced by Morr’s Split Packing.

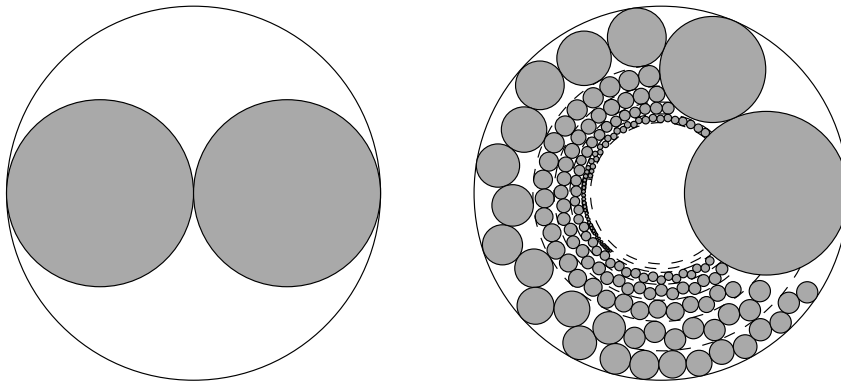
discussed by Kraviz [15] in 1967: Even when the input consists of just 13 unit disks, the optimal value for the densest disk-in-disk packing was only established in 2003 [9], while the optimal value for 14 unit disks is still unproven. The enormous challenges of establishing densest disk packings are also illustrated by a long-standing open conjecture by Erdős and Oler from 1961 [23] regarding optimal packings of  $n$  unit disks into an equilateral triangle, which has only been proven up to  $n = 15$ . For other examples of mathematical work on densely packing relatively small numbers of identical disks, see [11, 19, 7, 8], and [25, 18, 12] for related experimental work. Many authors have considered heuristics for circle packing problems, see [28, 13] for overviews of numerous heuristics and optimization methods. The best known solutions for packing equal disks into squares, triangles and other shapes are continuously published on Specht’s website <http://packomania.com> [26].

For the case of packing not necessarily equal disks into a square container, Demaine, Fekete, and Lang in 2010 [2] showed that deciding whether a given set of disks can be packed is NP-hard by using a reduction from 3-PARTITION. This means that there is (probably) no deterministic polynomial-time algorithm that can decide whether a given set of disks can be packed into a given container.

On the other hand, the literature on exact approximation algorithms which actually give performance guarantees is small. Miyazawa et al. [20] devised asymptotic polynomial-time approximation schemes for packing disks into the smallest number of unit square bins. More recently, Hokama, Miyazawa, and Schouery [14] developed a bounded-space competitive algorithm for the online version of that problem.

The related problem of packing square objects has also been studied for a long time. The decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly NP-complete by Leung et al. [17], also using a reduction from 3-PARTITION. Already in 1967, Moon and Moser [21] found a sufficient condition. They proved that it is possible to pack a set of squares into the unit square in a shelf-like manner if their combined area, the sum of all squares’ areas, does not exceed  $\frac{1}{2}$ . At the same time,  $\frac{1}{2}$  is the *largest upper area bound* one can hope for, because two squares larger than the quarter-squares shown in Figure 1 cannot be packed. We call the ratio between the largest combined object area that can always be packed and the area of the container the problem’s *critical density*, or *optimal worst-case density*.

The equivalent problem of establishing the critical packing density for disks in a square was posed by Demaine, Fekete, and Lang [2] and resolved by Morr, Fekete and Scheffer [22, 4]. Making use of a recursive procedure for cutting the container into triangular pieces, they proved that the critical packing density of disks in a square is  $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$ .



■ **Figure 2** (1) A critical instance that allows a packing density no better than  $\frac{1}{2}$ . (2) An example packing produced by our algorithm.

It is quite natural to consider the analogous question of establishing the critical packing density for disks in a disk. However, the shelf-packing approach of Moon and Moser [21] uses the fact that rectangular shapes of the packed objects fit well into parallel shelves, which is not the case for disks; on the other hand, the split packing method of Morr et al. [22, 4] relies on recursively splitting triangular containers, so it does not work for a circular container that cannot be partitioned into smaller circular pieces.

Note that the main objective of this line of work is to compute tight worst-case bounds. For specific instances, a packing may still be possible, even if the density is higher; this also implies that proofs of infeasibility for specific instances may be trickier. However, the idea of using the total item volume for computing packing bounds can still be applied. See the work by Fekete and Schepers [5, 6], which shows how classes of functions called *dual-feasible* can be used to compute a *modified* volume for geometric objects, yielding good lower bounds for one- or higher-dimensional scenarios.

## 1.1 Results

We prove that the critical density for packing disks into a disk is  $1/2$ : Any set of not necessarily equal disks with a combined area of not more than half the area of a circular container can be packed; this is best possibly, as for any  $\varepsilon > 0$  there are instances of total area  $1/2 + \varepsilon$  that cannot be packed. See Fig. 2 for the critical configuration.

Our proofs are constructive, so they can also be used as a constant-factor approximation algorithm for the smallest-area container of a given shape in which a given set of disks can be packed. Due to the higher geometric difficulty of fitting together circular objects, the involved methods are considerably more complex than those for square containers. We make up for this difficulty by developing more intricate recursive arguments, including appropriate and powerful tools based on *interval arithmetic*.

## 2 Preliminaries

Let  $r_1, \dots, r_n$  be a set of disks in the plane. Two point sets  $A, B \subset \mathbb{R}^2$  *overlap* if their interiors have a point in common. A *container disk*  $\mathcal{C}$  is a disk that may overlap with disks from  $\{r_1, \dots, r_n\}$ . The *original* container disk  $O$  is the unit disk. Due to recursive calls of our algorithm there may be several container disks that lie nested inside each other. Hence, the largest container disk will be the unit disk  $O$ . For simplification, we simultaneously denote by  $r_i$  or  $\mathcal{C}$  the disk with radius  $r_i$  or  $\mathcal{C}$  and its radius. W.l.o.g., we assume  $r_1 \geq \dots \geq r_n$ . We

pack the disks  $r_1, \dots, r_n$  by positioning their centers inside a container disk such that  $r_i$  lies inside  $C$  and two disks from  $\{r_1, \dots, r_n\}$  do not overlap. Given two sets  $A \subseteq B \subseteq \mathbb{R}^2$ , we say that  $A$  is a *sector* of  $B$ . Furthermore, we denote the volume of a point set  $A$  by  $|A|$ .

### 3 A Worst-Case Optimal Algorithm

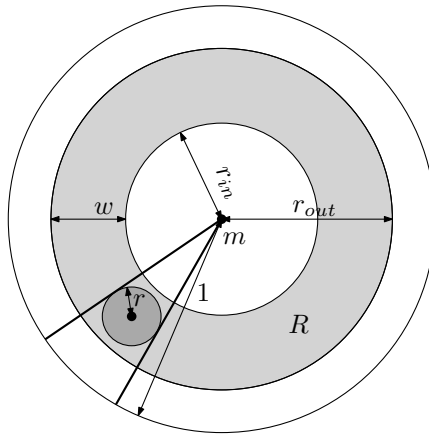
► **Theorem 1.** *Every set of disks with total area  $\frac{\pi}{2}$  can be packed into the unit disk  $O$  with radius 1. This induces a worst-case optimal packing density of  $\frac{1}{2}$ , i.e., a ratio of  $\frac{1}{2}$  between the area of the unit disk and the total area to be packed.*

The worst case consists of two disks  $D_1, D_2$  with radius  $\frac{1}{2}$ , see Fig. 2. The total area of these two disks is  $\frac{\pi}{4} + \frac{\pi}{4} = \frac{\pi}{2}$ , while the smallest disk containing  $D_1, D_2$  has an area of  $\pi$ .

In the remainder of Section 3, we give a constructive proof for Theorem 1. Before we proceed to describe our algorithm in Section 3.4, we give some definitions and describe *Boundary Packing* and *Ring Packing* as two subroutines of our algorithm.

#### 3.1 Preliminaries for the Algorithm

We make use of the following definitions, see Fig. 3.



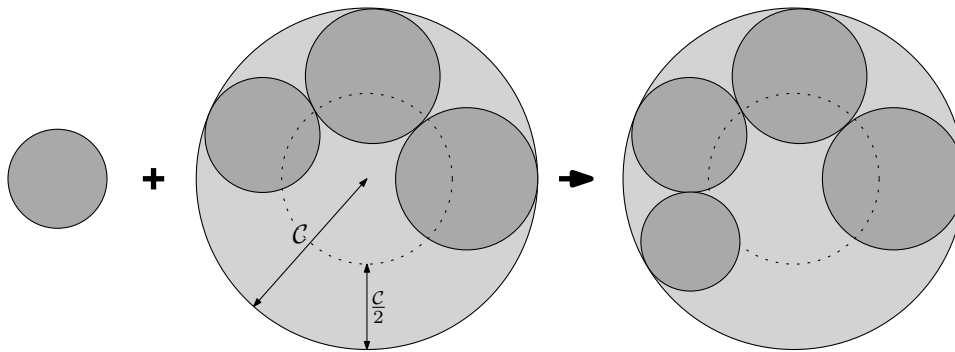
■ **Figure 3** A ring  $R \subset O$  with width  $w$  and a disk with its corresponding tangents.

For  $r_{out} > r_{in} > 0$  and a container disk  $C$  such that  $r_{out} \leq 2r_{in}$ , we define a *ring*  $R := R[r_{out}, r_{in}]$  of  $C$  as the closure of  $r_{out} \setminus r_{in}$ , see Fig. 3. The boundary of  $R$  consists of two connected components. The *inner boundary* is the component lying closer to the center  $m$  of  $r_{out}$  and the *outer boundary* is the other component. The *inner radius* and the *outer radius* of  $R$  are the radius of the inner boundary and the radius of outer boundary. Each ring is associated with one of three states  $\{\text{OPEN}, \text{CLOSED}, \text{FULL}\}$ . Initially, each ring is OPEN.

Let  $r$  be a disk inside a container disk  $C$ . The two *tangents* of  $r$  are the two rays starting in the midpoint of  $C$  and touching the boundary of  $r$ . We say that a disk lies *adjacent* to  $r_{out}$  when the disk is touching the boundary of  $r_{out}$  from the inside of  $r_{out}$ .

#### 3.2 Boundary Packing: A Subroutine

Consider a container disk  $C$ , a (possibly empty) set  $S$  of already packed disks that overlap with  $C$ , and another disk  $r_i$  to be packed, see Fig. 4. We *pack*  $r_i$  into  $C$  adjacently to the boundary of  $C$  as follows: Let  $\alpha$  be the maximal polar angle realized by a midpoint of a disk



■ **Figure 4** Boundary Packing places disks into a container disk  $C$  adjacent to the boundary of  $C$  as long as the diameter of the disks to be packed is at least as large as a given threshold  $\mathcal{T}$  or until the current disk does no longer fit into  $C$ . Initially, we have  $\mathcal{T} = \frac{1}{4}$ .

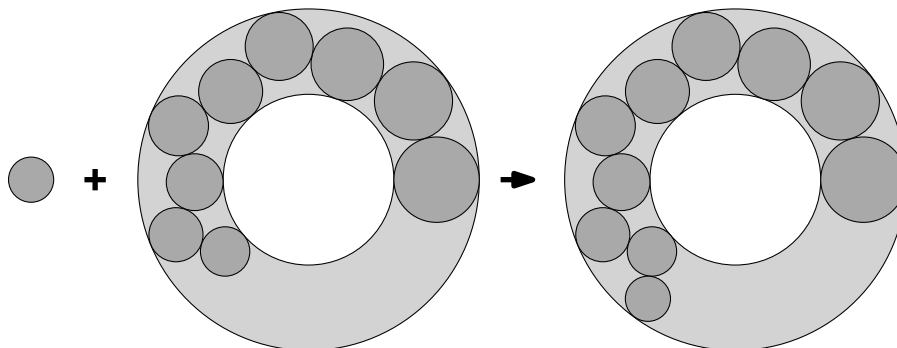
from  $S$ . We choose the midpoint of  $r_i$  realizing the smallest possible polar angle  $\beta \geq \alpha$  such that  $r_i$  touches the outer boundary of  $C$  from the interior of  $C$  without overlapping another disk from  $S$ , see Fig. 4. If  $r_i$  cannot be packed into  $C$ , we say that  $r_i$  does not fit into  $R$ .

Let  $0 < \mathcal{T} \leq \frac{1}{4}$ , called the *threshold*. *Boundary Packing* iteratively packs disks in decreasing order into  $C$  until the current disk  $r_i$  does not fit into  $C$  or the radius of  $r_i$  is smaller than  $\mathcal{T}$ .

### 3.3 Ring Packing: A Subroutine

Consider a ring  $R := R[r_{\text{out}}, r_{\text{in}}]$  with inner radius  $r_{\text{in}}$  and outer radius  $r_{\text{out}}$ , a (possibly empty) set  $S$  of already packed disks that overlap with  $R$ , and another disk  $r_i$  to be packed, see Fig. 5. We *pack*  $r_i$  into  $R$  adjacent to the outer (inner) boundary of  $R$  as follows: Let  $\alpha$  be the maximal polar angle realized by a midpoint of a disk from  $S$ . We choose the midpoint of  $r_i$  realizing the smallest possible polar angle  $\beta \geq \alpha$  such that  $r_i$  touches the outer (inner) boundary of  $R$  from the interior of  $R$  without overlapping another disk from  $S$ . If  $r_i$  cannot be packed into  $R$ , we say that  $r_i$  does not fit into  $R$  (*adjacent to the outer (inner) boundary*).

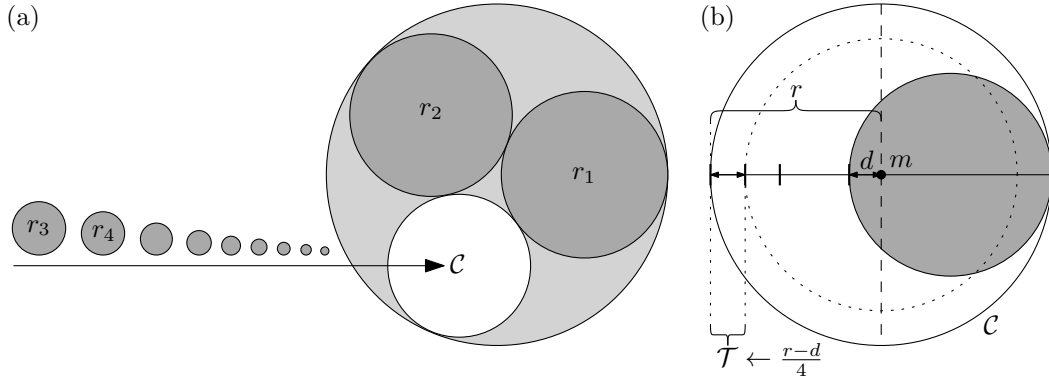
*Ring Packing* iteratively packs disks into  $R$  alternating adjacent to the inner and outer boundary. If the current disk  $r_i$  does not fit into  $R$  Ring Packing stops and we declare  $R$  to be FULL. If  $r_{i-1}$  and  $r_i$  could pass each other, i.e., the sum of the diameters of  $r_{i-1}$  and  $r_i$  are smaller than the width of  $R$ , Ring Packing stops and we declare  $R$  to be CLOSED.



■ **Figure 5** Ring Packing packs disks into a ring  $R[r_{\text{out}}, r_{\text{in}}]$ , alternating adjacent to the outer and to the inner boundary of  $R$ .



### 3.4 Description of the Algorithm



■ **Figure 6** (a): If  $r_1, r_2 \geq 0.495\mathcal{C}$ , Boundary Packing packs  $r_1, r_2$  into  $\mathcal{C}$ . We update the current container disk  $\mathcal{C}$  as the largest disk that fits into  $\mathcal{C}$  and recurse on  $\mathcal{C}$  with  $r_3, \dots, r_n$ . (b): Determining the threshold  $\mathcal{T}$  for disks packed by Boundary Packing.

Our algorithm *creates* rings. A ring only exists after it is created. We stop packing at any point in time when all disks are packed. Furthermore, we store the current threshold  $\mathcal{T}$  for Boundary Packing and the smallest inner radius  $r_{\min}$  of a ring created during the entire run of our algorithm. Initially, we set  $\mathcal{T} \leftarrow \frac{1}{4}, r_{\min} \leftarrow 1$ . Our algorithm works in five phases:

- **Phase 1 – Recursion:** If  $r_1, r_2 \geq 0.495\mathcal{C}$ , apply Boundary Packing to  $r_1, r_2$ , update  $\mathcal{C}$  as the largest disk that fits into  $\mathcal{C}$  and  $\mathcal{T}$  as the radius of  $\mathcal{C}$ , and recurse on  $\mathcal{C}$ , see Fig. 6.
- **Phase 2 – Boundary Packing:** Let  $r$  be the radius of  $\mathcal{C}$ . If the midpoint  $m$  of  $\mathcal{C}$  lies inside a packed disk  $r_i$ , let  $d$  be the minimal distance of  $m$  to the boundary of  $r_i$ , see Fig. 6(b). Otherwise, we set  $d = 0$ .

We apply Boundary Packing to the container disk  $\mathcal{C}$  with the threshold  $\mathcal{T} \leftarrow \frac{r-d}{4}$ .

- **Phase 3 – Ring Packing:** We apply Ring Packing to the ring  $R := R[r_{\text{out}}, r_{\text{in}}]$  determined as follows: Let  $r_i$  be the largest disk not yet packed. If there is no open ring inside  $\mathcal{C}$ , we create a new open ring  $R[r_{\text{out}}, r_{\text{in}}] \leftarrow R[r_{\min}, r_{\min} - 2r_i]$ . Else, let  $R[r_{\text{out}}, r_{\text{in}}]$  be the open ring with the largest inner radius  $r_{\text{in}}$ .
- **Phase 4 – Managing Rings:** Let  $R[r_{\text{out}}, r_{\text{in}}]$  be the ring filled in Phase 3. We declare  $R[r_{\text{out}}, r_{\text{in}}]$  to be closed and proceed as follows: Let  $r_i$  be the largest disk not yet packed. If  $r_i$  and  $r_{i+1}$  can pass one another inside  $R[r_{\text{out}}, r_{\text{in}}]$ , i.e., if  $2r_i + 2r_{i+1} \leq r_{\text{out}} - r_{\text{in}}$ , we create two new open rings  $R[r_{\text{out}}, r_{\text{out}} - 2r_i]$  and  $R[r_{\text{out}} - 2r_i, r_{\text{in}}]$ .
- **Phase 5 – Continue:** If there is an open ring, we go to Phase 3. Otherwise, we set  $\mathcal{C}$  as the largest disk not covered by created rings, set  $\mathcal{T}$  as the radius of  $\mathcal{C}$ , and go to Phase 2.

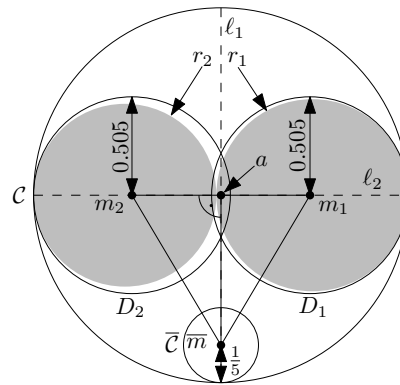
## 4 Analysis of the Algorithm

### 4.1 Analysis of Phase 1 - The Recursion

If  $r_2 \geq 0.495$ , Lemma 2 allows us to recurse on  $\mathcal{C}$  as required by Phase 1.

► **Lemma 2.** *If  $r_1, r_2 \geq 0.495\mathcal{C}$ , the volume of the largest container disk that fits into  $\mathcal{C}$  after packing  $r_1, r_2$  is at least twice the total volume of  $r_3, \dots, r_n$ , see Fig. 7.*

**Proof.** W.l.o.g., assume that the original container disk is the unit disk. Lemma 3 implies  $r_1 + r_2 \leq 1$ , which means  $r_1, r_2 \leq 0.505$ , because  $r_2 \geq 0.495$ . Furthermore,  $r_1 + r_2 \leq 1$  implies that we can move (w.l.o.g.)  $r_1, r_2$  into two disks  $D_1, D_2$  with radius 0.505, touching the



■ **Figure 7** If  $r_2 \geq 0.495$ , we can pack  $r_1, r_2$  into container disks  $D_1, D_2$  and recurse on a third disk  $\bar{C}$  whose area is twice the total area of the remaining disks.

boundary of  $C$  and with their midpoints  $m_1, m_2$  on the horizontal diameter of  $C$ , see Fig. 7. This decreases the volume of the largest disk that still fits into  $C$ . Consider the disk  $\bar{C} := \frac{1}{5}$  lying adjacent to  $C$  and with its midpoint  $\bar{m}$  on the vertical diameter  $\ell_1$  of  $C$ . Pythagoras' Theorem implies that  $|m_1\bar{m}| = \sqrt{(1 - 0.505)^2 + (1 - \frac{1}{5})^2} \approx 0.94075 > 0.505 + \frac{1}{5}$ . Finally, we observe that the area of  $\bar{C}$  is  $\frac{\pi}{25} = 0.4\pi > 0.0199 = 2(\frac{\pi}{2} - 2 \cdot \pi 0.495^2)$ . This means that the area of  $\bar{C}$  is twice the total area of the remaining disks  $r_3, r_4, r_5, \dots$ , concluding the proof. ◀

A technical key ingredient in the proof of Lemma 2 is the following lemma:

► **Lemma 3.** *The area of two disks  $r_1, r_2$  is at least  $\frac{\pi}{2} (r_1 + r_2)^2$ .*

**Proof.** The first derivative of the function mapping a radius onto the area of the corresponding disk is the periphery of the corresponding circle. As  $r_1 \geq r_2$ , decreasing  $r_1$  and increasing  $r_2$  by the same value  $\delta$  reduces the total area of  $r_1, r_2$ , while the value  $r_1 + r_2$  stays the same. Hence, we assume w.l.o.g. that  $r_1 = r_2$ . This implies that the total area of  $r_1, r_2$  is  $2\pi r_1^2 = \frac{\pi}{2} (r_1 + r_2)^2$ , concluding the proof. ◀

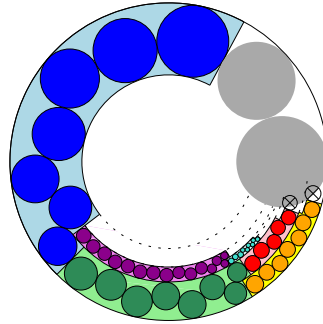
This allows us to assume  $r_2 < 0.495C$  during the following analysis.

## 4.2 Outline of the Remaining Analysis

Once our algorithm stops making recursive calls, i.e., stops applying Phase 1, Phase 1 is never applied again. W.l.o.g., let  $r_1, \dots, r_n$  be the remaining disks and  $O$  the container disk after the final recursion call.

The main idea of the remaining analysis is the following: We cover the original container disk  $O$  by a set of *sectors* that are subsets of  $O$ . Let  $r_i$  be a disk packed by Boundary Packing into the current container disk  $C$ . We define the *cone* induced by  $r_i$  as the area of  $C$  between the two tangents of  $r_i$ . We say that  $C$  is the radius of the cone. A *sector* is a subset of  $O$ .

Each disk *pays* portions, called *atomic potentials*, of its volume to different sectors of  $O$ . The total atomic potential paid by a disk  $r$  will be at most the volume of the disk  $r$ . Let  $A_1, \dots, A_k$  be the total atomic potentials paid to the sectors  $S_1, \dots, S_k \subset O$ . The *potential* of a sector  $S \subseteq O$  is the sum of the proportionate atomic potentials from  $S_1, \dots, S_k$ , i.e., the sum of all  $\frac{|S_i \cap S|}{|S_i|} A_i$  for  $i = 1, \dots, k$ . The (*virtual packing*) *density*  $\rho(S)$  of the sector  $S$  is defined as the ratio between the potential of  $S$  and the volume of  $S$ . If a sector achieves a density of  $\frac{1}{2}$ , we say that the sector is *saturated*, otherwise its *unsaturated*.



■ **Figure 8** Different sequences of rings packed by different applications of Ring Packing. The minimal rings into which the orange and red disks are packed are full. The minimal ring into which the turquoise disks are packed is open. The uncolored, crossed-out circles illustrate that the corresponding disk did not fit into the current ring, causing it to be declared full.

Our approach for proving Theorem 1 is by induction over  $n$ . In particular, we assume that  $O \setminus \mathcal{C}$  is saturated; we show that each disk  $r_i$  can be packed by our algorithm, as long as  $\mathcal{C}$  is unsaturated implying that each set of disks with total volume of at most  $\frac{|O|}{2}$  is packed. We assume for the remainder of the paper that  $\mathcal{C}$  is the unit disk, i.e.,  $\mathcal{C} = 1$ .

We consider the configuration achieved after termination.

If there is a ring that is neither full nor closed, all disks are packed.

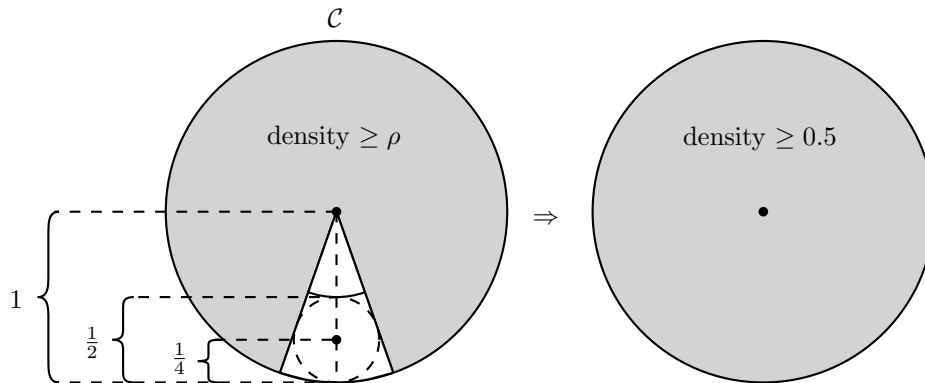
Thus, we assume that all rings computed by our algorithm are full or closed. In order to avoid that Boundary Packing stops due to a disk  $r$  not fitting, we consider the gap that is left by Boundary Packing, see Fig. 9. This gap achieves its maximum for  $r = \frac{1}{4}$ . We guarantee that  $\mathcal{C}$  has a density of

$$\rho := \frac{180^\circ}{360^\circ - 2 \arcsin\left(\frac{1/4}{3/4}\right)} < 0.56065.$$

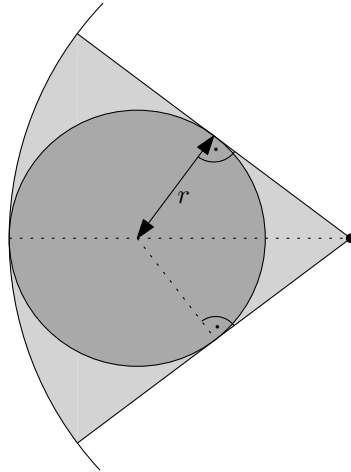
### 4.3 Analysis of Boundary Packing

The following lemma is the key ingredient for the analysis of Boundary Packing.

► **Lemma 4.** *Let  $r \in [0.2019, \frac{1}{2}]$  be a disk lying adjacent to  $\mathcal{C}$ . The cone  $C$  induced by  $r$  has a density better than  $\rho$  if  $r \in [\frac{1}{4}, 0.495]$  and at least  $\frac{1}{2}$  if  $r \in [0.2019, \frac{1}{2}]$ , see Fig. 10.*



■ **Figure 9** Ensuring a density of at least 0.5 for a ring  $R$  needs a density of 0.5606 for  $R \setminus C$ .



■ **Figure 10** A disk  $r \in [\frac{1}{4}, 0.495]$  lying adjacent to  $\mathcal{C}$  induces a cone with density of at least 0.56127 if  $r \in [\frac{1}{4}, 0.495]$  and of least  $\frac{1}{2}$  if  $r \in [\frac{1}{4}, \frac{1}{2}]$ .

**Proof.** Let  $f(r) := \frac{\pi r^2}{\arcsin(\frac{r}{1-r})}$  for  $\frac{1}{4} \leq r \leq \frac{1}{2}$ . Thus we have

$$f'(r) = \frac{2\pi r}{\arcsin\left(\frac{r}{1-r}\right)} - \frac{\pi r^2 \left(\frac{1}{1-r} + \frac{r}{(1-r)^2}\right)}{\arcsin\left(\frac{r}{1-r}\right)^2 \sqrt{1 - \frac{r^2}{(1-r)^2}}}.$$

Solving  $f'(r) = 0$  yields  $r \approx 0.39464$ . Furthermore, we have  $f(\frac{1}{4}) \approx 0.57776$ ,  $f(0.39464) = 0.68902$ ,  $f(\frac{1}{2}) = 0.5$ , and  $f(0.495) \approx 0.56127$ . Thus,  $f$  restricted to  $[\frac{1}{4}, 0.495]$  achieves at 0.495 its global minimum 0.56127. A similar approach implies that  $f$  restricted to  $[0.2019, \frac{1}{2}]$  attains its global minimum  $\frac{1}{2}$  at  $\frac{1}{2}$ . ◀

The following lemma proves that all disks  $r_i \geq \frac{c}{4}$  that are in line to be packed into a container disk  $\mathcal{C}$  can indeed be packed into  $\mathcal{C}$ .

► **Lemma 5.** *All disks  $r_i \geq \frac{1}{4}$  that are in line to be packed into  $\mathcal{C}$  by Boundary Packing do fit into  $\mathcal{C}$ .*

**Proof.** Assume that there is a largest disk  $r_k \geq \frac{1}{4}$  not packed adjacent to  $\mathcal{C}$ . Each disk  $r_i$  from  $r_1, \dots, r_{k-1}$  pays its entire volume to the cone induced by  $r_i$ . Lemma 4 implies that each cone is saturated. As  $r_k$  does not fit between  $r_1, r_{k-1}$  and is adjacent to  $\mathcal{C}$ , Lemma 4 implies that the area of  $\mathcal{C}$  that is not covered by a cone induced by  $r_1, \dots, r_{k-1}$  has a volume smaller than twice the volume of  $r_k$ . This implies that the total volume of  $r_1, \dots, r_k$  is larger than half of the volume of  $\mathcal{C}$ . This implies that the total input volume of  $r_1, \dots, r_n$  is larger than twice the volume of the container. This is a contradiction, concluding the proof. ◀

► **Corollary 6.** *If  $r_n \geq \frac{1}{4}$ , our algorithm packs all input disks.*

Thus, we assume w.l.o.g.  $r_n < \frac{1}{4}$ , implying that our algorithm creates rings.

#### 4.4 Analysis of Ring Packing

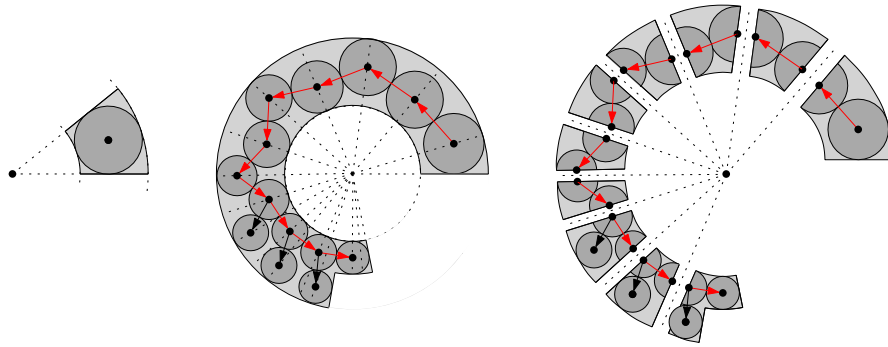
For the following definition, see Fig. 11 (Middle).

► **Definition 7.** *A zipper  $Z$  is a (maximal) sequence  $\langle r_k, \dots, r_\ell \rangle$  of disks that are packed into a ring  $R$  during an application of Ring Packing. The length of  $Z$  is defined as  $k - \ell + 1$ .*

## 35:10 Worst-Case Optimal Disks Packing into Disks

Consider a zipper  $\langle r_k, \dots, r_\ell \rangle$  packed into a ring  $R$ . For a simplified presentation, we assume in Section 4.4 that the lower tangent of  $r_k$  realizes a polar angle of zero, see Fig. 11.

We refine the potential assignments of zippers as follows. Let  $Z = \langle r_k, \dots, r_\ell \rangle$  be an arbitrary zipper and  $R$  the ring into which  $Z$  is packed. In order to subdivide  $R$  into sectors corresponding to specific parts of the zipper, we consider for each disk  $r_i$  the *center ray*, which is the ray starting from  $m$  and passing the midpoint of  $r_i$ . Let  $t_1, t_2$  be two rays starting in  $m$ . We say that  $t_1$  lies above  $t_2$  when the polar angle realized by  $t_1$  is at least as large as the polar angle realized by  $t_2$ .  $t_1$  is the *minimum* (*maximum*) of  $t_1, t_2$  if  $t_1$  does not lie above (below)  $t_2$ . Furthermore, the *upper tangent* (*lower tangent*) of a disk  $r_i$  is the maximal (minimal) tangent of  $r_i$ .



■ **Figure 11** A maximal sequence of disks that are packed into a ring during an application of Boundary Packing. The corresponding sectors are illustrated in light gray. **Left:** A zipper of size one and the corresponding sector. **Middle:** A zipper of size 14, the resulting directed adjacency graph (black/red), and the path (red) leading from the largest disk to the smallest disk. The first seven edges of  $P$  are diagonal and the remaining edges of  $P$  are vertical. **Right:** The zipper and the sector disassembled into smaller sectors corresponding to the edges of the red path.

If the zipper  $Z$  consists of one disk  $r_k$ , the *sector*  $S$  of  $Z$  is that part of  $R$  between the two tangents to  $r_k$  and  $r_k$  pays its entire volume to  $S$ .

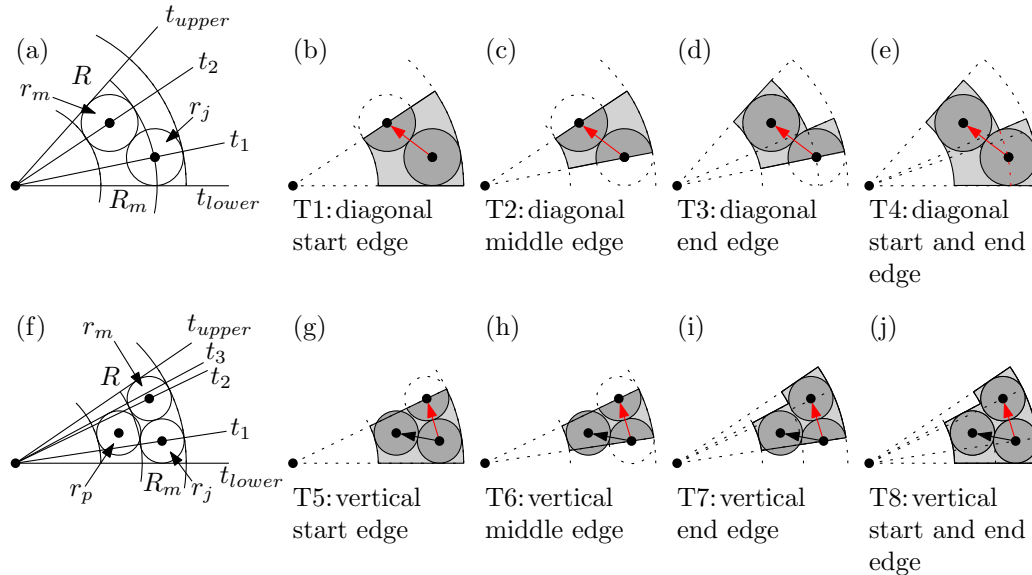
► **Lemma 8.** *The density of the sector  $S$  of a zipper of length one is at least 0.77036.*

**Proof.** As the zipper consists of only one disk  $r_k$ ,  $r_k$  touches both the inner and the outer boundary of  $R$ . Hence, the density of  $S$  is not increased by assuming that the inner radius of  $R$  is equal to the diameter of  $r_k$ . Hence, the density of  $S$  is at least  $\frac{\pi}{12 \arcsin(1/3)} \approx 0.77036$ . ◀

Assume the zipper  $\langle r_k, \dots, r_\ell \rangle$  consists of at least two disks. We define the *adjacency graph*  $G = (\{r_k, \dots, r_\ell\}, E)$  as a directed graph as follows: There is an edge  $(r_j, r_i)$  if (1)  $r_i \leq r_j$  and (2)  $r_i, r_j$  are touching each other, see Fig. 11 (Right). As Ring Packing packs each disk  $r_i$  with midpoint  $m_i$  such that  $m_i$  realizes the smallest possible polar angle, there is a path  $e_k, \dots, e_{\ell-1} =: P$  connecting  $r_k$  to  $r_\ell$  in the adjacency graph  $G$ , see Fig. 11 (Middle).  $e_k$  is the *start* edge of  $P$  and  $e_{\ell-1}$  is the *end* edge of  $P$ . The remaining edges of  $P$  that are neither the start nor the end edge of  $G$ , are *middle* edges of  $P$ . Furthermore, an edge  $(r_j, r_m) = e_i \in P$  is *diagonal* if  $r_j, r_m$  are touching different boundary components of  $R$ . Otherwise, we call  $e_i$  *vertical*.

Depending on whether  $e_i$  is a start, middle, or an end edge and on whether  $e_i$  is diagonal or vertical, we classify the edges of the path  $P$  by eight different types T1-T8. For each type we individually define the sector  $A_i$  belonging to an edge  $(r_j, r_m) = e_i \in P$  and the potential assigned to  $A_i$ , called the *potential* of  $e_i$ . Let  $t_{\text{lower}}$  be the minimum of the lower tangents of  $r_j, r_m$  and  $t_{\text{upper}}$  the maximum of the upper tangents of  $r_j, r_m$ , see Fig. 12 (a). Furthermore, let  $t_1, t_2$  be the center rays of  $r_j, r_m$ , such that  $t_1$  does not lie above  $t_2$ .

For the case that  $e_i = (r_j, r_m)$  is a vertical edge, we consider additionally the disk  $r_p$  that is packed into  $R$  after  $r_j$  and before  $r_m$ , see Fig. 12 (f). Let  $t_3$  be the maximum of  $t_2$  and the upper tangent of  $r_p$ , see Fig. 12.



■ **Figure 12** The eight possible configurations of an edge  $e_i$  (red) of  $P$ , the corresponding sectors (light gray), and the potentials (dark gray) paid by the involved disks to the sector.

**T1 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a diagonal start edge (as shown in Fig. 12(b)), the sector of  $e_i$  is that part of  $R$  that lies between  $t_{lower}$  and  $t_2$ .

**The potential of  $e_i$ :**  $r_j$  pays its entire volume and  $r_m$  the half of its volume to the sector of  $e_i$ .

**T2 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a diagonal middle edge, (as shown in Fig. 12(c)), the sector of  $e_i$  is that part of  $R$  that lies between  $t_1$  and  $t_2$ .

**The potential of  $e_i$ :**  $r_j$  and  $r_m$  pay the half of its volume to the sector of  $e_i$ .

**T3 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a diagonal end edge, (as shown in Fig. 12(d)), the sector of  $e_i$  consists of two parts: (1) The first is the part of  $R$  that lies between the upper tangent and the center ray of  $r_j$ . (2) Let  $R_m$  be the smallest ring enclosing  $r_m$ . The second part of the sector is that part of  $R_m$  that lies between the upper tangent of  $r_m$  and the minimum of  $t_1$  and the lower tangent of  $r_m$ .

**The potential of  $e_i$ :**  $r_j$  pays the half of its volume and  $r_m$  its entire volume to the sector of  $e_i$ .

**T4 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a diagonal start and end edge, (as shown in Fig. 12(e)), the sector of  $e_i$  is the union of two sectors: (1) The first is the part of  $R$  that lies between the lower and the upper tangent of  $r_j$ . (2) The second is that part of  $R_m$  that lies between the lower and the upper tangent of  $r_m$ .

**The potential of  $e_i$ :**  $r_j, r_m$  pay their entire volume to the sector of  $e_i$ .

**T5 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a vertical start edge, (as shown in Fig. 12(g)), the sector of  $e_i$  is that part of  $R$  that lies between the minimum of the lower tangents of  $r_j, r_p$  and the center ray of  $r_m$ .

**The potential of  $e_i$ :**  $r_j, r_p$  pay their entire volume and  $r_m$  the half of its volume to the sector of  $e_i$ .

### 35:12 Worst-Case Optimal Disks Packing into Disks

**T6 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a vertical middle edge, (as shown in Fig. 12(h)), the sector of  $e_i$  is that part of  $R$  that lies between the center rays of  $r_j, r_m$ .

**The potential of  $e_i$ :**  $r_p$  pays its entire volume and  $r_j, r_m$  pay half of their respective volume to the sector of  $e_i$ .

**T7 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a vertical end edge, (as shown in Fig. 12(i)), the sector of  $e_i$  consists of two parts: (1) The first is that part of  $R$  that lies between the center ray of  $r_j$  and the upper tangent of  $r_p$ . (2) Let  $R_m$  be the smallest ring enclosing  $r_m$ . The second part of the sector is the part of  $R_m$  that lies between the center ray of  $r_j$  and the upper tangent of  $r_m$ .

**The potential of  $e_i$ :**  $r_j$  pays the half of its volume and  $r_p, r_m$  their entire volumes to the sector of  $e_i$ .

**T8 The sector of  $e_i$ :** If  $e_i = (r_j, r_m)$  is a vertical start and end edge, (as shown in Fig. 12(j)), the sector of  $e_i$  consists of two parts: (1) The first is that part of  $R$  that lies between the minimum of the lower tangents of  $r_j, r_p$  and the maximum of the upper tangents  $r_j, r_p$ . (2) Let  $R_m$  be the smallest ring enclosing  $r_m$ . The second part of the sector is that part of  $R_m$  that lies between the lower and the upper tangent of  $r_m$ .

**The potential of  $e_i$ :**  $r_j, r_p, r_m$  pay their entire volume to the sector of  $e_i$ .

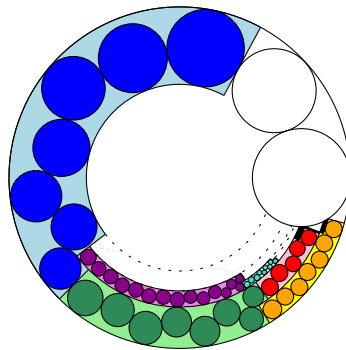
For simplicity, we also call the density of the sector of an edge  $e_i \in P$  the *density of  $e_i$* . The *sector* of a zipper is the union of the sectors of the edges of  $P$ .

► **Lemma 9.** *Let  $Z = \langle r_k, \dots, r_\ell \rangle$  be a zipper of length at least two and  $P$  a path in the adjacency graph of  $Z$  connecting  $r_k$  with  $r_\ell$ . Each edge  $e_i \in P$  has a density of at least  $\rho$ .*

The proof of Lemma 9 is the only computer-assisted proof. All remaining proofs are analytic. Due to space constraints, the proof of Lemma 9 is given in the full version of the paper [3]. Combining Lemmas 8 and 9 yields the following.

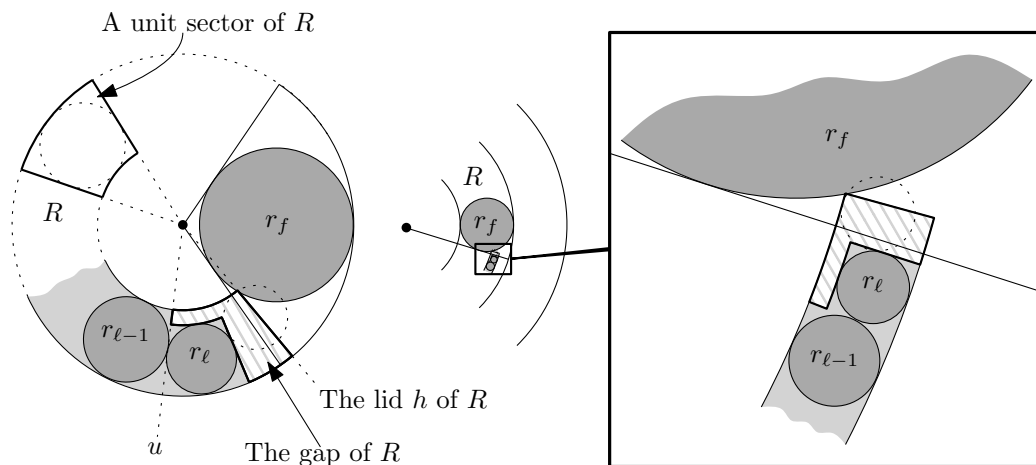
► **Corollary 10.** *Sectors of zippers have a density of at least  $\rho$ .*

Ring Packing stops when the sum of the diameters of the current disk  $r_i$  and the disk packed last  $r_{i-1}$  is smaller than the width  $w$  of the current ring, i.e., if  $2r_{i-1} + 2r_i < w$ . If  $2r_{i-1} + 2r_i < w$ , Phase 5 partitions the current ring into two new open rings with widths  $2r_i, w - 2r_i$ . Hence, the sectors of zippers packed by Ring Packing become firmly interlocked without leaving any gaps between two zippers, see Fig. 13. The only sectors that we need to



■ **Figure 13** The sectors of rings packed by Ring Packing become firmly interlocked without leaving any gaps between two sectors. The minimal rings into which the orange and the red zippers are packed are full. The minimal ring into which the turquoise zipper is packed is open.

care about are the gaps that are left by Ring Packing due to the second break condition, i.e., the current disk does not fit into the current ring, see the black sectors in Fig. 13.



■ **Figure 14** The lid, the gap (shaded white-gray), and a unit sector of a ring  $R$ .

► **Corollary 11.** *Let  $R$  be a minimal ring and  $G$  its gap.  $R \setminus G$  has a density of at least  $\rho$ .*

In order to analyze the gaps left by Ring Packing, we first need to observe for which rings we need to consider gaps. In particular, we have two break conditions for Ring Packing:

(1) The current disk  $r_i$  does not fit into the current ring  $R$ , causing us to close the ring and disregard it for the remainder of the algorithm?

(2) The current and the last disk  $r_{i-1}$  packed into  $R$  can pass one another, resulting in  $R$  to be partitioned into several rings with smaller widths. Thus, we obtain that two computed rings  $R_1, R_2$  either do not overlap or  $R_1$  lies inside  $R_2$ .

► **Definition 12.** *Consider the set of all rings  $R_1, \dots, R_k$  computed by our algorithm. A ring  $R_i$  is maximal if there is no ring  $R_j$  with  $R_i \subset R_j$ . A ring  $R_i$  is minimal if there is no ring  $R_j$  with  $R_i \supset R_j$ .*

By construction of the algorithm, each ring is partitioned into minimal rings. Thus, we define gaps only for minimal rings, see Figure 14 and Definition 13.

► **Definition 13.** *Let  $Z = \langle \dots, r_{\ell-1}, r_\ell \rangle$  be a zipper of length at least 2 inserted into a minimal ring  $R$ . The lid  $h$  of  $R$  is the ray above the upper tangent  $u$  of  $r_\ell$  such that  $h$  realizes a maximal polar angle while  $h \cap R$  does not intersect an already packed disk  $r_f$  with  $f \leq \ell - 1$ , see Fig. 14. The gap of  $R$  is the part of  $R$  between the upper tangent  $u$  of  $r_{\ell-1}$  and the lid of  $R$  which is not covered by sectors of  $Z$ , see the white-gray striped sectors in Fig. 14.*

*A unit sector of  $R$  is a sector of  $R$  that lies between the two tangents of a disk touching the inner and the outer boundary of  $R$ , see Fig. 14. The unit volume  $U_R$  of  $R$  is the volume of a unit sector of  $R$ .*

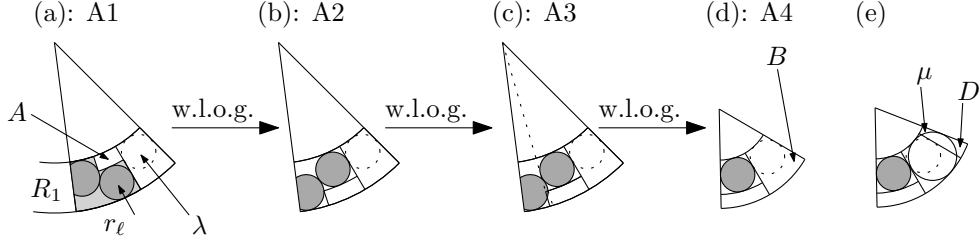
The lid of a gap lies either inside a cone induced by a disk packed by Boundary Packing, see Fig. 14 (Left), or inside the sector of a zipper packed by Ring Packing, see Fig. 14 (Right). This leads to the following observation: Each minimal ring  $R$  is covered by the union of cones induced by disks packed by Boundary Packing into  $R$ , sectors of zippers packed by Ring Packing into  $R$ , and the gap of  $R$ .

Next, we upper bound the volume of the gap of minimal rings.

► **Lemma 14.** *The gap of a minimal ring  $R$  has a volume of at most  $1.07024U_R$ .*



### 35:14 Worst-Case Optimal Disks Packing into Disks



■ **Figure 15** Simplifying assumptions that do not increase the density.

**Proof.** As we want to upper bound the volume of the gap w.r.t. the unit volume  $U_R$  of  $R$ , w.l.o.g. we make the following assumptions (A1)-(A4), see Fig. 15:

- (A1) The largest disk  $\lambda$  inside  $R$  touching  $h$  from below, the upper tangent of  $r_\ell$  from above, and the inner boundary of  $R$ , such that  $\lambda$  does not overlap with any other disks from below, has the same radius as  $r_\ell$ , see Fig. 15(a).
- (A2) The last disk  $r_\ell$  packed into  $R$  touches the inner boundary of  $R$ , see Fig. 15(b).
- (A3) The empty pocket  $A$  left by the sector of the end edge of the zipper inside  $R$  is bounded from below by the lower tangent of  $r_\ell$  but not by the upper tangent of  $r_{\ell-1}$ , see Fig. 15(c).
- (A4)  $r_{\text{out}} = 1, r_{\text{in}} = \frac{1}{2}$ , see Fig. 15(d).

Let  $B$  be the sector of  $R$  that lies between the two tangents of  $\lambda$ , see Fig. 15(d). We upper bound the volume of the gap of  $R$  as  $|A| + |B| \leq 1.07024U_R$ , as follows.

Let  $\mu \subset R$  be the disk touching the inner and the outer boundary of  $R_1$  and the upper tangent of  $r_\ell$  from above, see Fig. 15(e). Furthermore, let  $D$  be the part of the cone induced by  $\mu$  which lies inside  $R$  and between the upper and lower tangent of  $\mu$ , see Fig. 15(e).

In the following, we show that  $|A| - |D| \leq 0.07024U_R$ .

$$\begin{aligned}
 |A| - |D| &\leq \frac{2 \arcsin\left(\frac{\lambda}{\frac{1}{2} + \lambda}\right)}{2\pi} \pi \left(1 - \left(\frac{1}{2} + 2\lambda\right)^2\right) \\
 &\quad - \frac{2 \arcsin\left(\frac{1}{3}\right) - 2 \arcsin\left(\frac{\lambda}{\frac{1}{2} + \lambda}\right)}{2\pi} \pi \left(\frac{3}{4}\right) \\
 &= \arcsin\left(\frac{\lambda}{\frac{1}{2} + \lambda}\right) \left(\frac{7}{4} - \left(\frac{1}{2} + 2\lambda\right)^2\right) \\
 &\quad - \frac{3}{4} \arcsin\left(\frac{1}{3}\right) =: V_{AD}.
 \end{aligned}$$

The first derivative of  $V_{AD}$  is

$$\begin{aligned}
 \frac{d V_{AD} \lambda}{d \lambda} &= \frac{\left(\frac{1}{\frac{1}{2} + \lambda} - \frac{\lambda}{\left(\frac{1}{2} + \lambda\right)^2}\right) \left(\frac{7}{4} - \left(2\lambda + \frac{1}{2}\right)^2\right)}{\sqrt{1 - \frac{\lambda^2}{\left(\frac{1}{2} + \lambda\right)^2}}} \\
 &\quad - 4 \arcsin\left(\frac{\lambda}{\frac{1}{2} + \lambda}\right) \left(2\lambda + \frac{1}{2}\right).
 \end{aligned}$$

Solving  $\frac{d V_{AD} \lambda}{d \lambda} = 0$  yields  $\lambda \approx 0.196638$ . Finally, we observe that  $V_{AD}\left(\frac{1}{8}\right) \approx -0.01576$ ,  $V_{AD}(0.196638) \approx 0.01756$ ,  $V_{AD}\left(\frac{1}{4}\right) = 0$ . This implies that  $|A| - |D| \leq 0.01756 \leq 0.07024U_R$ , because  $U_R \geq \frac{1}{4}$ . ◀

### 4.5 Analysis of the Algorithm for the Case $r_1 \leq 0.495$

We show that each computed minimal ring is saturated, see Corollary 17. Let  $R_1, \dots, R_h \subseteq \mathcal{C}$  be the created minimal rings ordered decreasingly w.r.t. their outer radii. The inner boundary of  $R_i$  is the outer boundary of  $R_{i+1}$  for  $i = 1, \dots, h - 1$ .

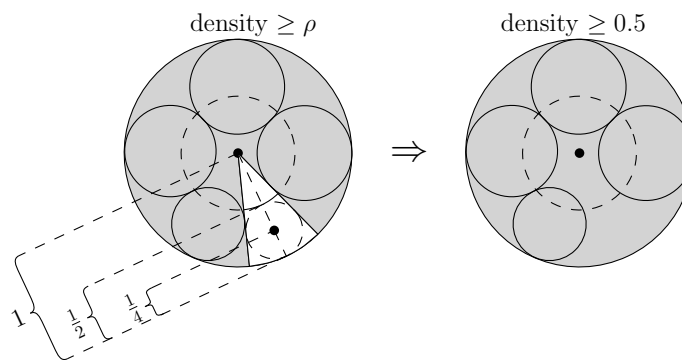
We show by induction over  $h$  that  $R := R[r_{\text{out}}, r_{\text{in}}] := R_h$  is saturated. Thus, we assume that  $R_1, \dots, R_{h-1}$  are saturated, implying that  $\mathcal{C} \setminus r_{\text{out}}$  is saturated, where  $r_{\text{out}}$  is the outer radius of  $R_h$ .

For the remainder of Section 4.5, each disk  $r_i$  packed by Boundary Packing pays its entire volume to the cone induced by  $r_i$ .

► **Lemma 15.** *Assume  $r_n < \frac{1}{4}$ . There is at least one disk  $r_k$  packed into  $R$  and touching both the inner and the outer boundary of  $R$ .*

**Proof.** Assume that our algorithm did not pack a disk with radius smaller than  $\frac{1}{4}$  adjacent to  $\mathcal{C}$ . Let  $r_k$  be the largest disk not packed adjacent to  $\mathcal{C}$  into  $R$ .

By Lemma 5, we obtain that  $r_k$  is smaller than  $\frac{1}{4}$ . This implies that the volume of the sector that is not covered by the cones induced by  $r_1, \dots, r_{k-1}$  is upper bounded by  $\arcsin(\frac{1}{3})$ , see Fig. 16.



■ **Figure 16** Ensuring density of at least  $\rho$  for all cones induced by disks packed by Boundary Packing implies a density of at least 0.5 for the entire container disk.

Each disk  $r_i$  from  $r_1, \dots, r_{k-1}$  pays its entire volume to the cone induced by  $r_i$ . Lemma 4 implies that each cone has a density of at least  $\rho$ , because  $r_1, \dots, r_n \leq 0.495$ . This implies that the total volume of  $r_1, \dots, r_{k-1}$  is at least  $\pi \cdot \rho \cdot \frac{2\pi - 2\arcsin(1/3)}{2\pi} = \rho(\pi - \arcsin(1/3)) > \frac{\pi}{2}$  contradicting the assumption that the total input volume is no larger than  $\frac{\pi}{2}$ . ◀

► **Lemma 16.**  *$R_h$  is saturated.*

**Proof.** Let  $S_1$  be the sector of  $R_h$  that is covered by cones induced by disks packed by Boundary Packing or by sectors of zippers packed by Ring Packing. Lemma 15 implies that there is a disk  $r_k$  packed into  $R_h$  such that  $r_k$  touches the inner and the outer boundary of  $R_h$ . Let  $S_2$  be the sector of  $R_h$  between the lower and the upper tangent of  $r_k$ .

We move potentials  $\delta_1, \delta_2$  from  $S_1, S_2$  to a potential variable  $\Delta$  and guarantee that  $\Delta$  is at least  $\frac{1}{2}$  times the volume of the gap  $G$  of  $R_h$ . Finally, we move  $\Delta$  to  $G$ , implying that  $G$  is saturated, which in turn implies that  $R_h$  is saturated.

Lemma 8 implies that the density of  $S_2$  is at least 0.77036. We move a potential  $\delta_2 := (0.77036 - \rho) |S_1| > 0.20971 U_{R_h}$  from  $S_2$  to  $\Delta$ , implying that  $S_2$  has still a density of  $\rho$ .

## 35:16 Worst-Case Optimal Disks Packing into Disks

Combining Lemma 4 and Corollary 10 yields that  $S_1$  has a density of at least  $\rho$ . Lemma 14 implies that the volume of the gap of  $R_h$  is at most  $1.07024U_R$ . The volume of  $R_h$  is at least  $\frac{2\pi}{2\arcsin(\frac{1}{3})}U_{R_h} > 9.24441U_{R_h}$ . Thus, the volume of  $S_1$  is at least  $(9.24441 - 1.07024)U_{R_h} = 8.17417U_{R_h}$ . Hence, we move a potential  $\delta_1 := (\rho - \frac{1}{2})8.17417U_{R_h} > 0.49576U_{R_h}$  to  $\Delta$ .

We have  $\Delta = \delta_1 + \delta_2 > 0.49576 + 0.20971 = 0.70547$ , which is large enough to saturate a sector of volume  $V_\Delta = 2 \cdot 0.70547 = 1.41094U_{R_h}$ . As  $|G| \leq 1.07024$ , moving  $\Delta$  to  $G$  yields that  $G$  is saturated, which implies that  $R_h$  is saturated. This concludes the proof. ◀

► **Corollary 17.** *Each minimal ring is saturated.*

As each ring can be partitioned into minimal rings, we obtain the following.

► **Corollary 18.** *All rings are saturated.*

Combining Lemma 5 and Corollary 18 yields that all disks are packed.

► **Lemma 19.** *Our algorithm packs all input disks.*

**Proof.** By induction assumption we know that  $O \setminus \mathcal{C}$  is saturated and Corollary 18 implies that all rings inside  $\mathcal{C}$  are also saturated.

Let  $\bar{\mathcal{C}}$  be the disk left after removing all rings from  $\mathcal{C}$ , implying that  $\bar{\mathcal{C}}$  is empty. Lemma 5 implies that a final iteration of Boundary Packing to  $\bar{\mathcal{C}}$  yields that all remaining disks are packed into  $\bar{\mathcal{C}}$ . This concludes the proof. ◀

### 4.6 Analysis of the Algorithm for the Case $0.495 \leq r_1$

We prove that all disks are packed if  $0.495 \leq r_1$  by distinguishing whether  $0.495 \leq r_1 \leq \frac{1}{2}$  or  $\frac{1}{2} < r_1$ . If  $0.495 \leq r_1 \leq \frac{1}{2}$ , we apply a similar approach as used for the case  $r_1 \leq 0.495$ . The additional difficulty for the case of  $0.495 \leq r_1 \leq \frac{1}{2}$  is that the cone induced by  $r_1$  may have a density of  $\frac{1}{2}$ . Thus, we have to generate some extra potential from the remaining sectors in order to ensure that the gaps of the rings are saturated, see [3] for details.

► **Lemma 20.** *If  $0.495 \leq r_1 \leq \frac{1}{2}$ , our algorithm packs all disks into the container disk.*

If  $\frac{1}{2} < r_1$ , we need to refine our analysis because the midpoint of the container disk  $\mathcal{C}$  lies inside  $r_1$ . In particular, we consider a *half disk*  $H$  lying inside  $\mathcal{C}$  such that  $H$  and  $r_1$  are touching each other. The volume of  $H$  is at least twice the volume of the remaining disks to be packed, see Figure 17. Finally, applying a similar approach as used in the case of  $0.495 \leq r_1 \leq \frac{1}{2}$  to  $H$  yields that all disks are packed, see [3] for details.

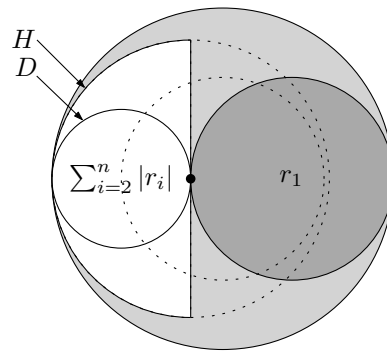
► **Lemma 21.** *If  $\frac{1}{2} < r_1$ , our algorithm packs all disks into the original container disk.*

Lemma 21 concludes the proof of Theorem 1.

## 5 Hardness

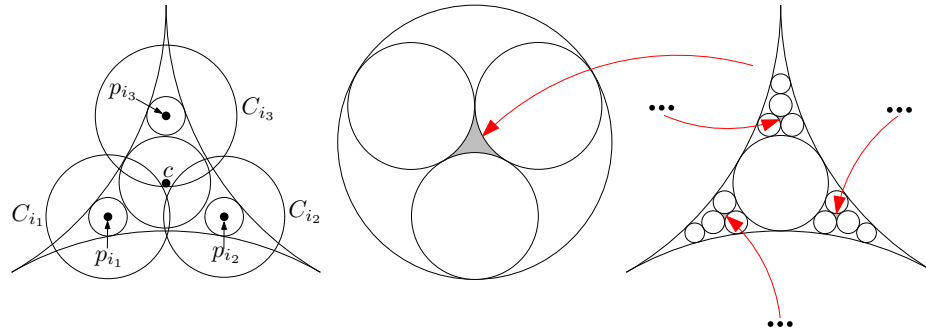
It is straightforward to see that the hardness proof for packing disks into a square can be adapted to packing disks into a disk, as follows.

► **Theorem 22.** *It is NP-hard to decide whether a given set of disks fits into a circular container.*



■ **Figure 17** The total volume of the remaining disks to be packed is smaller than the volume of the white disk  $D$ . As  $|H| = 2|D|$ , it suffices to guarantee that  $H$  is saturated.

The proof is completely analogous to the one by Demaine, Fekete, and Lang in 2010 [2], who used a reduction from 3-PARTITION. Their proof constructs a disk instance which first forces some symmetrical free “pockets” in the resulting disk packing. The instance’s remaining disks can then be packed into these pockets if and only if the related 3-PARTITION instance has a solution. Similar to their construction, we construct a symmetric triangular pocket by using a set of three identical disks of radius  $\frac{\sqrt{3}}{2+\sqrt{3}}$  that can only be packed into a unit disk by touching each other. Analogous to [2], this is further subdivided into a sufficiently large set of identical pockets. The remaining disks encode a 3-PARTITION instance that can be solved if and only if the disks can be partitioned into triples of disks that fit into these pockets.



■ **Figure 18** Elements of the hardness proof: (1) A symmetric triangular pocket from [2], allowing three disks with centers  $p_{i_1}, p_{i_2}, p_{i_3}$  to be packed if and only if the sum of the three corresponding numbers from the 3-PARTITION instance is small enough. (2) Creating a symmetric triangular pocket in the center by packing three disks of radius  $\frac{\sqrt{3}}{2+\sqrt{3}}$  and the adapted argument from [2] for creating a sufficiently large set of symmetric triangular pockets.

## 6 Conclusions

We have established the critical density for packing disks into a disk, based on a number of advanced techniques that are more involved than the ones used for packing squares or disks into a square. Numerous questions remain, in particular the critical density for packing disks of bounded size into a disk or the critical density of packing squares into a disk. These remain for future work; we are optimistic that some of our techniques will be useful.

## References

- 1 I. Castillo, F. J. Kampas, and J. D. Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.
- 2 E. D. Demaine, S.P. Fekete, and R. J. Lang. Circle Packing for Origami Design is Hard. In *Origami<sup>5</sup>: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011. [arXiv:1105.0791](#).
- 3 S. P. Fekete, P. Keldenich, and C. Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. *Computing Research Repository (CoRR)*, 2019. [arXiv:1903.07908](#).
- 4 S. P. Fekete, S. Morr, and C. Scheffer. Split Packing: Algorithms for Packing Circles with Optimal Worst-Case Density. *Discrete & Computational Geometry*, 2018. [doi:10.1007/s00454-018-0020-2](#).
- 5 S. P. Fekete and J. Schepers. New Classes of Fast Lower Bounds for Bin Packing Problems. *Math. Program.*, 91(1):11–31, 2001.
- 6 S. P. Fekete and J. Schepers. A General Framework for Bounds for Higher-Dimensional Orthogonal Packing Problems. *Math. Methods Oper. Res.*, 60:311–329, 2004.
- 7 F. Fodor. The Densest Packing of 19 Congruent Circles in a Circle. *Geometriae Dedicata*, 74:139–145, 1999.
- 8 F. Fodor. The Densest Packing of 12 Congruent Circles in a Circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 41:401–409, 2000.
- 9 F. Fodor. The Densest Packing of 13 Congruent Circles in a Circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 44:431–440, 2003.
- 10 H. J. Fraser and J. A. George. Integrated container loading software for pulp and paper industry. *European Journal of Operational Research*, 77(3):466–474, 1994.
- 11 M. Goldberg. Packing of 14, 16, 17 and 20 circles in a circle. *Mathematics Magazine*, 44:134–139, 1971.
- 12 R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, and P.R.J. Östergård. Dense Packings of Congruent Circles in a Circle. *Discrete Mathematics*, 181:139–154, 1998.
- 13 M. Hifi and R. M’hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009. Article ID 150624.
- 14 P. Hokama, F. K. Miyazawa, and R. C. S. Schouery. A bounded space algorithm for online circle packing. *Information Processing Letters*, 116(5):337–342, May 2016.
- 15 S. Kravitz. Packing cylinders into cylindrical containers. *Mathematics Magazine*, 40:65–71, 1967.
- 16 R. J. Lang. A computational algorithm for origami design. *Proceedings of the Twelfth Annual Symposium on Computational Geometry (SoCG)*, pages 98–105, 1996.
- 17 J. Y. T. Leung, T. W. Tam, C. S. Wong, G. H. Young, and F. Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.
- 18 B.D. Lubachevsky and R.L. Graham. Curved Hexagonal Packings of Equal Disks in a Circle. *Discrete & Computational Geometry*, 18:179–194, 1997.
- 19 H. Melissen. Densest Packing of Eleven Congruent Circles in a Circle. *Geometriae Dedicata*, 50:15–25, 1994.
- 20 F. K. Miyazawa, L. L. C. Pedrosa, R. C. S. Schouery, M. Sviridenko, and Y. Wakabayashi. Polynomial-time approximation schemes for circle packing problems. In *Proceedings of the 22nd European Symposium on Algorithms (ESA)*, pages 713–724, 2014.
- 21 J. W. Moon and L. Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.
- 22 S. Morr. Split Packing: An Algorithm for Packing Circles with Optimal Worst-Case Density. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.
- 23 N. Oler. A finite packing problem. *Canadian Mathematical Bulletin*, 4:153–155, 1961.

- 24 R. Peikert, D. Würtz, M. Monagan, and C. de Groot. Packing circles in a square: A review and new results. In *Proceedings of the 15th IFIP Conference*, pages 45–54, 1992.
- 25 G.E. Reis. Dense Packing of Equal Circles within a Circle. *Mathematics Magazine*, issue 48:33–37, 1975.
- 26 E. Specht. Packomania, 2015. URL: <http://www.packomania.com/>.
- 27 K. Sugihara, M. Sawai, H. Sano, D.-S. Kim, and D. Kim. Disk packing for the estimation of the size of a wire bundle. *Japan Journal of Industrial and Applied Mathematics*, 21(3):259–278, 2004.
- 28 P. G. Szabó, M. C. Markót, T. Csentes, E. Specht, L. G. Casado, and I. García. *New Approaches to Circle Packing in a Square*. Springer US, 2007.
- 29 H. Wang, W. Huang, Q. Zhangn, and D. Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440–453, September 2002.
- 30 D. Würtz, M. Monagan., and R. Peikert. The history of packing circles in a square. *Maple Technical Newsletter*, page 35–42, 1994.



# Semi-Algebraic Colorings of Complete Graphs

**Jacob Fox**

Stanford University, 450 Serra Mall, Building 380, Stanford, CA 94305, USA  
jacobfox@stanford.edu

**János Pach**

École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland,  
Rényi Institute, Hungarian Academy of Sciences, Budapest, Hungary  
pach@cims.nyu.edu

**Andrew Suk**

University of California San Diego, 9500 Gilman Dr, La Jolla, CA 92093, USA  
asuk@ucsd.edu

---

## Abstract

We consider  $m$ -colorings of the edges of a complete graph, where each color class is defined semi-algebraically with bounded complexity. The case  $m = 2$  was first studied by Alon et al., who applied this framework to obtain surprisingly strong Ramsey-type results for intersection graphs of geometric objects and for other graphs arising in computational geometry. Considering larger values of  $m$  is relevant, e.g., to problems concerning the number of distinct distances determined by a point set.

For  $p \geq 3$  and  $m \geq 2$ , the classical Ramsey number  $R(p; m)$  is the smallest positive integer  $n$  such that any  $m$ -coloring of the edges of  $K_n$ , the complete graph on  $n$  vertices, contains a monochromatic  $K_p$ . It is a longstanding open problem that goes back to Schur (1916) to decide whether  $R(p; m) = 2^{O(m)}$ , for a fixed  $p$ . We prove that this is true if each color class is defined semi-algebraically with bounded complexity, and that the order of magnitude of this bound is tight. Our proof is based on the Cutting Lemma of Chazelle *et al.*, and on a Szemerédi-type regularity lemma for multicolored semi-algebraic graphs, which is of independent interest. The same technique is used to address the semi-algebraic variant of a more general Ramsey-type problem of Erdős and Shelah.

**2012 ACM Subject Classification** Mathematics of computing → Combinatoric problems

**Keywords and phrases** Semi-algebraic graphs, Ramsey theory, regularity lemma

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.36

**Funding** *Jacob Fox*: Supported by a Packard Fellowship and by NSF CAREER award DMS 1352121.  
*János Pach*: Supported by Swiss National Science Foundation grants 200020-162884 and 200021-165977.

*Andrew Suk*: Supported by an NSF CAREER award and an Alfred Sloan Fellowship.

## 1 Introduction

The Ramsey number  $R(p; m)$  is the smallest integer  $n$  such that any  $m$ -coloring on the edges of the complete  $n$ -vertex graph contains a monochromatic copy of  $K_p$ . The existence of  $R(p; m)$  follows from the celebrated theorem of Ramsey [18] from 1930, and for the special case when  $p = 3$ , Issai Schur proved the existence of  $R(3; m)$  in 1916 in his work related to Fermat’s Last Theorem [19]. He showed that

$$\Omega(2^m) \leq R(3; m) \leq O(m!).$$

While the upper bound has remained unchanged over the last 100 years, the lower bound was successively improved and the current record is  $R(3; m) \geq \Omega(3.199^m)$  due to Xiaodong *et al.* [22]. It is a major open problem in Ramsey theory, for which Erdős offered some prize money, to close the gap between the lower and upper bounds for  $R(3; m)$ .



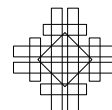
© Jacob Fox, János Pach, and Andrew Suk;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 36; pp. 36:1–36:12

Leibniz International Proceedings in Informatics



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





In this paper, we study edge-colorings of complete graphs where each color class is defined algebraically with bounded complexity. Over the last decade, several researchers have shown that some of the classical theorems in extremal combinatorics can be significantly improved if the underlying graphs are intersection graphs of geometric objects of bounded “description complexity” or bounded VC-dimension, graphs of incidences between points and hyperplanes, distance graphs, or, more generally, semi-algebraic graphs [1, 10, 5, 9, 20]. To make this statement more precise, we need to introduce some terminology. Let  $V$  be an ordered point set in  $\mathbb{R}^d$ , and let  $E \subset \binom{V}{2}$ . We say that  $E$  is a *semi-algebraic* relation on  $V$  with *complexity* at most  $t$  if there are at most  $t$  polynomials  $g_1, \dots, g_s \in \mathbb{R}[x_1, \dots, x_{2d}]$ ,  $s \leq t$ , of degree at most  $t$  and a Boolean formula  $\Phi$  such that for vertices  $u, v \in V$  such that  $u$  comes before  $v$  in the ordering,

$$(u, v) \in E \quad \Leftrightarrow \quad \Phi(g_1(u, v) \geq 0; \dots; g_s(u, v) \geq 0) = 1.$$

At the evaluation of  $g_\ell(u, v)$ , we substitute the variables  $x_1, \dots, x_d$  with the coordinates of  $u$ , the variables  $x_{d+1}, \dots, x_{2d}$  with the coordinates of  $v$ . We may assume that the semi-algebraic relation  $E$  is *symmetric*, i.e., for all points  $u, v \in \mathbb{R}^d$ ,  $(u, v) \in E$  if and only if  $(v, u) \in E$ . Indeed, given such an ordered point set  $V \subset \mathbb{R}^d$  and a not necessarily symmetric semi-algebraic relation  $E$  of complexity at most  $t$ , we can define  $V^* \subset \mathbb{R}^{d+1}$  with points  $(v, i)$  where  $v \in V$  and  $v$  is the  $i$ th smallest element in the given ordering of  $V$ . Then we can define a symmetric semi-algebraic relation  $E^*$  on the pairs of  $V^*$  with complexity at most  $2t + 2$ , by comparing the value of the last coordinates of the two points, and checking the relation  $E$  using the first  $d$  coordinates of the two points. We will therefore assume throughout this paper that all semi-algebraic relations we consider are symmetric, and the vertices are not ordered. Hence, all edges are unordered and we denote  $uv = \{u, v\}$ . We also assume that the dimension  $d$  and complexity  $t$  are fixed parameters, and  $n = |V|$  tends to infinity.

Let  $R_{d,t}(p; m)$  be the minimum  $n$  such that every  $n$ -element point set  $V$  in  $\mathbb{R}^d$  equipped with  $m$  semi-algebraic binary relations (edge-colorings)  $E_1, \dots, E_m \subset \binom{V}{2}$ , each of complexity at most  $t$ , where  $E_1 \cup \dots \cup E_m = \binom{V}{2}$ , contains a subset  $S \subset V$  of size  $p$  such that  $\binom{S}{2} \subset E_k$  for some  $k$ . Note that the relations  $E_i$  are not necessarily disjoint, that is, an edge  $uv$  may have several colors. Clearly  $R_{d,t}(p; m) \leq R(p; m)$ . It was known that for fixed  $d, t \geq 1$ ,  $R_{d,t}(3; m) = 2^{O(m \log \log m)}$ , which is much smaller than Schur’s bound  $R(3; m) = O(m!)$  mentioned in the first paragraph of the Introduction; see [20]. In this paper, we completely settle Schur’s problem for semi-algebraic graphs, by showing that in this setting Schur’s lower bound (which is semi-algebraic with bounded complexity) is tight. In fact, we prove this in a more general form, for any  $p \geq 3$ .

► **Theorem 1.** *For fixed integers  $d, t \geq 1$  and  $p \geq 3$ , we have*

$$R_{d,t}(p; m) = 2^{O(m)}.$$

Our proof uses geometric techniques and is based on the Cutting Lemma of Chazelle, Edelsbrunner, Guibas, and Sharir [3] described in Section 2.

Edge-colorings of semi-algebraic graphs with  $m$  colors can be used, e.g., for studying problems concerning the number of distinct distances determined by a point set; see [11]. One can explore the fact that multicolored semi-algebraic graphs have a very nice structural characterization, reminiscent of Szemerédi’s classic regularity lemma for general graphs [21], but possessing much stronger homogeneity properties. Our next theorem provides such a characterization, which is of independent interest. To state our result, we need some notation and terminology.

A partition is called *equitable* if any two parts differ in size by at most one. According to Szemerédi’s lemma, for every  $\varepsilon > 0$  there is a  $K = K(\varepsilon)$  such that the vertex set of every graph has an equitable partition into at most  $K$  parts such that all but at most an  $\varepsilon$ -fraction of the pairs of parts are  $\varepsilon$ -regular.<sup>1</sup> It follows from Szemerédi’s proof that  $K(\varepsilon)$  may be taken to be an exponential tower of 2s of height  $\varepsilon^{-O(1)}$ . Gowers [13] used a probabilistic construction to show that such an enormous bound is indeed necessary.

Alon *et al.* [1] (see also Fox, Gromov *et al.* [8]) established a strengthening of the regularity lemma for point sets in  $\mathbb{R}^d$  equipped with a semi-algebraic relation  $E$ . It was shown in [1] that for any semi-algebraic graph of bounded complexity defined on the vertex set  $V \subset \mathbb{R}^d$  (that is, for any semi-algebraic binary relation  $E \subset \binom{V}{2}$ ),  $V$  has an equitable partition into a bounded number of parts such that all but at most an  $\varepsilon$ -fraction of the pairs of parts  $(V_1, V_2)$  behave not only regularly, but *homogeneously* in the sense that either  $V_1 \times V_2 \subseteq E$  or  $V_1 \times V_2 \cap E = \emptyset$ . The first proof of this theorem was essentially qualitative: it gave a poor estimate for the number of parts in such a partition. Fox, Pach, and Suk [10] gave a stronger quantitative form of this result, showing that the number of parts can be taken to be polynomial in  $1/\varepsilon$ .

Let  $V$  be an  $n$ -element point set in  $\mathbb{R}^d$  equipped with  $m$  semi-algebraic relations  $E_1, \dots, E_m$  such that  $E_1 \cup \dots \cup E_m = \binom{V}{2}$  of bounded complexity. In other words, suppose that the edges of the complete graph on  $V$  are colored with  $m$  colors, where each color class is semi-algebraic. Then, for any  $\varepsilon > 0$ , an  $m$ -fold repeated application of the result of Fox, Pach, and Suk [10] gives an equitable partition of  $V$  into at most  $K \leq (1/\varepsilon)^{cm}$  parts such that all but an  $\varepsilon$ -fraction of the pairs of parts are complete with respect to some relation  $E_k$ , i.e., all edges between the two parts are of color  $k$ , for some  $k$ . In Section 4, we strengthen this result by showing that the number of parts can be taken to be *polynomial* in  $m/\varepsilon$ .

► **Theorem 2.** *For any positive integers  $d, t \geq 1$  there exists a constant  $c = c(d, t) > 0$  with the following property. Let  $0 < \varepsilon < 1/2$  and let  $V$  be an  $n$ -element point set in  $\mathbb{R}^d$  equipped with semi-algebraic relations  $E_1, \dots, E_m$  such that each  $E_k$  has complexity at most  $t$  and  $\binom{V}{2} = E_1 \cup \dots \cup E_m$ . Then  $V$  has an equitable partition  $V = V_1 \cup \dots \cup V_K$  into at most  $4/\varepsilon \leq K \leq (m/\varepsilon)^c$  parts such that all but an  $\varepsilon$ -fraction of the pairs of parts are complete with respect to some relation  $E_k$ .*

In Section 5, we apply this result to solve a problem of Erdős and Shelah [6] in the semi-algebraic setting. Let  $d, t, p, q, n$  be positive integers,  $p \geq 3$ , and  $2 \leq q \leq \binom{p}{2}$ . Let  $f_{d,t}(n, p, q)$  be the minimum  $m$  such that there exists a semi-algebraic  $m$ -coloring of the edges of the complete graph of  $n$  vertices (with parameters  $d$  and  $t$ , as above) with the property that each edge has exactly one color and any set of  $p$  vertices induce at least  $q$  distinct colors. Notice that here we must assume that the color classes  $E_i$  are disjoint, since otherwise  $f_{d,t}(n, p, q) = q$  by assigning all  $q$  colors to every edge. Our next theorem precisely determines the smallest  $q$  for a given  $p$ , where  $f_{d,t}(n, p, q)$  changes from a  $\log n$  to a power of  $n$ .

► **Theorem 3.** *For fixed integers  $d, t \geq 1$ , there is a  $c = c(d, t) > 0$  such that for  $p \geq 3$ , we have*

$$f_{d,t}(n, p, \lceil \log p \rceil + 1) \geq \Omega\left(n^{\frac{1}{c \log^2 p}}\right).$$

Moreover, for  $t \geq 4$ ,

$$f_{d,t}(n, p, \lceil \log p \rceil) \leq O(\log n).$$

<sup>1</sup> For a pair  $(V_i, V_j)$  of vertex subsets,  $e(V_i, V_j)$  denotes the number of edges in the graph running between  $V_i$  and  $V_j$ . The density  $d(V_i, V_j)$  is defined as  $\frac{e(V_i, V_j)}{|V_i||V_j|}$ . The pair  $(V_i, V_j)$  is called  $\varepsilon$ -regular if for all  $V'_i \subset V_i$  and  $V'_j \subset V_j$  with  $|V'_i| \geq \varepsilon|V_i|$  and  $|V'_j| \geq \varepsilon|V_j|$ , we have  $|d(V'_i, V'_j) - d(V_i, V_j)| \leq \varepsilon$ .

In [11], we studied a geometric instance of this problem, where every set of  $p$  points induces at least  $q$  *distinct distances*.

Our paper is organized as follows. In the next section, we describe the Cutting Lemma of Chazelle *et al.*, which is the main geometric tool used in all proofs. In Section 3, we establish Theorem 1. Section 4 contains the proof of our multicolored semi-algebraic regularity lemma, Theorem 2, which is applied in the following section to deduce Theorem 3. We end this paper with some concluding remarks.

## 2 The cutting lemma

The main tool we use to prove Theorems 1 and 2 is commonly referred to as the *cutting lemma*, which we now recall. A set  $\Delta \subset \mathbb{R}^d$  is *semi-algebraic* if there are polynomials  $g_1, \dots, g_t$  and a boolean formula  $\Phi$  such that

$$A = \{x \in \mathbb{R}^d : \Phi(g_1(x) \geq 0; \dots; g_t(x) \geq 0) = 1\}.$$

We say that a semi-algebraic set in  $d$ -space has *description complexity* at most  $t$  if the number of inequalities is at most  $t$ , and each polynomial  $g_i$  has degree at most  $t$ . Let  $\sigma \subset \mathbb{R}^d$  be a *surface* in  $\mathbb{R}^d$ , that is,  $\sigma$  is the zero set of some polynomial  $h \in \mathbb{R}[x_1, \dots, x_d]$ . The *degree* of a surface  $\sigma = \{x \in \mathbb{R}^d : h(x) = 0\}$  is the degree of the polynomial  $h$ . We say that the surface  $\sigma \subset \mathbb{R}^d$  *crosses* a semi-algebraic set  $\Delta$  if  $\sigma \cap \Delta \neq \emptyset$  and  $\Delta \not\subset \sigma$ .

Let  $\Sigma$  be a collection of surfaces in  $\mathbb{R}^d$ , each having bounded degree. A  $(1/r)$ -*cutting* for  $\Sigma$  is a family  $\Psi$  of disjoint (possibly unbounded) semi-algebraic sets of bounded complexity such that

1. each  $\Delta \in \Psi$  is crossed by at most  $|\Sigma|/r$  surfaces from  $\Sigma$ , and
2. the union of all  $\Delta \in \Psi$  is  $\mathbb{R}^d$ .

In [3], Chazelle *et al.* (see also [14]) proved the following.

► **Lemma 4** (Cutting lemma). *Let  $\Sigma$  be a multiset of  $N$  surfaces in  $\mathbb{R}^d$ , each surface having degree at most  $t$ , and let  $r$  be an integer parameter such that  $1 \leq r \leq N$ . Then there is a constant  $c_1 = c_1(d, t)$  such that  $\Sigma$  admits a  $(1/r)$ -cutting  $\Psi$ , where  $|\Psi| \leq c_1 r^{2d}$ , and each semi-algebraic set  $\Delta \in \Psi$  has complexity at most  $c_1$ .*

We note that the original statement of Chazelle *et al.* [3] and Koltun [14] is stronger. Namely, they also guarantee that the number of cells in the cutting  $\Psi$  is at most  $r^{2d-4+\epsilon}$  for  $d \geq 4$ . Here, for simplicity, we use the weaker bound of  $c_1 r^{2d}$ , as stated above.

## 3 Multicolor Ramsey numbers for small cliques – Proof of Theorem 1

Theorem 1 will easily follow from Theorem 5 below. For integers  $p_1, \dots, p_m \geq 2$ ,  $d, t \geq 1$ , let  $R_{d,t}(p_1, \dots, p_m)$  be the minimum integer  $n$  with the following property. Every complete graph  $K_n$ , whose  $n$  vertices lie in  $\mathbb{R}^d$  and whose edges are colored with  $m$  colors such that each color class is defined by a semi-algebraic relation of description complexity  $t$ , contains a monochromatic copy of  $K_{p_k}$  in color  $k$  for some  $1 \leq k \leq m$ .

► **Theorem 5.** *For any  $d, t \geq 1$  and  $p \geq 3$ , there exists a constant  $c = c(d, t, p)$  satisfying the following condition. For any  $m$  integers  $p_1, \dots, p_m \leq p$ , we have*

$$R_{d,t}(p_1, \dots, p_m) \leq 2^c \sum_{k=1}^m p_k.$$

**Proof.** Fix  $d, t \geq 1, p \geq 3$  and set  $c = c(d, t, p)$  to be a large constant that will be determined later. We will show that  $R_{d,t}(p_1, \dots, p_m) \leq 2^c \sum_{k=1}^m p_k$  by induction on  $s = \sum_{k=1}^m p_k$ . The base case  $s \leq 10 \cdot 2^{10dtp}$  follows for  $c$  sufficiently large.

Now assume that the statement holds for  $s' < s$ . Set  $n = 2^{cs}$  and let  $V$  be an  $n$ -element point set in  $\mathbb{R}^d$  equipped with semi-algebraic relations  $E_1, \dots, E_m \subset \binom{V}{2}$  such that  $\binom{V}{2} = E_1 \cup \dots \cup E_m$  and each  $E_k$  has complexity at most  $t$ . Recall that an edge  $uv$  may have several colors. We will show that there is a subset  $S \subset V$  of size  $p_k$  such that  $\binom{S}{2} \subset E_k$  for some  $k \in [m]$ , in other words, we will find a monochromatic copy of  $K_{p_k}$  in color  $k$  for some  $k \in [m]$ . Throughout the proof, we will let  $c_1$  be as defined in Lemma 4.

For each relation  $E_k$ , there are  $t$  polynomials  $g_{k,1}, \dots, g_{k,t}$  of degree at most  $t$ , and a Boolean function  $\Phi_k$  such that

$$uv \in E_k \iff \Phi_k(g_{k,1}(u, v) \geq 0, \dots, g_{k,t}(u, v) \geq 0) = 1.$$

For  $1 \leq k \leq m, 1 \leq \ell \leq t, v \in V$ , we define the surface  $\sigma_{k,\ell}(v) = \{x \in \mathbb{R}^d : g_{k,\ell}(v, x) = 0\}$ .

Before we continue, let us briefly sketch the idea of the proof. We start by applying Lemma 4 (the cutting lemma) to  $\Sigma = \{\sigma_{k,\ell}(v) : k \in [m], \ell \in [t], v \in V\}$ , the set of surfaces which determines the neighborhoods of each vertex, and obtain a space partition which induces a partition of the vertex set  $V = V_1 \cup \dots \cup V_K$ . If there is a “large” part  $V_j$  with many distinct colors appearing in  $V_j \times (V \setminus V_j)$ , then we show that  $V_j$  induces few distinct colors, and by induction we can find a monochromatic copy of  $K_{p_k}$  for some  $k \in [m]$ . If none of the “large” parts has the above property, the colors of nearly all edges can be defined by much fewer polynomial inequalities, i.e., by a much smaller set of surfaces  $\Sigma' \subset \Sigma$ . Now we can repeat.

In what follows, we spell out these ideas in full detail. Set  $m_0 = m$  and define  $m_i = 4d \log(c_1 m_{i-1} t)$  for  $i > 0$ . We will establish the following claim.

▷ **Claim 6.** Let  $V$  and  $E_1, \dots, E_m \subset \binom{V}{2}$  be defined as above. Then for  $i \geq 0$  we will recursively find either

1. a monochromatic copy of  $K_{p_k}$  in color  $k$  for some  $k \in [m]$ , or
2. a function  $\chi_i : V \rightarrow 2^{[m]}$  such that  $|\chi_i(v)| \leq m_i$ , and the number of edges  $uv \in \binom{V}{2}$  with the property that for one of its endpoints, say  $u$ , no color assigned to  $uv$  belongs to  $\chi_i(u)$ , is at most  $\frac{4n^2}{tm_{i-1}}$ . We will refer to these edges as bad at stage  $i$ . All edges that are not bad are called good at this stage, meaning that, there is a color  $k$  appearing on  $uv$  such that  $k \in \chi_i(u)$ , and there is a color  $k'$  appearing on  $uv$  such that  $k' \in \chi_i(v)$ .

**Proof.** We start by setting  $\chi_0(v) = [m]$  for all  $v \in V$ , and  $m_0 = m$ . Having found  $\chi_i$  with the properties above, we will produce  $\chi_{i+1}$  as follows. We have  $m_{i+1} = 4d \log(c_1 m_i t)$ , and let us assume that  $m_i > (8c_1 dtp)^2$ . Hence, there are at most  $\frac{4n^2}{tm_{i-1}}$  bad edges. Let  $\Sigma$  be the set of surfaces  $\sigma_{k,\ell}(v)$ , where  $v \in V, k \in \chi_i(v)$ , and  $1 \leq \ell \leq t$ . This implies that  $|\Sigma| \leq nm_i t$ .

We apply Lemma 4 to  $\Sigma$  with parameter  $r = (tm_i)^2$  to obtain a  $(1/(tm_i)^2)$ -cutting  $\Psi = \{\Delta_1, \Delta_2, \dots, \Delta_{K_0}\}$ , such that  $K_0 \leq c_1 (tm_i)^{4d}$ . Hence, we have a partition  $\mathcal{P}_0 : V = V_1 \cup \dots \cup V_{K_0}$ , where  $V_j = V \cap \Delta_j$  for  $\Delta_j \in \Psi$ . For each part  $V_j$  of size greater than  $2n/(tm_i)$ , we (arbitrarily) partition  $V_j$  into parts of size  $\lfloor 2n/(tm_i) \rfloor$  and possibly one additional part of size less than  $2n/(tm_i)$ . Let  $\mathcal{P} : V = V_1 \cup \dots \cup V_K$  be the resulting partition, where  $K \leq 2c_1 (tm_i)^{4d}$  and  $|V_j| \leq 2n/(tm_i)$  for all  $j$ .

Now we define  $\chi_{i+1}(v)$  for all  $v \in V$ .

**Case 1.** If  $v \in V_j$  for some  $V_j$  with  $|V_j| < \frac{n}{2c_1 (tm_i)^{4d+1}}$ , we set  $\chi_{i+1}(v) = \emptyset$ .

**Case 2.** Suppose  $v \in V_j$  such that  $|V_j| \geq \frac{n}{2c_1(tm_i)^{4d+1}}$ . In order to define  $\chi_{i+1}(v)$ , we need some preparation. Let  $\Delta_j \in \Psi$  such that  $V_j \subset \Delta_j$ . We define  $X_j \subset V \setminus V_j$  to be the set of vertices from  $V \setminus V_j$  that gives rise to a surface in  $\Sigma$  that crosses  $\Delta_j$ . By Lemma 4, the cutting lemma, we have  $|X_j| \leq n/(tm_i)$ .

Fix a vertex  $v \in V \setminus \{V_j, X_j\}$ . Since none of the surfaces of the form  $\sigma_{k,\ell}(v)$ , where  $k \in \chi_i(v)$  and  $\ell \in \{1, \dots, t\}$ , cross  $\Delta_j$ , either  $v \times V_j$  is monochromatic with color  $k$  for some  $k \in \chi_i(v)$ , or none of the colors in  $\chi_i(v)$  appear in  $v \times V_j$ . Let  $S_j$  be the set of vertices  $v \in V \setminus \{V_j, X_j\}$  satisfying the former condition and let  $T_j$  denote a set of vertices  $v \in V \setminus \{V_j, X_j\}$  satisfying the latter one. Since there are at most  $4n^2/(tm_{i-1})$  bad edges, we have

$$|T_j| \frac{n}{2c_1(tm_i)^{4d+1}} \leq \frac{4n^2}{tm_{i-1}},$$

which implies

$$|T_j| \leq \frac{8nc_1(tm_i)^{4d+1}}{tm_{i-1}} \leq \frac{n}{tm_i},$$

where the last inequality follows from the fact that  $m_{i-1} = 2^{m_i/4d}/(c_1t)$ , and the assumption  $m_i > (8c_1dtp)^2$ . Now, suppose there are at least  $m_{i+1} = 4d \log(c_1tm_i)$  distinct colors between  $V_j$  and  $S_j$ . Let  $I = \{k_1, \dots, k_{m_{i+1}}\} \subset [m]$  be the set of these  $m_{i+1}$  distinct colors. Then there are  $m_{i+1}$  vertices  $v_1, \dots, v_{m_{i+1}} \in S_j$ , possibly with repetition, such that  $v_w \times V_j$  is monochromatic with color  $k_w \in I$ , for each  $w \in \{1, \dots, m_{i+1}\}$ . Hence, if  $V_j$  contains a monochromatic copy of  $K_{p_k-1}$  in color  $k \in I$ , we would have a monochromatic copy of  $K_{p_k}$  in color  $k$ . On the other hand, if  $V_j$  does not contain a monochromatic copy of  $K_{p_k-1}$  in color  $k$  for no  $k \in I$ , then, using that

$$|V_j| \geq \frac{n}{2c_1(tm_i)^{4d+1}} > 2^{cs-8d \log(c_1m_it)} > 2^{c(s-m_{i+1})} = 2^c \left( \sum_{k \in I} (p_k-1) + \sum_{k \notin I} p_k \right)$$

for a sufficiently large  $c$ , we obtain by induction that there is a monochromatic copy of  $K_{p_k}$  in color  $k$  where  $k \notin I$ .

Therefore, we can assume that the number of distinct colors between  $V_j$  and  $S_j$  is less than  $m_{i+1} = 4d \log(c_1m_it)$ . For every vertex  $v \in V_j$ , define  $\chi_{i+1}(v)$  as the set of all colors that appear on the edges belonging to  $v \times S_j$ .

Now that we have defined  $m_{i+1}$  and  $\chi_{i+1}$  such that  $|\chi_{i+1}(v)| \leq m_{i+1}$  for all  $v \in V$ , it remains to show that the number of edges  $uv \in \binom{V}{2}$  with the property that for one of its endpoints, say  $u$ , no color assigned to  $uv$  belongs to  $\chi(u)$ , is at most  $\frac{4n^2}{tm_i}$ . Let  $B \subset \binom{V}{2}$  be the collection of such edges. Notice that if  $uv \in B$ , then either

1. both  $u$  and  $v$  lie inside the same part in the partition  $\mathcal{P}$ , or
  2.  $u$  or  $v$  lies inside a part  $V_j$  such that  $|V_j| < \frac{n}{2c_1(tm_i)^{4d+1}}$ , or
  3.  $u \in V_j$  with  $|V_j| \geq \frac{n}{2c_1(tm_i)^{4d+1}}$  and  $v \in X_j \cup T_j$ , or
  4.  $v \in V_j$  with  $|V_j| \geq \frac{n}{2c_1(tm_i)^{4d+1}}$  and  $u \in X_j \cup T_j$ .
- Since each part in  $\mathcal{P}$  has at most  $2n/(tm_i)$  vertices, the number of edges of type 1 is at most  $\frac{tm_i}{2} \left( \frac{2n}{tm_i} \right)^2 / 2 = n^2/(tm_i)$ . The number of edges of type 2 is at most

$$\left( \frac{n}{2c_1(tm_i)^{4d+1}} \right) K \cdot n \leq \frac{n^2}{tm_i}.$$

Since  $|X_j|, |T_j| \leq n/(tm_i)$ , the number of edges of types 3 and 4 is at most

$$\sum_j |V_j| \frac{2n}{tm_i} \leq \frac{2n^2}{tm_i}.$$

Hence,  $|B| \leq \frac{4n^2}{tm_i}$ . Therefore, either we have found a monochromatic copy of  $K_{p_k}$  in color  $k$  for some  $k \in [m]$ , or we have found  $m_{i+1}$  and  $\chi_{i+1}$  with the desired properties.  $\triangleleft$

Let  $w$  be the minimum integer such that  $m_w \leq (8c_1dtp)^2$ . Then either we have found a monochromatic copy of  $K_{p_k}$  in color  $k$  for some  $k \in [m]$ , or we have obtained  $m_w$  and  $\chi_w$  with the desired properties. Since there are at most  $4n^2/(tm_{w-1}) < n^2/8$  bad edges, there is a vertex  $v \in V$  incident to at least  $n/2$  good edges. Moreover, since  $|\chi_w(v)| \leq m_w \leq (8c_1dtp)^2$ , at least  $\frac{n}{2(8c_1dtp)^2}$  of these edges incident to  $v$  have color  $k'$  for some color  $k' \in \chi_w(v)$ . Let  $S \subset V$  be the set of endpoints of these edges. If  $S$  contains a monochromatic copy of  $K_{p_{k'}-1}$  in color  $k'$ , then we are done. On the other hand, if  $S$  does not contain a monochromatic copy of  $K_{p_{k'}-1}$  in color  $k'$ , and using the lower bound

$$|S| \geq \frac{n}{2(8c_1dtp)^2} = \frac{2^{cs}}{2(8c_1dtp)^2} \geq 2^{c(\sum_{k \neq k'} p_k + (p_{k'} - 1))},$$

for  $c = c(d, t, p)$  sufficiently large, we conclude by induction that  $S$  contains a monochromatic copy of  $K_{p_k}$  for some  $k \neq k'$ . This completes the proof of Theorem 5.  $\blacktriangleleft$

#### 4 Multicolor semi-algebraic regularity lemma – Proof of Theorem 2

First, we prove the following variant of Theorem 2, which easily implies Theorem 2.

**► Theorem 7.** *For any  $\varepsilon > 0$ , every  $n$ -element point set  $V \subset \mathbb{R}^d$  equipped with semi-algebraic binary relations  $E_1, \dots, E_m \subset \binom{V}{2}$  such that  $\binom{V}{2} = E_1 \cup \dots \cup E_m$  and each  $E_k$  has complexity at most  $t$ , can be partitioned into  $K \leq c_2(\frac{m}{\varepsilon})^{5d^2}$  parts  $V = V_1 \cup \dots \cup V_K$ , where  $c_2 = c_2(d, t)$ , such that*

$$\sum \frac{|V_i||V_j|}{n^2} \leq \varepsilon,$$

where the sum is taken over all pairs  $(i, j)$  such that  $(V_i, V_j)$  is not complete with respect to  $E_k$  for all  $k = 1, \dots, m$ .

**Proof.** For each relation  $E_k$ , let  $g_{k,1}, \dots, g_{k,t} \in \mathbb{R}[x_1, \dots, x_{2d}]$  be polynomials of degree at most  $t$ , and let  $\Phi_k$  be a boolean formula such that

$$uv \in E_k \iff \Phi_k(g_{k,1}(u, v) \geq 0; \dots; g_{k,t}(u, v) \geq 0) = 1.$$

For each point  $x \in \mathbb{R}^d$ ,  $k \in \{1, \dots, m\}$ , and  $\ell \in \{1, \dots, t\}$ , we define the surface

$$\sigma_{k,\ell}(x) = \{y \in \mathbb{R}^d : g_{k,\ell}(x, y) = 0\}.$$

Let  $\Sigma$  be the family of  $tmn$  surfaces in  $\mathbb{R}^d$  defined by

$$\Sigma = \{\sigma_{k,\ell}(u) : u \in V, 1 \leq k \leq m, 1 \leq \ell \leq t\}.$$

We apply Lemma 4 to  $\Sigma$  with parameter  $r = tm/\varepsilon$  to obtain a  $(1/r)$ -cutting  $\Psi$ , where  $|\Psi| = s \leq c_1(\frac{tm}{\varepsilon})^{2d}$ , such that each semi-algebraic set  $\Delta_i \in \Psi$  has complexity at most  $c_1$ , where  $c_1$  is defined in Lemma 4. Hence, at most  $tmn/r = \varepsilon n$  surfaces from  $\Sigma$  cross  $\Delta_i$  for every  $i$ . This implies that at most  $\varepsilon n$  points in  $V$  give rise to at least one surface in  $\Sigma$  that crosses  $\Delta_i$ .

Let  $U_i = V \cap \Delta_i$  for each  $i \leq s$ . We now partition  $\Delta_i$  as follows. For  $k \in \{1, \dots, m\}$  and  $j \in \{1, \dots, s\}$ , define  $\Delta_{i,j,k} \subset \mathbb{R}^d$  by

$$\Delta_{i,j,k} = \{x \in \Delta_i : \sigma_{k,1}(x) \cup \dots \cup \sigma_{k,t}(x) \text{ crosses } \Delta_j\}.$$

## 36:8 Semi-Algebraic Colorings of Complete Graphs

That is,  $\Delta_{i,j,k}$  will correspond to the vertices in  $\Delta_i$  that may not be homogeneous to the set of vertices in  $\Delta_j$  with respect to color  $k$ .

► **Observation 8.** For any  $i, j$ , and  $k$ , the semi-algebraic set  $\Delta_{i,j,k}$  has complexity at most  $c_3 = c_3(d, t)$ .

**Proof.** Set  $\sigma_k(x) = \sigma_{k,1}(x) \cup \dots \cup \sigma_{k,t}(x)$ , which is a semi-algebraic set with complexity at most  $c_4 = c_4(d, t)$ . Then

$$\Delta_{i,j,k} = \left\{ x \in \Delta_i : \begin{array}{l} \exists y_1 \in \mathbb{R}^d \text{ s.t. } y_1 \in \sigma_k(x) \cap \Delta_j, \text{ and} \\ \exists y_2 \in \mathbb{R}^d \text{ s.t. } y_2 \in \Delta_j \setminus \sigma_k(x). \end{array} \right\}.$$

We can apply quantifier elimination (see Theorem 2.74 in [2]) to make  $\Delta_{i,j,k}$  quantifier-free, with description complexity at most  $c_3 = c_3(d, t)$ . ◀

Set  $\mathcal{F}_i = \{\Delta_{i,j,k} : 1 \leq k \leq m, 1 \leq j \leq s\}$ . We partition the points in  $U_i$  into equivalence classes, where two points  $u, v \in U_i$  are equivalent if and only if  $u$  belongs to the same members of  $\mathcal{F}_i$  as  $v$  does. Since  $\mathcal{F}_i$  gives rise to at most  $c_3|\mathcal{F}_i|$  polynomials of degree at most  $c_3$ , by the Milnor-Thom theorem (see [16] Chapter 6), the number of distinct sign patterns of these  $c_3|\mathcal{F}_i|$  polynomials is at most  $(50c_3(c_3|\mathcal{F}_i|))^d$ . Hence, there is a constant  $c_5 = c_5(d, t)$  such that  $U_i$  is partitioned into at most  $c_5(ms)^d$  equivalence classes. After repeating this procedure to each  $U_i$ , we obtain a partition of our point set  $V = V_1 \cup \dots \cup V_K$  with

$$K \leq sc_5(ms)^d = c_5m^d s^{d+1} \leq c_5t^{2d(d+1)}c_1^{d+1} \left(\frac{m}{\varepsilon}\right)^{5d^2} = c_2 \left(\frac{m}{\varepsilon}\right)^{5d^2},$$

where we define  $c_2 = c_5t^{2d(d+1)}c_1^{d+1}$ .

For fixed  $i$ , consider the part  $V_i$ . Then there is a semi-algebraic set  $\Delta_{w_i}$  obtained from Lemma 4 such that  $U_{w_i} = V \cap \Delta_{w_i}$  and  $V_i \subset U_{w_i} \subset \Delta_{w_i}$ . Now consider all other parts  $V_j$  such that not all of their elements are related to every element of  $V_i$  with respect to any relation  $E_k$  where  $1 \leq k \leq m$ . Then each point  $u \in V_j$  gives rise to a surface in  $\Sigma$  that crosses  $\Delta_{w_i}$ . By Lemma 4, the total number of such points in  $V$  is at most  $\varepsilon n$ . Therefore, we have

$$\sum_j |V_i||V_j| = |V_i| \sum_j |V_j| \leq |V_i|\varepsilon n,$$

where the sum is over all  $j$  such that  $V_i \times V_j$  is not contained in the relation  $E_k$  for any  $k$ . Summing over all  $i$ , we have

$$\sum_{i,j} |V_i||V_j| \leq \varepsilon n^2,$$

where the sum is taken over all pairs  $i, j$  such that  $(V_i, V_j)$  is not complete with respect to  $E_k$  for all  $k$ . ◀

**Proof of Theorem 2.** Apply Theorem 7 with approximation parameter  $\varepsilon/2$ . Hence, there is a partition  $\mathcal{Q} : V = U_1 \cup \dots \cup U_{K'}$  into  $K' \leq (m/\varepsilon)^c$  parts with  $c = c(d, t)$  and  $\sum |U_i||U_j| \leq (\varepsilon/2)|V|^2$ , where the sum is taken over all pairs  $(i, j)$  such that  $(U_i, U_j)$  is not complete with respect to  $E_k$  for all  $k$ .

Let  $K = 8\varepsilon^{-1}K'$ . Partition each part  $U_i$  into parts of size  $|V|/K$  and possibly one additional part of size less than  $|V|/K$ . Collect these additional parts and divide them into parts of size  $|V|/K$  to obtain an equitable partition  $\mathcal{P} : V = V_1 \cup \dots \cup V_K$  into  $K$  parts. The



number of vertices of  $V$  which are in parts  $V_i$  that are not contained in a part of  $\mathcal{Q}$  is at most  $K'|V|/K$ . Hence, the fraction of pairs  $V_i \times V_j$  with not all  $V_i, V_j$  are subsets of parts of  $\mathcal{Q}$  is at most  $2K'/K = \varepsilon/4$ . As  $\varepsilon/2 + \varepsilon/4 < \varepsilon$ , we obtain that less than an  $\varepsilon$ -fraction of the pairs of parts of  $\mathcal{P}$  are not complete with respect to any relation  $E_1, \dots, E_m$ . ◀

### 5 Generalized Ramsey numbers for semi-algebraic colorings – Proof of Theorem 3

Due to the lack of understanding of the classical Ramsey number  $R(p; m)$ , Erdős and Shelah (see [6]) introduced the following generalization, which was studied by Erdős and Gyárfás in [7].

▶ **Definition 9.** For integers  $p$  and  $q$  with  $2 \leq q \leq \binom{p}{2}$ , a  $(p, q)$ -coloring is an edge-coloring of a complete graph in which every  $p$  vertices induce at least  $q$  distinct colors.

Let  $f(n, p, q)$  be the minimum integer  $m$  such that there is a  $(p, q)$ -coloring of  $K_n$  with at most  $m$  colors. Here, both  $p$  and  $q$  are considered fixed integers, where  $p \geq 3$ ,  $2 \leq q \leq \binom{p}{2}$ , and  $n$  tends to infinity. Trivially, we have  $f(n, p, \binom{p}{2}) = \binom{n}{2}$ , and at the other end, estimating  $f(n, p, 2)$  is equivalent to estimating  $R(p; m)$  since  $f(n, p, 2)$  is the inverse of  $R(p; m)$ . In particular,

$$\Omega\left(\frac{\log n}{\log \log n}\right) \leq f(n, 3, 2) \leq O(\log n).$$

Erdős and Gyárfás [7] determined certain ranges for  $q \in \{2, 3, \dots, \binom{p}{2}\}$  for which  $f(n, p, q)$  is quadratic, linear, and subpolynomial in  $n$ . In particular, they showed that

$$\Omega\left(n^{\frac{1}{p-2}}\right) \leq f(n, p, p) \leq O\left(n^{\frac{2}{p-1}}\right),$$

which implies that  $f(n, p, q)$  is polynomial in  $n$  for  $q \geq p$ . Surprisingly, estimating  $f(n, p, p-1)$  is much more difficult. They [7] asked for  $p$  fixed if  $f(n, p, p-1) = n^{o(1)}$ . The trivial lower bound is  $f(n, p, p-1) \geq f(n, p, 2) \geq \Omega\left(\frac{\log n}{\log \log n}\right)$ , which was improved by several authors [15, 12], and it is now known [4] that  $f(n, p, p-1) \geq \Omega(\log n)$ . In the other direction, Mubayi [17] found an elegant construction which implies  $f(n, 4, 3) \leq e^{O(\sqrt{\log n})}$ , and later, Conlon *et al.* [4] gave another example which implies  $f(n, p, p-1) \leq e^{(\log n)^{1-1/(p-2)+o(1)}}$ . Hence, it is now known that  $f(n, p, p-1)$  does not grow as a power in  $n$ .

Here, we study the variant of the function  $f(n, p, q)$  for point sets  $V \subset \mathbb{R}^d$  equipped with semi-algebraic relations. Let  $f_{d,t}(n, p, q)$  be the minimum  $m$  such that there is a  $(p, q)$ -coloring of  $K_n$  with  $m$  colors, whose vertices can be chosen as points in  $\mathbb{R}^d$ , and each color class can be defined by a semi-algebraic relation on the point set with complexity at most  $t$ . We note that here we require that each edge receives *exactly* one color. Clearly, we have  $f(n, p, q) \leq f_{d,t}(n, p, q)$ . Theorem 3 stated in the Introduction shows the exact value of  $q$  for which  $f_{d,t}(n, p, q)$  changes from  $\log n$  to a power of  $n$ .

In the rest of this section, we prove Theorem 3. Let  $V$  be a set of points in  $\mathbb{R}^d$  equipped with semi-algebraic relations  $E_1, \dots, E_m$  such that each  $E_k$  has complexity at most  $t$ ,  $\binom{V}{2} = E_1 \cup \dots \cup E_m$ , and  $E_k \cap E_\ell = \emptyset$  for all  $k \neq \ell$ . Let  $S_1, S_2 \subset V$  be  $q$ -element subsets of  $V$ . We say that  $S_1$  and  $S_2$  are *isomorphic*, denoted by  $S_1 \simeq S_2$ , if there is a bijective function  $h : S_1 \rightarrow S_2$  such that for  $u, v \in S_1$  we have  $uv \in E_k$  if and only if  $h(u)h(v) \in E_k$ .

Let  $S \subset V$  be such that  $|S| = 2^s$  for some positive integer  $s$ . We say that  $S$  is *s-layered* if  $s = 1$  or if there is a partition  $S = S_1 \cup S_2$  such that  $|S_1| = |S_2| = 2^{s-1}$ ,  $S_1$  and  $S_2$  are  $(s-1)$ -layered,  $S_1 \simeq S_2$ , and for all  $u \in S_1$  and  $v \in S_2$  we have  $uv \in E_k$  for some fixed  $k$ .



## 36:10 Semi-Algebraic Colorings of Complete Graphs

Notice that given an  $s$ -layered set  $S$ , there are at most  $s$  relations  $E_{k_1}, \dots, E_{k_s}$  such that  $\binom{S}{2} \subset E_{k_1} \cup \dots \cup E_{k_s}$ . Hence, the lower bound in Theorem 3 is a direct consequence of the following result.

► **Theorem 10.** *Let  $s \geq 1$  and let  $V$  be an  $n$ -element point set in  $\mathbb{R}^d$  equipped with semi-algebraic relations  $E_1, \dots, E_m$  such that each  $E_k$  has complexity at most  $t$ ,  $E_1 \cup \dots \cup E_m = \binom{V}{2}$ , and  $E_k \cap E_\ell = \emptyset$  for all  $k \neq \ell$ . If  $m \leq n^{\frac{1}{cs^2}}$ , then there is a subset  $S \subset V$  such that  $|S| = 2^s$  and  $S$  is  $s$ -layered, where  $c = c(d, t)$ .*

**Proof.** We proceed by induction on  $s$ . The base case  $s = 1$  is trivial. For the inductive step, assume that the statement holds for  $s' < s$ . We will specify  $c = c(d, t)$  later. We start by applying Theorem 2 with parameter  $\varepsilon = \frac{1}{m^s}$  to the point set  $V$ , which is equipped with semi-algebraic relations  $E_1, \dots, E_m$ , and obtain an equitable partition  $\mathcal{P} : V = V_1 \cup \dots \cup V_K$ , where

$$K \leq c_2 \left( \frac{m}{\varepsilon} \right)^{5d^2} \leq c_2 m^{10sd^2},$$

and  $c_2 = c_2(d, t)$ . Since all but an  $\varepsilon$  fraction of the pairs of parts in  $\mathcal{P}$  are complete with respect to  $E_k$  for some  $k$ , by Turán's theorem, there are  $m^{s-1} + 1$  parts  $V'_i \in \mathcal{P}$  such that each pair  $(V'_i, V'_j) \in \mathcal{P} \times \mathcal{P}$  is complete with respect to some relation  $E_k$ . Since  $\mathcal{P}$  is an equitable partition, we have  $|V'_i| \geq \frac{n}{c_2 m^{10d^2 s}}$ . By picking  $c = c(d, t)$  sufficiently large, we have

$$|V'_i|^{\frac{1}{c(s-1)^2}} \geq \left( \frac{n}{c_2 m^{10d^2 s}} \right)^{\frac{1}{c(s-1)^2}} \geq m^{\frac{cs^2 - 10c_2 d^2 s}{c(s-1)^2}} \geq m.$$

By the induction hypothesis, each  $V'_i$  contains an  $(s-1)$ -layered set  $S_i$  for  $i \in \{1, \dots, m^{s-1} + 1\}$ . By the pigeonhole principle, there are two  $(s-1)$ -layered sets  $S_i, S_j$  such that  $S_i \simeq S_j$ . Since  $S_i \times S_j \subset E_k$  for some  $k$ , the set  $S = S_i \cup S_j$  is an  $s$ -layered set. This completes the proof. ◀

To prove the upper bound for  $f_{d,t}(n, p, \lceil \log p \rceil)$ , when  $d \geq 1$  and  $t \geq 100$ , it is sufficient to construct a  $2^m$ -element point set  $V \subset \mathbb{R}$  equipped with  $m$  distinct semi-algebraic relations  $E_1, \dots, E_m$  that is  $m$ -layered. More precisely, for each integer  $m \geq 1$ , we construct a set  $V_m$  of  $2^m$  points in  $\mathbb{R}$  equipped with semi-algebraic relations  $E_1, \dots, E_m$  such that

1.  $V_m$  with respect to relations  $E_1, \dots, E_m$  is  $m$ -layered,
2.  $E_1 \cup \dots \cup E_m = \binom{V_m}{2}$  is a partition,
3. each  $E_i$  has complexity at most four, and
4. each  $E_i$  is *shift invariant*, that is  $uv \in E_i$  if and only if  $(u + c, v + c) \in E_i$  for  $c \in \mathbb{R}$ .

We start by setting  $V_1 = \{1, 2\}$  and defining  $E_1 = \{u, v \in V_1 : |u - v| = 1\}$ . Having defined the point set  $V_i$  and relations  $E_1, \dots, E_i$ , we define  $V_{i+1}$  and  $E_{i+1}$  as follows. Let  $C = C(i)$  be a sufficiently large integer such that  $C > 10 \max_{u \in V_i} u$ . Then we have  $V_{i+1} = V_i \cup (V_i + C)$ , where  $V_i + C$  is a translated copy of  $V_i$ . We now define the relation  $E_{i+1}$  by

$$uv \in E_{i+1} \quad \Leftrightarrow \quad C/2 < |u - v| < 2C.$$

Hence,  $V_{i+1}$  with respect to relations  $E_1, \dots, E_{i+1}$  satisfies the properties stated above and is clearly  $(i+1)$ -layered. One can easily check that any set of  $p$  points in  $V_m$  induces at least  $\lceil \log p \rceil$  distinct relations (colors).

Let us remark that the arguments above hold for semi-algebraic relations  $E_1, \dots, E_m$  that are not necessarily disjoint if one defines a  $(p, q)$ -coloring as follows. Given a coloring  $\chi : \binom{V(K_n)}{2} \rightarrow 2^{[m]}$  on the edges of  $K_n$ , where each edge receives *at least* one color among  $[m]$ ,  $\chi$  is a  $(p, q)$ -coloring if for every set  $S \subset V$  of size  $p$ , no matter how you choose one color in  $\chi(uv)$  for each edge  $uv \in \binom{S}{2}$ ,  $S$  will induce at least  $q$  distinct colors.

## 6 Concluding remarks

In [20], it was shown that  $R_{1,t}(3; m) > (1681)^{m/7}$  for  $t > 5$ , thus implying that the upper bound in Theorem 1 is tight up to a constant factor in the exponent. This can be improved as follows. Let  $C(p) = \lim_{m \rightarrow \infty} R(p; m)^{1/m}$ . Note that this limit exists by considering product colorings, but may be finite or infinite. Then for each  $C < C(p)$ , there is a  $t = t(C, p)$ , such that for all  $m$  sufficiently large we have

$$R_{1,t}(p; m) > C^m.$$

Indeed, take a fixed coloring of the edges of  $K_N$  which realizes  $R(p; m_0) > C^{m_0}$ , and recursively blow up this graph by introducing  $m_0$  new colors at each stage. Then this coloring can be realized semi-algebraically in  $\mathbb{R}$  with  $t = O(m_0^2)$  linear constraints for each color class based on distances.

---


## References

- 1 N. Alon, J. Pach, R. Pinchasi, R. Radoičić, and M. Sharir. Crossing patterns of semi-algebraic sets. *J. Combin. Theory Ser. A*, 111:310–326, 2005.
- 2 S. Basu, R. Pollack, and M.-R. Roy. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, Berlin, 2003.
- 3 B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly exponential stratification scheme for real semi-algebraic varieties and its applications. *Theor. Comput. Sci.*, 84:77–105, 1991.
- 4 D. Conlon, J. Fox, C. Lee, and B. Sudakov. The Erdős-Gyárfás problem on generalized Ramsey numbers. *Proc. Lond. Math. Soc.*, 110:1–15, 2015.
- 5 D. Conlon, J. Fox, J. Pach, B. Sudakov, and A. Suk. Ramsey-type results for semi-algebraic relations. *Trans. Amer. Math. Soc.*, 366:5043–5065, 2014.
- 6 P. Erdős. Solved and unsolved problems in combinatorics and combinatorial number theory. *Proc. 12th Southeastern Conf. on Combinatorics, Graph Theory and Computing, (Baton Rouge, La.)*, *Congr. Numer.*, 1:49–62, 1981.
- 7 P. Erdős and A. Gyárfás. A variant of the classical Ramsey problem. *Combinatorica*, 17:459–467, 1997.
- 8 J. Fox, M. Gromov, V. Lafforgue, A. Naor, and J. Pach. Overlap properties of geometric expanders. *J. Reine Angew. Math. (Crelle's Journal)*, 671:49–83, 2012.
- 9 J. Fox, J. Pach, A. Sheffer, A. Suk, and J. Zahl. A semi-algebraic version of Zarankiewicz's problem. *J. Eur. Math. Soc.*, 19:1785–1810, 2017.
- 10 J. Fox, J. Pach, and A. Suk. A polynomial regularity lemma for semi-algebraic hypergraphs and its applications in geometry and property testing. *SIAM J. Comput.*, 45:2199–2223, 2016.
- 11 J. Fox, J. Pach, and A. Suk. More distinct distances under local conditions. *Combinatorica*, 38:501–509, 2018.
- 12 J. Fox and B. Sudakov. Ramsey-type problem for an almost monochromatic  $K_4$ . *SIAM J. Discrete Math.*, 23:155–162, 2008.
- 13 W. T. Gowers. Lower bounds of tower type for Szemerédi's uniformity lemma. *Geom. Funct. Anal.*, 7:322–337, 1997.
- 14 V. Koltun. Almost tight upper bounds for vertical decompositions in four dimensions. *J. ACM*, 51:699–730, 2004.
- 15 A. V. Kostochka and D. Mubayi. When is an almost monochromatic  $K_4$  guaranteed? *Combin. Probab. Comput.*, 17:823–830, 2008.
- 16 J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.
- 17 D. Mubayi. Edge-coloring cliques with three colors on all 4-cliques. *Combinatorica*, 18:293–296, 1998.

## 36:12 Semi-Algebraic Colorings of Complete Graphs

- 18 F. Ramsey. On a problem in formal logic. *Proc. London Math. Soc.*, 30:264–286, 1930.
- 19 I. Schur. Über die Kongruenz  $x^m + y^m = z^m \pmod{p}$ . *Jahresber. Deutch. Math. Verein.*, 25:114–117, 1916.
- 20 A. Suk. Semi-algebraic Ramsey numbers. *J. Combin. Theory Ser. B*, 116:465–483, 2016.
- 21 E. Szemerédi. Regular partitions of graphs. *Problèmes Combinatoires et Théorie des Graphes, Orsay (1976)*, 260:299–401, 1976.
- 22 X. Xiaodong, X. Zheng, G. Exoo, and S.P. Radziszowski. Constructive lower bounds on classical multicolor Ramsey numbers. *Electron. J. Combin.*, 11, 2004.

# Chunk Reduction for Multi-Parameter Persistent Homology

Ulderico Fugacci 

Graz University of Technology, Graz, Austria  
fugacci@tugraz.at

Michael Kerber 

Graz University of Technology, Graz, Austria  
kerber@tugraz.at

---

## Abstract

The extension of persistent homology to multi-parameter setups is an algorithmic challenge. Since most computation tasks scale badly with the size of the input complex, an important pre-processing step consists of simplifying the input while maintaining the homological information. We present an algorithm that drastically reduces the size of an input. Our approach is an extension of the chunk algorithm for persistent homology (Bauer et al., *Topological Methods in Data Analysis and Visualization III*, 2014). We show that our construction produces the smallest multi-filtered chain complex among all the complexes quasi-isomorphic to the input, improving on the guarantees of previous work in the context of discrete Morse theory. Our algorithm also offers an immediate parallelization scheme in shared memory. Already its sequential version compares favorably with existing simplification schemes, as we show by experimental evaluation.

**2012 ACM Subject Classification** Theory of computation → Randomness, geometry and discrete structures; Computing methodologies → Shared memory algorithms

**Keywords and phrases** Multi-parameter persistent homology, Matrix reduction, Chain complexes

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.37

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1812.08580>.

**Funding** Supported by the Austrian Science Fund (FWF) grant number P 29984-N35.

**Acknowledgements** We thank Sara Scaramuccia for initial discussions on this project, Wojciech Chacholski, Michael Lesnick and Francesco Vaccarino for helpful suggestions, and Federico Iuricich for his help on the experimental comparison with [10]. The datasets used in the experimental evaluation are courtesy of the AIM@SHAPE data repository [9].

## 1 Introduction

In the last decades, topology-based tools are gaining a more and more relevant role in the analysis and in the extraction of the core information of unorganized, high-dimensional and potentially large datasets. Thanks to its capability of keeping track of the changes in the homological features of a dataset which evolves with respect to a parameter, *persistent homology* has represented a real game-changer in this field. Recently, an extension of persistent homology called *multi-parameter persistent homology* is drawing the attention of a growing number of researchers. In a nutshell, multi-parameter persistence generalizes the classic persistent homology by studying multivariate datasets which are filtered by two or more (independent) scale parameters. Multi-parameter persistent homology of a dataset cannot be captured by complete discrete invariant [5], but this has not prevented the researchers from defining several descriptors based on multi-parameter persistence [6, 12, 15]. Since

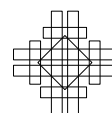


© Ulderico Fugacci and Michael Kerber;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 37; pp. 37:1–37:14  
Leibniz International Proceedings in Informatics



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



these descriptors tend to have high algorithmic complexity, it is a natural pre-processing step to pass to a smaller but equivalent representation of the input complex and to invoke any demanding computation on this smaller representation.

**Contribution.** Inspired by the chunk algorithm [3] for persistent homology, we present a reduction algorithm which for a filtered dataset returns a filtered chain complex having the same multi-parameter persistence but a drastically smaller size. Our approach is based on the observation that even in the absence of a global persistence diagram, local matrix reductions yield pairs of simplices which can be eliminated from the boundary matrix without affecting the homological information. Our approach proceeds in two steps, first identifying such local pairs of simplices, and in a second step manipulating the non-local columns of the boundary matrix such that all indices of locally paired simplices disappear. Both steps permit a parallel implementation with shared memory. We prove that our simplification scheme is optimal in the sense that any chain complex quasi-isomorphic to the input complex must contain at least as many generators as our output complex. Our algorithm yields similar time and space complexity bounds as its one-parameter counterpart. We implemented our algorithm, making use of various techniques that have proven effective for persistence computation, such as the twist reduction [8] and efficient data structure for the columns of a boundary matrix [4]. We experimentally show that our implementation is effective (see next paragraph). For the sake of clarity, our approach is described for the two-parameter case. No constraint prevents the generalization of the proposed method to an arbitrary number of parameters.

**Comparison with related work.** Our work is motivated by a line of research on complex simplification using discrete Morse theory (DMT) [11, 1, 2, 17]. In these works, the idea is to build a discrete gradient locally and to return the resulting Morse complex on critical simplices as a simplification. In analogy to the one-parameter case, our chunk algorithm is an attempt to realize this simplification scheme using persistent homology instead of DMT. This gives more flexibility, as in DMT, the paired cells are constrained to be incident, while this restriction is not present for persistence pairs. Consequently, we are able to prove optimality of our output size in general, while DMT-based approaches only succeeded to give guarantees for special cases such as multi-filtrations of 3D regular grids and of abstract simplicial 2-complexes [16]. In practice, the theoretical benefit in terms of output size is rather marginal, as our experiments show; however, the timings show that our improved theoretical guarantee comes without performance penalty; on the contrary, our algorithm is always faster than the DMT-based approach presented in [17] on all tested examples. We remark, however, that the DMT-based algorithm returns a complex endowed with a Forman gradient as well as the corresponding discrete Morse complex, which can be of potential use for other application domains than computing persistent homology.

A related question is the computation of a *minimal presentation* of a persistence module induced by a simplicial or general chain complex. Roughly speaking, a presentation consists of a finite set of (graded) generators and relations, and the minimal presentation is one with the minimal possible number of generators and relations. Our algorithm does not yield a minimal presentation, however, since more computations are needed to identify generators of the chain complex as generators or as relations. An algorithm by Lesnick and Wright<sup>1</sup> [13] computes such a minimal presentation through matrix reduction in cubic time, carefully choosing a column order to reduce the number of reduction instances. Their algorithm is used

---

<sup>1</sup> <https://www.ima.umn.edu/2017-2018/SW8.13-15.18/27428>

in their software package RIVET [14]. Our contribution can serve as a pre-processing step for their algorithm, reducing the size of the matrix through efficient and possibly parallelized computation and invoking their global reduction step on a much smaller chain complex.

## 2 Background

**Homology.** Fixed a base field  $\mathbb{F}$  and a positive integer  $d$ , let us consider, for each  $0 \leq k \leq d$ , a finite collection of elements denoted as  $k$ -generators (or, equivalently, generators of dimension  $k$ ). A finitely generated chain complex  $C_* = (C_k, \partial_k)$  over  $\mathbb{F}$  is a collection of pairs  $(C_k, \partial_k)$  where:

- $C_k$  is the  $\mathbb{F}$ -vector space spanned by the generators of dimension  $k$ ,
- $\partial_k : C_k \rightarrow C_{k-1}$  is called *boundary map* and it satisfies the property that  $\partial_{k-1}\partial_k = 0$ .

The elements of  $C_k$  are called  $k$ -chains and, by definition, are  $\mathbb{F}$ -linear combinations of the generators of dimension  $k$ . The *support* of a  $k$ -chain is the set of  $k$ -generators whose coefficient in the chain is not zero. In the following, we will simply use the term chain complex in place of finitely generated chain complex. Moreover, we will assume that, given a chain complex, an explicit set of generators is provided.

Given a chain complex  $C_*$ , we denote as  $Z_k := \ker \partial_k$  the space of the  $k$ -cycles of  $C_*$ , and as  $B_k := \text{Im } \partial_{k+1}$  the space of the  $k$ -boundaries of  $C_*$ . The  $k^{\text{th}}$  *homology space* of  $C_*$  is defined as the vector space  $H_k(C_*) := Z_k/B_k$ . The rank  $\beta_k$  of the  $k^{\text{th}}$  homology space of a chain complex  $C_*$  is called the  $k^{\text{th}}$  *Betti number* of  $C_*$ .

A *chain map*  $f_* : C_* \rightarrow D_*$  is a collection of linear maps  $f_k : C_k \rightarrow D_k$  which commutes with the boundary operators of  $C_*$  and of  $D_*$ . A simple example of a chain map is the inclusion map, if  $C_*$  is a subcomplex of  $D_*$ . In general, a chain map  $f_*$  induces linear maps  $H_k(C_*) \rightarrow H_k(D_*)$  for every  $k$ .

Chain complexes allow for capturing the combinatorial and the topological structure of a discretized topological space. Given a finite simplicial complex  $K$ , the chain complex  $C_*$  associated to  $K$  is defined by setting  $C_k$  as the  $\mathbb{F}$ -vector space generated by the  $k$ -simplices of  $K$  and the boundary  $\partial_k(c)$  of a  $k$ -chain  $c$  corresponding to a  $k$ -simplex  $\sigma$  as the collection of the  $(k-1)$ -simplices lying on the geometrical boundary of  $\sigma$ . Consequently, homology of a finite simplicial complex  $K$  is defined as the homology of the chain complex  $C_*$  associated to  $K$ . Intuitively, homology spaces of  $K$  reveal the presence of “holes” in the simplicial complex. The non-null elements of each homology space are cycles, which do not represent the boundary of any collection of simplices of  $K$ . Specifically,  $\beta_0$  counts the number of connected components of  $K$ ,  $\beta_1$  its tunnels and holes, and  $\beta_2$  the shells surrounding voids or cavities.

**Multi-parameter persistent homology.** In the following, we will focus for simplicity on datasets filtered by two independent scale parameters. All definitions and results in this paper can be generalized to more parameters without problems. Let  $p = (p_x, p_y), q = (q_x, q_y) \in \mathbb{R}^2$ , we will write throughout  $p \leq q$  if  $p_x \leq q_x$  and  $p_y \leq q_y$ . Given a chain complex  $C_* = (C_k, \partial_k)$ , let us assume to have an assignment which, for each generator  $g \in C_k$ , returns a value  $v(g) \in \mathbb{R}^2$  such that, for any generator  $g' \in C_{k-1}$ , if  $\langle \partial_k g, g' \rangle \neq 0$ , then  $v(g') \leq v(g)$ . The assignment  $v$  can be extended to a function  $v : C_* \rightarrow \mathbb{R}^2$  assigning to each chain  $c$  the least common upper bound  $v(c)$  of the values  $v(g)$  of the generators in the support of  $c$ . In the following,  $v(c)$  will be called the *value* of  $c$ .

### 37:4 Chunk Reduction for Multi-Parameter Persistent Homology

Given a chain complex  $C_* = (C_k, \partial_k)$  endowed with a value function  $v : C_* \rightarrow \mathbb{R}^2$  and fixed a value  $p \in \mathbb{R}^2$  we define  $C_*^p = (C_k^p, \partial_k^p)$  as the chain complex for which:

- $C_k^p$  is the space of the  $k$ -chains of  $C_k$  having value lower than or equal to  $p$ ,
- $\partial_k^p$  is the restriction of  $\partial_k$  to  $C_k^p$ .

By the definition of  $v$ , for any chain  $c$  of dimension  $k$ , we have that  $v(\partial_k(c)) \leq v(c)$ . So,  $C_*^p$  is well-defined. For  $p, q \in \mathbb{R}^2$  with  $p \leq q$ ,  $C_*^p$  is a chain subcomplex of  $C_*^q$ . For this reason, we denote the collection  $\mathcal{C}$  of the chain complexes  $C_*^p$  with  $p \in \mathbb{R}^2$  as *bifiltered chain complex*.

Given  $p, q \in \mathbb{R}^2$  with  $p \leq q$ , the inclusion map from  $C_*^p$  to  $C_*^q$  induces a linear map between the corresponding homology spaces  $H_k(C_*^p)$  and  $H_k(C_*^q)$ . The *multi-parameter persistence  $k^{\text{th}}$  module*  $H_k(\mathcal{C})$  of a bifiltered chain complex  $\mathcal{C}$  is the collection of the homology spaces  $H_k(C_*^p)$  with  $p$  varying in  $\mathbb{R}^2$  along with all the linear maps induced by the inclusion maps.

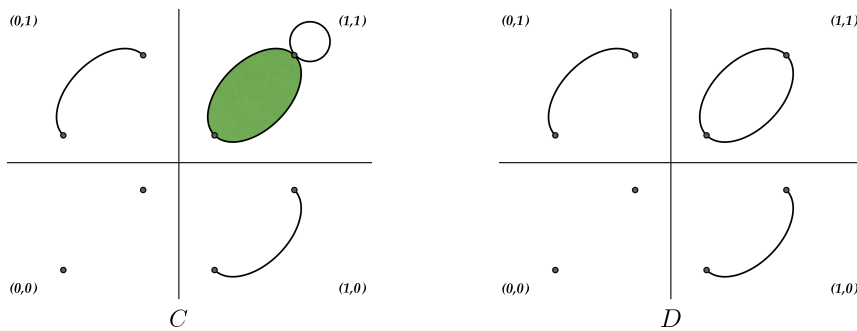
Let  $C, D$  be two bifiltered chain complexes. We call  $C$  and  $D$  *homology-equivalent* if, for any fixed  $k \in \mathbb{N}$ ,  $H_k(C_*^p)$  and  $H_k(D_*^p)$  are isomorphic via a map  $\psi_k^p$  and, for any  $p, q \in \mathbb{R}^2$  with  $p \leq q$ , the diagram

$$\begin{array}{ccc} H_k(C_*^p) & \longrightarrow & H_k(C_*^q) \\ \downarrow \psi_k^p & & \downarrow \psi_k^q \\ H_k(D_*^p) & \longrightarrow & H_k(D_*^q) \end{array}$$

commutes where horizontal maps are induced by inclusion maps. Moreover, we call  $C$  and  $D$  *quasi-isomorphic* if the isomorphisms of the above diagram are induced by a collection of chain maps  $f_*^p : C_*^p \rightarrow D_*^p$  satisfying, for any  $p \leq q$  and any  $k$ , the commutative diagram

$$\begin{array}{ccc} C_k^p & \longrightarrow & C_k^q \\ \downarrow f_k^p & & \downarrow f_k^q \\ D_k^p & \longrightarrow & D_k^q \end{array}$$

in which horizontal maps are the inclusion maps. By definition, quasi-isomorphic bifiltered chain complexes are homology-equivalent. The converse, as depicted in Figure 1, does not hold in general. Two chain maps  $\alpha_*, \beta_* : C_* \rightarrow D_*$  are called *chain-homotopic* if there exists



■ **Figure 1** The two depicted bifiltered chain complexes are homology-equivalent but not quasi-isomorphic.

a collection of maps  $\phi_k : C_k \rightarrow D_{k+1}$  such that, for any  $k$ ,  $\partial_{k+1}^D \phi_k + \phi_{k-1} \partial_k^C = \alpha_k - \beta_k$ . Two bifiltered chain complexes  $C$  and  $D$  are called *homotopy-equivalent* if, there exist two collections of chain maps  $f_*^p : C_*^p \rightarrow D_*^p$  and  $g_*^p : D_*^p \rightarrow C_*^p$  such that, for any  $p \in \mathbb{R}^2$ ,



$f_*^p$  and  $g_*^p$  are homotopy-inverse one with respect to the other (i.e.,  $g_*^p f_*^p$  and  $f_*^p g_*^p$  are chain-homotopic to  $\text{id}_{C_*^p}$  and  $\text{id}_{D_*^p}$ , respectively) and they satisfy, for any  $p \leq q$ , the following commutative diagram

$$\begin{array}{ccc}
 C_k^p & \longrightarrow & C_k^q \\
 f_k^p \downarrow & \uparrow g_k^p & f_k^q \downarrow \uparrow g_k^q \\
 D_k^p & \longrightarrow & D_k^q
 \end{array}$$

in which horizontal maps are the inclusion maps. Homotopy-equivalent bifiltered chain complexes are necessarily quasi-isomorphic.

**Representation of bifiltered chain complexes.** We assume that the bifiltered chain complex in input is provided as a finite sequence of triples of the form

$$(p, k, \Delta)$$

where  $p \in \mathbb{R}^2$ ,  $k$  is a positive integer, and  $\Delta$  is a finite list of pairs  $(l, \lambda)$  with  $\lambda \in \mathbb{F}$  and  $l \in \mathbb{N}$ . In order to retrieve the bifiltered chain complex  $C$  represented by this list, each triple  $(p, k, \Delta)$  has to be interpreted as a  $k$ -generator  $c$  with value  $v(c) = p$  and whose boundary is encoded through  $\Delta$ . Specifically, by defining as  $c_l$  the generator stored in position  $l$  of the sequence of triples, the pair  $(l, \lambda)$  in  $\Delta$  represents that  $c_l$  appears in the boundary of  $c$  with coefficient  $\lambda$ . In order to ensure that the above-described interpretation of the input sequence actually returns a valid bifiltered chain complex, we require that, if  $c'$  is a generator appearing in the boundary of  $c$ , then the value  $v$  associated to  $c'$  is lower than or equal to the one taken by  $c$ .

Given a chain complex  $C_*$  endowed with a value function  $v$ , let us consider an injective function  $i$  providing the set of generators of  $C_*$  with a total order which is consistent with the partial order induced by the value function  $v$ . More precisely, given two generators  $c_1, c_2$  of  $C_*$ , the *index* function  $i$  has to satisfy that  $v(c_1) \leq v(c_2)$  implies  $i(c_1) \leq i(c_2)$ .

We uniquely represent the bifiltered chain complex  $C$  as follows. Each generator  $c$  of  $C_*$  is stored in a data structure that we call the *column* of the generator. A column stores:

- the index  $i(c)$ ,
- the value  $v(c)$ ,
- the dimension  $k$  of  $g$ ,
- the boundary of  $c$ .

The boundary is stored as a (possibly empty) container of entries of the form  $(i(c'), \lambda)$ , where  $c'$  is a generator of dimension  $k - 1$  such that  $\lambda := \langle \partial_k(c), c' \rangle \neq 0$ . In such a case, we say that  $c'$  is in the boundary of  $c$ . The boundary of  $c$  is then the linear combination induced by this container. The name “column” comes from the idea that the collection of all columns can be visualized as a matrix. If we decorate the columns of the matrix with extra information (index, value, and dimension), we obtain the data structure from above. We will also write  $k$ -column if the column represents a generator of dimension  $k$ . Due to the fact that a column is nothing but an encoding of a generator, in the following, with a small abuse of notation we will often use interchangeably the two terms. Given a  $k$ -column  $c$ , we define the *local pivot* of  $c$  as the generator in the boundary of  $c$  with maximal index such that  $v(c') = v(c)$ . If no such generator exists, we simply say that  $c$  has no local pivot.

Arguably, the most common case of input data is simplicial complexes. Let  $K$  denote a finite simplicial complex. A *bifiltration* is a collection  $(K^p)_{p \in \mathbb{R}^2}$  of subcomplexes of  $K$  such that  $K^p \subseteq K^q$  whenever  $p \leq q$ . Fixing a simplex  $\sigma \in K$ , we say that  $p \in \mathbb{R}^2$  is *critical* for  $\sigma$ ,



if  $\sigma \in K^p$ , but  $\sigma \notin K^{p-(\epsilon,0)} \cup K^{p-(0,\epsilon)}$  for every  $\epsilon > 0$ . The bifiltration is called *h-critical* with  $h \geq 1$ , if for every  $\sigma \in K$ , there are at most  $h$  critical positions in  $\mathbb{R}^2$ . For instance, 1-critical means that every simplex in  $K$  enters the filtration at a unique minimal value. Such 1-critical bifiltrations can easily be described by a sequence of the form  $(p, k, \Delta)$  as above by adding one line per simplex which defines its critical position, its dimensions, and its boundary. The case of  $h$ -critical bifiltrations can be handled with the following trick (see also [7]). Letting  $p_1, \dots, p_h$  denote the critical positions of  $\sigma$ , sorted by  $x$ -coordinate, we introduce  $h$  distinct  $k$ -generators  $c_1, \dots, c_h$  of the form  $(p_i, k, \Delta)$ , that is,  $h$  copies of the same simplex. For two consecutive positions  $p_i = (x_i, y_i)$  and  $p_{i+1} = (x_{i+1}, y_{i+1})$ , we add an additional  $(k+1)$ -generator at  $(x_{i+1}, y_i)$  (which is the smallest point  $q$  such that  $p_i \leq q$  and  $p_{i+1} \leq q$ ), whose boundary is equal to  $c_i - c_{i+1}$ .

### 3 Algorithm

Given a bifiltered chain complex  $C$  encoded as a collection of columns, we define the *chunk algorithm* that returns as output a bifiltered chain complex as a collection of columns that is homotopy-equivalent to  $C$  and has fewer generators. The algorithm works in three phases:

- *local reduction*;
- *compression*;
- *removal of local columns*.

**Phase I: Local reduction.** The goal of this phase is to label columns as global, local positive or local negative columns. Initially, all columns are unlabeled. We proceed in decreasing dimensions, from the top-dimensions of  $C$  down to 0. For dimension  $k$ , the algorithm traverses the  $k$ -columns in increasing order with respect to  $i$  and performs the following operations on a  $k$ -column  $c$ . If  $c$  is already labeled, do nothing. Otherwise, as long as  $c$  has a local pivot and there is a  $k$ -column  $c'$  with  $i(c') < i(c)$  and the same local pivot as  $c$ , perform the column addition  $c \leftarrow c + \lambda c'$ , where  $\lambda$  is chosen such that the local pivot of  $c$  disappears. If at the end of this loop, the column  $c$  does not have a local pivot, we label the column as global and proceed. Otherwise, we label  $c$  as local negative and its local pivot  $c'$  as local positive. We call  $(c', c)$  a *local pair* in this case. This ends the description of Phase I of the algorithm.

Note that within the local reduction, any column addition of the form  $c \leftarrow c + \lambda c'$  implies that  $v(c) = v(c')$ . Hence, the local reduction operates independently on columns of the same value. We call blocks of columns with the same value *chunks*; hence the name of the algorithm. Operations on one chunk do not affect columns on any other chunk, hence the local reduction phase can be readily invoked in parallel on the chunks of the chain complex.

Finally, note that by proceeding in decreasing dimension, we avoid performing any column additions on local positive columns. That is reminiscent of the *clearing optimization* in the one-parameter version [3, 8].

**Phase II: Compression.** In the second phase, the algorithm removes local (positive or negative) generators from the boundary of all global columns in the matrix:

For each global  $k$ -column  $c$ , while the boundary of the column contains a generator that is local positive or local negative, the algorithm picks the local  $(k-1)$ -generator  $c'$  with maximal index.

- if  $c'$  is negative, remove  $c'$  from the boundary of  $c$ ;
- if  $c'$  is positive, denote  $c''$  as the (unique) local negative  $k$ -column with  $c'$  as local pivot and perform the column addition  $c \leftarrow c + \lambda c''$ , where  $\lambda$  is chosen such that  $c'$  disappears in the boundary of  $c$ .

This ends the description of the compression phase. On termination, all columns in the boundary of a global  $k$ -column are global  $(k - 1)$ -columns.

The above process terminates for a column  $c$  because the index of the maximal local generator in the boundary of  $c$  is strictly decreasing in each step. That is clear for the case that  $c'$  is local negative. If  $c'$  is local positive, then  $c'$  is the generator in the boundary of  $c''$  with the maximal index, so the column addition does not introduce in the boundary of  $c$  any generators with a larger index.

Note that the compression of a global column does not affect the result on any other global column. Thus, the phase can be parallelized as well.

**Phase III: Removal of local pairs.** In this step, the chain complex becomes smaller. The procedure is simple: traverse all columns, and remove all columns labeled as local (positive or negative). Return the remaining (global) columns as resulting chain complex. This finishes the description of the phase and the entire chunk algorithm.

## 4 Correctness

In this section, we prove that the presented algorithm returns a bifiltered chain complex that is homotopy-equivalent to the input. For that, we define two elementary operations on chain complexes:

**Order-preserving column addition.** An operation of the form  $c \leftarrow c + \lambda c'$  is called *order-preserving* if  $v(c') \leq v(c)$ . Note that such an operation maintains the property that any generator  $c''$  in the boundary of  $c$  satisfies  $v(c'') \leq v(c)$ , by transitivity of  $v$ . We remark that order-preserving column additions are the generalization of left-to-right column additions in the one-parameter case.

**Removal of a local pair.** Fix a local pair  $(c_1, c_2)$ , that means,  $c_1$  is a  $k$ -column,  $c_2$  is a  $(k + 1)$ -column,  $v(c_1) = v(c_2)$  and  $c_1$  is the local pivot of  $c_2$ . We call *removal of the local pair*  $(c_1, c_2)$  the operation  $Del(c_1, c_2)$  which acts on the columns as follows.

- For every  $(k + 1)$ -column  $c$ , replace its boundary  $\partial_{k+1}(c)$  with  $\partial_{k+1}(c) - \lambda^{-1}\mu\partial_{k+1}(c_2)$ , where  $\lambda$  and  $\mu$  are the coefficients of  $c_1$  in  $\partial_{k+1}(c_2)$  and in  $\partial_{k+1}(c)$ , respectively. In particular, after this operation,  $c_1$  disappeared from the boundary of any  $(k + 1)$ -columns.
- For every  $(k + 2)$ -column  $c$ , update its boundary by setting the coefficient of  $c_2$  in  $\partial_{k+2}(c)$  to 0. Visualizing the chain complex as a matrix, this corresponds to removing the row corresponding to  $c_2$ .
- Delete the columns  $c_1$  and  $c_2$ .

Note that all column additions performed in the first step are order-preserving because the pair  $(c_1, c_2)$  is local, that is,  $v(c_1) = v(c_2)$ .

We will show at the end of this section that both elementary operations leave the homotopy type the bifiltered chain complex unchanged.

► **Theorem 1.** *Let  $\bar{C}$  denote the bifiltered chain complex computed by the chunk algorithm from the previous section on an input bifiltered chain complex  $C$ . Then,  $\bar{C}$  and  $C$  are homotopy-equivalent.*

**Proof.** The idea is to express the chunk algorithm by a sequence of order-preserving column additions and removals of local pairs. Because every column addition in Phase I is between columns of the same value, all column additions are order-preserving. Hence, after Phase I, the chain complex is equivalent to the input.

In Phase II, note that all column additions performed are order-preserving. Indeed, if  $c'$  is in the boundary of column  $c$ , then  $v(c') \leq v(c)$  holds. If  $c'$  is local positive, it triggers a column addition of the form  $c \leftarrow c + \lambda c''$  with its local negative counterpart  $c''$ . Since  $v(c') = v(c'')$ ,  $v(c'') \leq v(c)$  as well.

A further manipulation in Phase II is the removal of local negative columns from the boundary of global columns. These removals cannot be directly expressed in terms of the two elementary operations from above. Instead, we define a slight variation of our algorithm: in Phase II, when we encounter a local negative  $c'$ , we do nothing. In other words, the compression only removes the local positive generators from the boundary  $c$ , and keeps local negative and global generators. In Phase III, instead of removing local columns, we perform a removal of a local pair  $(c_1, c_2)$  whenever we encounter a local negative column  $c_2$  with local pivot  $c_1$ . We call that algorithm the *modified chunk algorithm*. Note that this modified algorithm is a sequence of order-preserving column additions, followed by a sequence of local pair removals, and thus produces a chain complex that is equivalent to the input  $C$ .

We argue next that the chunk algorithm and the modified chunk algorithm yield the same output. Since both versions eventually remove all local columns, it suffices to show that they yield the same global columns. Fix an index of a global column, and let  $c$  denote the column of that index returned by the original chunk algorithm. Let  $c^*$  denote the column of the same index produced by the modified algorithm after the modified Phase II. The difference of  $c^*$  and  $c$  lies in the presence of local negative generators in the boundary of  $c^*$  which have been removed in  $c$ . The modified Phase III affects  $c^*$  in the following way: when a local pair  $(c_1, c_2)$  is removed, the local negative  $c_2$  is, if it is present, removed from the boundary of  $c^*$ . There is no column addition during the modified Phase III involving  $c^*$  because all local positive columns have been eliminated. Hence, the effect of the modified Phase III on  $c^*$  is that all local negative columns are removed from its boundary which turns  $c^*$  to be equal to  $c$  at the end of the algorithm. Hence, the output of both algorithms is the same, proving the theorem. ◀

We proceed with the proofs that both elementary operations yield homotopy-equivalent bifiltered chain complexes.

**Order-preserving column addition.** Given two  $k$ -columns  $c_1, c_2$  such that  $v(c_1) \leq v(c_2)$  and a scalar value  $\lambda \in \mathbb{F}$ , the algorithm we propose allows for adding  $\lambda$  copies of the boundary of column  $c_1$  to the boundary of column  $c_2$ . In this section, we formalize that in terms of modifications of chain complexes and we prove that this operation does not affect multi-parameter persistent homology.

Given a chain complex  $C_* = (C_l, \partial_l)$  endowed with a value function  $v : C_* \rightarrow \mathbb{R}^2$ , let us consider two generators  $c_1, c_2$  among the ones of the space of the  $k$ -chains  $C_k$  such that  $\langle c_1, c_2 \rangle = 0$  and  $v(c_1) \leq v(c_2)$ . Chosen a scalar value  $\lambda \in \mathbb{F}$ , let us define  $\bar{C}_* = (\bar{C}_l, \bar{\partial}_l)$  by setting:

- $\bar{C}_l = C_l$ ,
- for any  $c \in \bar{C}_l$ ,

$$\bar{\partial}_l(c) = \begin{cases} \partial_k(c) + \lambda \langle c, c_2 \rangle \partial_k(c_1) & \text{if } l = k, \\ \partial_{k+1}(c) - \lambda \langle \partial_{k+1}(c), c_2 \rangle c_1 & \text{if } l = k + 1, \\ \partial_l(c) & \text{otherwise.} \end{cases}$$

As formally proven in the full version of the paper,  $\bar{C}_*$  is a chain complex.

Let us define the maps  $f_* : C_* \rightarrow \bar{C}_*$ ,  $g_* : \bar{C}_* \rightarrow C_*$  as follows:

- for any  $c \in C_l$ ,

$$f_l(c) = \begin{cases} c - \lambda \langle c, c_2 \rangle c_1 & \text{if } l = k, \\ c & \text{otherwise;} \end{cases}$$

- for any  $c \in \bar{C}_l$ ,

$$g_l(c) = \begin{cases} c + \lambda \langle c, c_2 \rangle c_1 & \text{if } l = k, \\ c & \text{otherwise.} \end{cases}$$

The just defined maps enable to prove the following result (see the full version of the paper for detailed proofs).

► **Proposition 2.**  $C_*$  and  $\bar{C}_*$  are isomorphic via the chain map  $f_*$  and its inverse  $g_*$ .

Since the function  $v$  is a valid value function for  $\bar{C}_*$  inducing a bifiltered chain complex  $\bar{C}$  and, for any  $p \in \mathbb{R}^2$ , the restrictions  $f_l^p : C_l^p \rightarrow \bar{C}_l^p$  and  $g_l^p : \bar{C}_l^p \rightarrow C_l^p$  of the maps  $f_l$  and  $g_l$ , respectively, are well-defined (see the full version of the paper), we are ready to prove the equivalence between the two bifiltered chain complexes  $C$  and  $\bar{C}$ .

► **Theorem 3.**  $C$  and  $\bar{C}$  are homotopy-equivalent.

**Proof.** The previous results enables us to claim that, for any  $p, q \in \mathbb{R}^2$  with  $p \leq q$ , the following diagram (in which horizontal maps are the inclusion maps)

$$\begin{array}{ccc} C_l^p & \longrightarrow & C_l^q \\ \downarrow f_l^p & & \downarrow f_l^q \\ \bar{C}_l^p & \longrightarrow & \bar{C}_l^q \end{array}$$

commutes and the maps  $f_l^p, f_l^q$  are isomorphisms. ◀

**Removal of a local pair.** Given a local pair  $(c_1, c_2)$  of columns, the algorithm we propose allows for deleting them from the boundary matrix. In this section, we formalize that in terms of modifications of chain complexes and we prove that this operation does not affect multi-parameter persistent homology.

Given a chain complex  $C_* = (C_l, \partial_l)$  endowed with a value function  $v : C_* \rightarrow \mathbb{R}^2$ , let us consider two generators  $c_1, c_2$  among the ones of the space of the  $k$ -chains  $C_k$  and of the space of the  $(k + 1)$ -chains  $C_{k+1}$ , respectively, such that  $\lambda := \langle \partial_{k+1}(c_2), c_1 \rangle \neq 0$  and  $v(c_1) = v(c_2)$ . Let us define  $\bar{C}_* = (\bar{C}_l, \bar{\partial}_l)$  by setting:

- the space of the  $l$ -chains  $\bar{C}_l$  as

$$\bar{C}_l = \begin{cases} \{c \in C_k \mid \langle c, c_1 \rangle = 0\} & \text{if } l = k, \\ \{c \in C_{k+1} \mid \langle c, c_2 \rangle = 0\} & \text{if } l = k + 1, \\ C_l & \text{otherwise;} \end{cases}$$

- for any  $c \in \bar{C}_l$ ,

$$\bar{\partial}_l(c) = \begin{cases} \partial_{k+1}(c) - \lambda^{-1} \langle \partial_{k+1}(c), c_1 \rangle \partial_{k+1}(c_2) & \text{if } l = k + 1, \\ \partial_{k+2}(c) - \langle \partial_{k+2}(c), c_2 \rangle c_2 & \text{if } l = k + 2, \\ \partial_l(c) & \text{otherwise.} \end{cases}$$

## 37:10 Chunk Reduction for Multi-Parameter Persistent Homology

As formally proven in the full version of the paper,  $\bar{C}_*$  is a chain complex.

Let us define the maps  $r_* : C_* \rightarrow \bar{C}_*$ ,  $s_* : \bar{C}_* \rightarrow C_*$  as follows:

■ for any  $c \in C_l$ ,

$$r_l(c) = \begin{cases} c - \lambda^{-1} \langle c, c_1 \rangle \partial_{k+1}(c_2) & \text{if } l = k, \\ c - \langle c, c_2 \rangle c_2 & \text{if } l = k + 1, \\ c & \text{otherwise;} \end{cases}$$

■ for any  $c \in \bar{C}_l$ ,

$$s_l(c) = \begin{cases} c - \lambda^{-1} \langle \partial_{k+1}(c), c_1 \rangle c_2 & \text{if } l = k + 1, \\ c & \text{otherwise;} \end{cases}$$

The just defined maps enable to prove the following result (see the full version of the paper for detailed proofs).

► **Proposition 4.** *The maps  $r_*$  and  $s_*$  are chain maps which are homotopy-inverse one with respect to the other.*

This latter combined with the fact that the function  $v$  is a valid value function for  $\bar{C}_*$  inducing a bifiltered chain complex  $\bar{C}$  and, for any  $p \in \mathbb{R}^2$ , the restrictions  $r_l^p : C_l^p \rightarrow \bar{C}_l^p$ ,  $s_l^p : \bar{C}_l^p \rightarrow C_l^p$  of the maps  $r_l$  and  $s_l$ , as well as the restrictions of the maps ensuring that  $r_*$  and  $s_*$  are homotopy-inverse, are well-defined (see the full version of the paper), guarantees the homotopy-equivalence between the two bifiltered chain complexes  $C$  and  $\bar{C}$ .

► **Theorem 5.**  *$C$  and  $\bar{C}$  are homotopy-equivalent.*

### 5 Optimality and complexity

**Optimality.** Let  $D$  be a bifiltered chain complex. For  $p \in \mathbb{R}^2$ , we define

$$D_*^{<p} := \sum_{q < p} D_*^q,$$

where the sum of chain complexes, analogously to the sum of the vector spaces, is the chain complex spanned by the union of the generators of the summands. Moreover, let  $\eta_k^p$  be the homology map in dimension  $k$  induced by the inclusion of  $D_*^{<p}$  into  $D_*^p$ . We denote the number of variations in the  $k^{\text{th}}$  homology space occurred at value  $p$  as

$$\delta_k^p(D) := \dim \ker \eta_{k-1}^p + \dim \text{coker } \eta_k^p,$$

and the number of  $k$ -generators added at value  $p$  in  $D$  as

$$\gamma_k^p(D) := \dim D_k^p - \dim D_k^{<p}.$$

► **Theorem 6.** *Let  $\bar{C}$  be the bifiltered chain complex obtained by applying the chunk algorithm to the bifiltered chain complex  $C$ . We have that  $\delta_k^p(C) = \gamma_k^p(\bar{C})$ .*

The detailed proof is given in the full version of the paper and can be summarized as follows. Every global column with value  $p$  either destroys a homology class of  $H(C^{<p})$ , or it creates a new homology class in  $H(C^p)$ , which is not destroyed by any other column of value  $p$ . Hence, each global column contributes a generator to  $\ker \iota_{k-1}^p$  or to  $\text{coker } \iota_k^p$ , where  $\iota_l^p$  is

the map between the  $l^{th}$  homology spaces induced by the inclusion of  $C_*^{<p}$  into  $C_*^p$ . Local columns do not contribute to either of these two spaces. The result follows from the fact that the number of global columns at value  $p$  is precisely the number of generators added at  $\bar{C}$ .

The next statement shows that our construction is optimal in the sense that any bifiltered chain complex  $D$  that is quasi-isomorphic to  $C$  must have at least as many generators as  $\bar{C}$ .

► **Theorem 7.** *Any bifiltered chain complex  $D$  quasi-isomorphic to  $C$  has to add at least  $\delta_k^p(C)$   $k$ -generators at value  $p$ . I.e.,  $\delta_k^p(C) \leq \gamma_k^p(D)$ .*

The detailed proof is given in the full version of the paper. To summarize it, it is not too hard to see that, for any bifiltered chain complex  $D$ ,

$$\delta_k^p(D) \leq \gamma_k^p(D)$$

holds. Moreover, the quasi-isomorphism of  $C$  and  $D$  implies that  $\dim \ker \eta_{k-1}^p = \dim \ker \iota_{k-1}^p$  and  $\dim \operatorname{coker} \eta_k^p = \dim \operatorname{coker} \iota_k^p$ . So,

$$\delta_k^p(C) = \delta_k^p(D)$$

which implies that claim. The equality of the dimensions is formally verified using the Mayer-Vietoris sequence and the 5-lemma to establish an isomorphism from  $H_k(C_*^{<p})$  to  $H_k(D_*^{<p})$  that commutes with the isomorphism at value  $p$ .

**Complexity.** In order to properly express the time and the space complexity of the proposed algorithm, let us introduce the following parameters. Given a bifiltered chain complex  $C$ , we denote as  $n$  the number of generators of  $C$ , as  $m$  the number of chunks (i.e., the number of different values assumed by  $v$ ), as  $\ell$  the maximal size of a chunk, and as  $g$  the number of global columns. Moreover, we assume the maximal size of the support of the boundary of the generators of  $C$  as a constant. The latter condition is always ensured for bifiltered simplicial complex of fixed dimension.

► **Theorem 8.** *The chunk algorithm has time complexity  $O(m\ell^3 \log \ell + g\ell n \log \ell)$  and space complexity  $O(n\ell + g^2)$ .*

**Proof.** Due to the similarity between the two algorithms, the analysis of complexity of the proposed algorithm is analogous to the first two steps of the one-parameter chunk algorithm [3]. The additional factor of  $\log \ell$  comes from our choice of using priority queues as column type and could be removed by using list representations as in [3].<sup>2</sup>

On the space complexity, during the Phase I, the generators in the boundary of any column can be at most  $O(\ell)$ . So,  $O(n\ell)$  is a bound on the accumulated size of all columns after Phase I. During Phase II, the boundary of any global column can consist of up to  $n$  generators, but reduces to  $g$  generators at the end of the compression of the column because all local entries have been removed. Hence, the final chain complex has at most  $g$  entries in each of its  $g$  columns, and requires  $O(g^2)$  space. Hence, the total space complexity is  $O(n\ell + n + g^2)$ , where the second summand is redundant.<sup>3</sup> ◀

<sup>2</sup> We remark, however, that this would result in a performance penalty in practice. See [4].

<sup>3</sup> We remark that this bound only holds for the sequential version of the algorithm. In a parallelized version, it can happen that several compressed columns achieve a size of  $O(n)$  at the same time.

## 6 Implementation and experimental results

We briefly introduce the developed implementation of the chunk algorithm and we evaluate its performance. The current C++ implementation of the chunk algorithm consists of approximately 350 lines of code. It takes as input a finite bifiltered chain complex expressed accordingly to the representation described in Section 2 and encodes each column as a `std::priority_queue`. Even if the chunk algorithm permits a parallelization in shared memory, this first implementation does not exploit this potential. Our experiments have been performed on a configuration Intel Xeon CPU E5-1650 v3 at 3.50 GHz with 64 GB of RAM.

For testing the implemented chunk algorithm, we have compared its performances with the simplification process based on discrete Morse theory proposed in [17] (DMT-based algorithm) whose implementation is publicly available [10]. We remark once more that the DMT-based approach yields a somewhat richer output than solely a simplified chain complex with the same homotopy; however, we only compare the size of the resulting structure (in terms of the number of generators per chain group) in this experimental comparison.

In our experiments, we have considered both synthetic and real datasets represented as simplicial complexes. The latter ones are courtesy of the AIM@SHAPE data repository [9]. Most of the datasets are of dimension 2 or 3 and are embedded in a 3D environment. For our experiments, we have adopted as filtering functions the ones obtained by extending to all the simplices of the complex the  $x$  and the  $y$  coordinates assigned to the vertices of the datasets. Table 1 displays the achieved results.

The data sets are given in off file format, that means, as a list of triangles specified by boundary vertices. In order to apply our algorithm, we first have to convert the data into a boundary matrix representation. Hence, we enlist in Table 1 the *preparation time* to create the boundary matrix and the *simplification time* to perform our chunk algorithm. In contrast, the DMT-based approach avoids the initial construction of the boundary matrix but transforms the input into a different data structure before starting its simplification step. We also list the running times of these separate steps in Table 1. In both cases, we do not list the time for reading the input file into memory and writing the output structure on disk.

The column *Size* in Table 1 collects the sizes (in terms of the number of simplices/columns) of the bifiltered chain complexes in input and in output. The compression factor achieved by both algorithms varies between 9 and 23. As average, the reduced chain complex in output is approximately 13 times smaller than the original one. Despite of the theoretical advantage that our approach yields an optimal size in all situations, the size of the output returned by the two algorithms is nearly the same in all listed examples. A difference can be noticed just for datasets including shapes that can be considered as “pathological”. For instance, an example of such a dataset is the conification of a dunce hat.

The column *Time* shows the computation times of both algorithms. In all tested examples, the chunk algorithm takes a small fraction of time with respect to the DMT-based approach. We also see that the creation of the boundary matrix (preparation step) takes more time than the chunk algorithm (simplification step) by a factor of 2 to 3, but even these two steps combined are faster than the simplification step of the DMT-based approach. It is worth investigating whether the boundary matrix creation becomes a more severe bottleneck for other datasets (e.g., in higher dimension), where the DMT-based approach might have advantages by not creating the boundary matrix explicitly.

The column *Memory Usage* shows the memory consumption of both approaches. The chunk algorithm does not need to encode auxiliary structures like the discrete Morse gradient stored in the DMT-based approach resulting in a slightly less amount of required space.



■ **Table 1** Results and performances obtained by the chunk and the DMT-based algorithms. The columns from left to right indicate: the name of the dataset (*Dataset*), the number of cells before and after the simplification algorithms (*Size*), the time (*Time*), expressed in seconds, needed to perform the preparation step (*Prep.*) and the simplification step (*Simpl.*), the maximum peak of memory, expressed in GB, required by the two algorithms (*Memory Usage*).

| Dataset        | Size   |        | Time (sec.) |        |       |        | Memory Usage (GB) |      |
|----------------|--------|--------|-------------|--------|-------|--------|-------------------|------|
|                | Input  | Output | Chunk       |        | DMT   |        | Chunk             | DMT  |
|                |        |        | Prep.       | Simpl. | Prep. | Simpl. |                   |      |
| Eros           | 2.9 M  | 202 K  | 1.7         | 0.8    | 2.7   | 15.8   | 0.36              | 0.46 |
| Donna          | 3.0 M  | 217 K  | 1.8         | 0.8    | 2.8   | 16.9   | 0.38              | 0.48 |
| Chinese Dragon | 3.9 M  | 321 K  | 2.5         | 1.1    | 3.9   | 22.3   | 0.52              | 0.64 |
| Circular Box   | 4.2 M  | 365 K  | 2.9         | 1.2    | 4.3   | 24.0   | 0.68              | 0.68 |
| Ramesses       | 5.0 M  | 407 K  | 3.4         | 1.3    | 5.5   | 29.4   | 0.68              | 0.81 |
| Pensatore      | 6.0 M  | 369 K  | 3.8         | 1.6    | 6.8   | 34.3   | 0.76              | 0.97 |
| Raptor         | 6.0 M  | 260 K  | 4.4         | 1.7    | 5.4   | 32.3   | 0.73              | 0.93 |
| Neptune        | 12.0 M | 893 K  | 8.4         | 4.4    | 14.9  | 69.2   | 1.52              | 1.94 |
| Cube 1         | 590 K  | 67 K   | 0.5         | 0.3    | 0.7   | 3.2    | 0.09              | 0.10 |
| Cube 2         | 2.4 M  | 264 K  | 1.8         | 1.1    | 2.6   | 13.1   | 0.35              | 0.40 |
| Cube 3         | 9.4 M  | 1.0 M  | 7.6         | 4.8    | 11.0  | 53.1   | 1.37              | 1.58 |
| Cube 4         | 37.7 M | 4.2 M  | 31.9        | 19.4   | 44.9  | 216.0  | 5.50              | 6.32 |

Depending on the dataset, the chunk approach requires between 0.09 and 5.50 GB of memory and it is, on average, 1.2 times more compact than the DMT-based one.

In summary, the chunk algorithm satisfies a stronger optimality condition than [17], but still is an order of magnitude faster and uses comparable memory.

## 7 Conclusion

We have presented a pre-processing procedure for improving the computation of multi-parameter persistent homology and we have provided theoretical and experimental evidence of its effectiveness. In the future, we want to further improve the proposed strategy as follows. First, we would like to develop a parallel implementation with shared memory of the chunk algorithm and compare its performances with the ones obtained by the current version. Similarly to [17], we also want to evaluate the impact of the chunk algorithm for the computation of the persistence module and of the persistence space.

Our optimality criterion is formulated for the class of chain complexes quasi-isomorphic to the input. It is possible to produce counterexamples showing that this statement cannot be generalized to the class of the chain complexes homology-equivalent to the input. Nevertheless, we want to further investigate the optimality properties satisfied by our algorithm and compare our algorithm with the minimal presentation algorithm for persistence modules from RIVET.

---

## References

- 1 M. Allili, T. Kaczynski, and C. Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 78:61–75, 2017. doi:10.1016/j.jsc.2015.11.020.
- 2 M. Allili, T. Kaczynski, C. Landi, and F. Masoni. Acyclic partial matchings for multidimensional persistence: algorithm and combinatorial interpretation. *Journal of Mathematical Imaging and Vision*, pages 1–19, 2018.



- 3 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: computing persistent homology in chunks. In *Topological methods in data analysis and visualization III*, pages 103–117. Springer, 2014.
- 4 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. Phat - persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78:76–90, 2017.
- 5 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.
- 6 A. Cerri and C. Landi. The persistence space in multidimensional persistent homology. In R. Gonzalez-Diaz, M.-J. Jimenez, and B. Medrano, editors, *Discrete Geometry for Computer Imagery*, pages 180–191. Springer Berlin Heidelberg, 2013.
- 7 W. Chachólski, M. Scolamiero, and F. Vaccarino. Combinatorial presentation of multidimensional persistent homology. *Journal of Pure and Applied Algebra*, 221(5):1055–1075, 2017.
- 8 C. Chen and M. Kerber. Persistent homology computation with a twist. In *Proceedings 27th European Workshop on Computational Geometry*, volume 11, pages 197–200, 2011.
- 9 Digital Shape WorkBench. AIM@SHAPE project, 2006. URL: <http://visionair.ge.imati.cnr.it>.
- 10 F. Iuricich. FG-multi, 2018. URL: [http://github.com/IuricichF/fg\\_multi](http://github.com/IuricichF/fg_multi).
- 11 F. Iuricich, S. Scaramuccia, C. Landi, and L. De Floriani. A discrete Morse-based approach to multivariate data analysis. In *SIGGRAPH ASIA 2016 Symposium on Visualization*, pages 5–12. ACM, 2016.
- 12 K. P. Knudson. A refinement of multi-dimensional persistence. *Homology, Homotopy and Applications*, 10(1):259 – 281, 2008.
- 13 M. Lesnick and M. Wright. Computing minimal presentations of bipersistence modules in cubic time. In preparation.
- 14 M. Lesnick and M. Wright. Interactive visualization of 2-D persistence modules. *arXiv preprint*, 2015. [arXiv:1512.00180](https://arxiv.org/abs/1512.00180).
- 15 E. Miller. Data structures for real multiparameter persistence modules. *arXiv preprint*, 2017. [arXiv:1709.08155](https://arxiv.org/abs/1709.08155).
- 16 S. Scaramuccia. *Computational and theoretical issues of multiparameter persistent homology for data analysis*. PhD thesis, University of Genova, Italy, 2018. URL: <http://hdl.handle.net/11567/929143>.
- 17 S. Scaramuccia, F. Iuricich, L. De Floriani, and C. Landi. Computing multiparameter persistent homology through a discrete Morse-based approach. *arXiv preprint*, 2018. [arXiv:1811.05396](https://arxiv.org/abs/1811.05396).

# The Crossing Tverberg Theorem

Radoslav Fulek 

IST Austria, Klosterneuburg, Austria  
radoslav.fulek@gmail.com

Bernd Gärtner

Department of Computer Science, ETH Zürich, Switzerland  
<https://people.inf.ethz.ch/gaertner/>  
gaertner@inf.ethz.ch

Andrey Kupavskii

University of Oxford, UK; Moscow Institute of Physics and Technology, Russia  
<http://kupavskii.com>  
kupavskii@ya.ru

Pavel Valtr 

Department of Applied Mathematics, Faculty of Mathematics and Physics,  
Charles University, Prague, Czech Republic  
Department of Computer Science, ETH Zürich, Switzerland  
<https://kam.mff.cuni.cz/~valtr/>

Uli Wagner 

IST Austria, Klosterneuburg, Austria  
uli@ist.ac.at

---

## Abstract

---

The Tverberg theorem is one of the cornerstones of discrete geometry. It states that, given a set  $X$  of at least  $(d + 1)(r - 1) + 1$  points in  $\mathbb{R}^d$ , one can find a partition  $X = X_1 \cup \dots \cup X_r$  of  $X$ , such that the convex hulls of the  $X_i$ ,  $i = 1, \dots, r$ , all share a common point. In this paper, we prove a strengthening of this theorem that guarantees a partition which, in addition to the above, has the property that the boundaries of full-dimensional convex hulls have pairwise nonempty intersections. Possible generalizations and algorithmic aspects are also discussed.

As a concrete application, we show that any  $n$  points in the plane in general position span  $\lfloor n/3 \rfloor$  vertex-disjoint triangles that are pairwise crossing, meaning that their boundaries have pairwise nonempty intersections; this number is clearly best possible. A previous result of Alvarez-Rebollar et al. guarantees  $\lfloor n/6 \rfloor$  pairwise crossing triangles. Our result generalizes to a result about simplices in  $\mathbb{R}^d$ ,  $d \geq 2$ .

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Combinatoric problems; Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Discrete geometry, Tverberg theorem, Crossing Tverberg theorem

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.38

**Funding** *Radoslav Fulek*: The author gratefully acknowledges support from Austrian Science Fund (FWF), Project M2281-N35.

*Andrey Kupavskii*: The research was supported by the Advanced Postdoc.Mobility grant no. P300P2\_177839 of the Swiss National Science Foundation.

*Pavel Valtr*: Research by P. Valtr was supported by the grant no. 18-19158S of the Czech Science Foundation (GAČR).

**Acknowledgements** Part of the research leading to this paper was done during the 16th Gremo Workshop on Open Problems (GWOP), Waltensburg, Switzerland, June 12-16, 2018. We thank Patrick Schnider for suggesting the problem, and Stefan Felsner, Malte Milatz and Emo Welzl for fruitful discussions during the workshop. We also thank Stefan Felsner and Manfred Scheucher for finding and communicating the example from Section 3.2. We thank Dömötör Pálvölgyi and the SoCG reviewers for various helpful comments.



© Radoslav Fulek, Bernd Gärtner, Andrey Kupavskii, Pavel Valtr, and Uli Wagner;  
licensed under Creative Commons License CC-BY

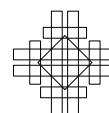
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 38; pp. 38:1–38:13

Leibniz International Proceedings in Informatics



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

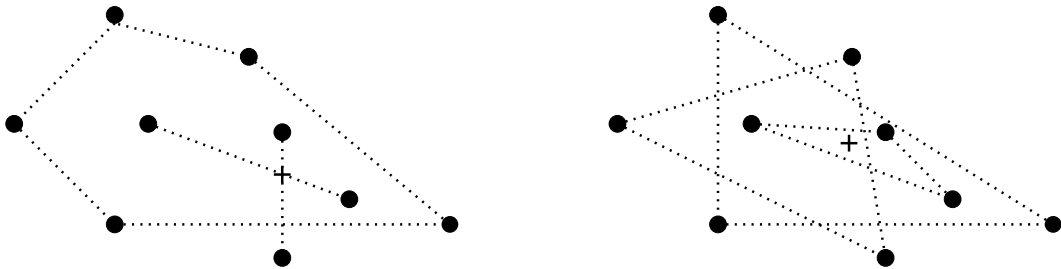


## 1 Introduction and Results

The following theorem was published by Johann Radon in 1921 [21]: any set of  $d + 2$  points in  $\mathbb{R}^d$  can be partitioned into two (disjoint) subsets, whose convex hulls intersect. In 1966, Helge Tverberg [26] proved the following important generalization of Radon's result.

► **Theorem 1** (Tverberg [26]). *Let  $X$  be a set of at least  $(d + 1)(r - 1) + 1$  points in  $d$ -space. Then  $X$  can be partitioned into  $r$  sets whose convex hulls all have a point  $o$  in common. (In the literature, the point  $o$  is referred to as a Tverberg point and the partition as a Tverberg partition.)*

Radon's theorem covers the case  $r = 2$ . Figure 1 illustrates the Tverberg theorem for  $d = 2$  and  $r = 3$ .



■ **Figure 1** The Tverberg theorem for  $d = 2$  and  $r = 3$ : any set of at least 7 points can be partitioned into three sets whose convex hulls all have a point in common. Our example uses 9 points and shows two Tverberg partitions as well as corresponding Tverberg points. Many other Tverberg partitions exist in this example.

This theorem largely influenced the course of discrete geometry and spurred a lot of research in the area. We do not go into more details in this paper and refer the reader to a recent survey by Bárány and Soberón [6].

Throughout this paper, a set of points in  $\mathbb{R}^d$  is said to be in *general position* if no  $d + 1$  points are on a common hyperplane (for example, no three points on a line in the plane).

There is a major open question from the field of geometric graphs, motivating our work. In [2], Aronov et al. conjectured that there exists an absolute constant  $c > 0$ , such that, given any set of  $n$  points in general position in the plane, one can find at least  $cn$  disjoint pairs among them such that their connecting segments cross pairwise. Such a collection of segments is called a *crossing family*. Despite considerable interest in this problem, the best published bound still comes from the original paper [2], stating that one can always find a crossing family of size at least  $c\sqrt{n}$  for some absolute  $c > 0$ . Only recently, after about 25 years, J. Pach, N. Rubin, and G. Tardos improved this lower bound to a nearly linear one [19].

In another attempt to approach this problem, Alvarez-Rebollar et al. asked whether one can find at least  $cn$  disjoint *triples* whose connecting *triangles* cross pairwise [1]. Throughout this paper, we say that two convex polytopes in  $\mathbb{R}^d$  *cross* if their boundaries have a non-empty intersection. We remark that if a convex polytope in  $\mathbb{R}^d$  is not full-dimensional then it has no interior and therefore coincides with its boundary. Our main results below would be false, in the absence of general position, if relative boundaries were considered in the above definition of crossing (an easy counterexample in this situation is a set of points lying on a line in  $\mathbb{R}^2$ ).

We also point out that our definition of “crossing” is different from a classical one due to Fejes-Tóth for general convex sets [25]. According to the latter, two convex sets  $C, C'$  cross if

neither  $C \setminus C'$  nor  $C' \setminus C$  are connected. Under general position, our notion of crossing convex polytopes generalizes the notion of a crossing of two vertex-disjoint segments in a crossing family (the two segments intersecting in their relative interiors) [2]. It also generalizes the notion of Alvarez-Rebollar et al. for vertex-disjoint triangles (an edge of one triangle crossing an edge of the other) [1].

Alvarez-Rebollar et al. showed the following: For every finite point set of size  $n$  in the plane in general position, there exist  $\lfloor \frac{n}{6} \rfloor$  vertex-disjoint and pairwise crossing triangles with vertices in  $P$ . As at most  $\lfloor \frac{n}{3} \rfloor$  disjoint triangles can be found, this leaves a factor-2 gap.

If we only want triangles that have a common point, this gap can be closed using a simple strengthening of the Tverberg theorem for point sets of size at most  $(d+1)r$  that we present next. However, these triangles might not be pairwise crossing, since triangles can be nested; see Figure 1 (right).

► **Theorem 2.** *Let  $X$  be a set of at least  $(d+1)(r-1)+1$  and at most  $(d+1)r$  points in  $d$ -space. Then  $X$  can be partitioned into  $r$  disjoint sets  $X_1, \dots, X_r$  of size at most  $d+1$ , whose convex hulls all have a point in common.*

To prove this, we apply Theorem 1 to  $X$ . We get sets  $X'_1, \dots, X'_r$  whose convex hulls contain a common point, say, the origin. Using Carathéodory's theorem, from every  $X'_i$  of size larger than  $d+1$  we can select  $d+1$  points  $X_i \subseteq X'_i$ , whose convex hull still contains the origin. Finally, some of the sets  $X'_i$  of size smaller than  $d+1$  are filled up to size  $d+1$  with the points removed from other  $X'_i$ 's. The origin is still a common point for all  $\text{conv}(X_i)$ 's.

The main result of this paper provides a crossing version of the Tverberg Theorem 2.

► **Theorem 3.** *Let  $X$  be a set of at least  $(d+1)(r-1)+1$  and at most  $(d+1)r$  points in  $d$ -space. Then  $X$  can be partitioned into  $r$  disjoint sets  $X_1, \dots, X_r$  of size at most  $d+1$ , whose convex hulls all have a point in common. Moreover, for any  $X_i, X_j$  of size  $d+1$ ,  $\text{conv}(X_i)$  and  $\text{conv}(X_j)$  cross, meaning that their boundaries have a non-empty intersection.*

We call such a partition a *crossing Tverberg partition*. An easy calculation shows that the number of sets with exactly  $d+1$  elements is at least  $|X| - dr \in \{r-d, r-d+1, \dots, r\}$ . In particular, for sets  $X$  of size exactly  $(d+1)r$ , we immediately deduce the following simpler-looking corollary.

► **Corollary 4.** *Let  $X$  be a set of  $(d+1)r$  points in  $d$ -space. Then  $X$  can be partitioned into  $r$  sets  $X_1, \dots, X_r$  of size  $d+1$ , whose convex hulls all have a point in common and such that for any  $i, j \in [r]$ ,  $\text{conv}(X_i)$  and  $\text{conv}(X_j)$  cross, meaning that their boundaries have a non-empty intersection.*

We also obtain an optimal strengthening of the result by Alvarez-Rebollar et al. [1] that moreover generalizes to all dimensions  $d \geq 2$ .

► **Corollary 5.** *For every finite point set  $X$  of size  $n$  in the plane in general position, there exist  $\lfloor n/3 \rfloor$  vertex-disjoint and pairwise crossing triangles with vertices in  $X$ .*

*More generally, for every finite point set  $X$  of size  $n$  in  $\mathbb{R}^d$  in general position, there exist  $\lfloor n/(d+1) \rfloor$  vertex-disjoint and pairwise crossing simplices with vertices in  $X$ .*

To derive this from Theorem 3, we remove  $n \bmod (d+1)$  points from  $X$  and then apply Theorem 3 on the remaining set of  $(d+1) \lfloor \frac{n}{d+1} \rfloor$  points.

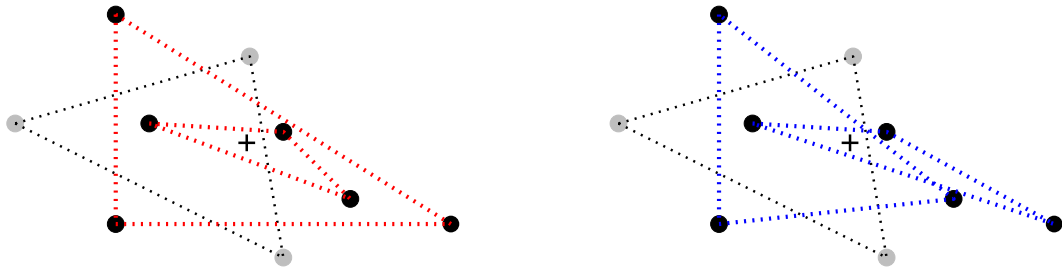
Finally, we get a crossing version of the actual Tverberg Theorem 1, for point sets of arbitrarily large size.

► **Theorem 6** (Crossing Tverberg theorem). *Let  $X$  be a set of at least  $(d + 1)(r - 1) + 1$  points in  $d$ -space. Then  $X$  can be partitioned into  $r$  sets whose convex hulls all have a point in common. Moreover, for any  $X_i, X_j$  of size at least  $d + 1$ ,  $\text{conv}(X_i)$  and  $\text{conv}(X_j)$  cross, meaning that their boundaries have a non-empty intersection.*

This is also easy to prove, using Theorem 3 and Corollary 4. If  $n := |X| \leq (d + 1)r$ , we apply Theorem 3. Otherwise, we apply Corollary 4 to an arbitrary subset  $Y \subset X$  of size  $(d + 1)r$ , resulting in a crossing Tverberg partition into  $r$  sets  $Y_1, \dots, Y_r$  of size  $d + 1$  each. Now we consecutively add the remaining points to suitable sets in such a way that the crossings between the convex hulls are maintained (the Tverberg point automatically remains valid). Suppose that some points have already been added, resulting in sets  $X'_1, \dots, X'_r$  whose convex hulls still cross pairwise. If the next point  $p$  is contained in one of these convex hulls, we simply add it to the corresponding set. Otherwise, we select an inclusion-minimal set  $\text{conv}(X'_k \cup \{p\})$  among the (different) sets  $\text{conv}(X'_i \cup \{p\}), i = 1, \dots, r$ , and add  $p$  to  $X'_k$ . We claim that this cannot result in nested convex hulls. For this, we only have to rule out that  $\text{conv}(X'_k \cup \{p\})$  “swallows” some other  $\text{conv}(X'_i)$ . But this cannot happen, as otherwise  $\text{conv}(X'_i \cup \{p\}) \subset \text{conv}(X'_k \cup \{p\})$ , contradicting the choice of  $k$ .

It remains to prove Theorem 3. We start with a Tverberg partition according to Theorem 2, i.e. a partition into sets of size at most  $d + 1$  such that their convex hulls have a point in common. Such a partition might look like in Figure 2 (left). If the full-dimensional convex hulls (which are simplices) cross pairwise, we are done. In general, however, we still have pairs of nested simplices. As long as this is the case, we *fix* one pair of nested simplices at a time until no pair of nested simplices exists anymore, and our desired crossing Tverberg partition is obtained.

By fixing, we mean that we repartition the  $2(d + 1)$  points involved in the two simplices in such a way that the resulting simplices are not nested anymore but still contain the common point. In the example of Figure 2 (left), there is one pair of nested simplices (red edges), and after fixing it (blue edges), we are actually done in this case; see Figure 2 (right).



■ **Figure 2** Fixing a pair of nested simplices spanned by the 6 black points: two nested simplices (red) are transformed into two simplices that cross (blue).

Repartitioning the points is always possible due to the following lemma, which may be of independent interest. This lemma is our main technical contribution, and we prove it in Section 2.

► **Lemma 7.** *Let  $T, T'$  be two disjoint  $(d + 1)$ -element sets in  $\mathbb{R}^d$  such that  $\mathbf{0} \in \text{conv}(T) \cap \text{conv}(T')$ . Then there exist two disjoint  $(d + 1)$ -element sets  $S, S'$  such that  $S \cup S' = T \cup T'$ ,  $\mathbf{0} \in \text{conv}(S) \cap \text{conv}(S')$ , and moreover,  $\text{conv}(S)$  and  $\text{conv}(S')$  cross.*

To conclude the proof of Theorem 3, we still need to show that after finitely many fixing steps, there are no nested pairs anymore, in which case we have a crossing Tverberg partition.

To see this, we observe that in any fixing operation that replaces  $X_i$  and  $X_j$  such that  $\text{conv}(X_i) \subset \text{conv}(X_j)$ , the simplex  $\text{conv}(X_j)$  is volume-wise the unique largest  $d$ -dimensional simplex that can be formed from the  $2(d+1)$  points  $X_i \cup X_j$  involved in the operation. Hence, the two simplices  $\text{conv}(S)$  and  $\text{conv}(S')$  replacing  $\text{conv}(X_i)$  and  $\text{conv}(X_j)$  are volume-wise both strictly smaller than  $\text{conv}(X_j)$ . Therefore, if we order all full-dimensional simplices by decreasing volume, the sequence of these volumes goes down lexicographically in every fixing operation.

Formally, let  $\mathcal{V} = (V_1, V_2 \dots V_s), s \leq r$  be the sequence of volumes in decreasing order before the fix, and  $\mathcal{V}' = (V'_1, V'_2 \dots V'_s)$  the decreasing order after the fix. Moreover, suppose that  $k$  is the largest index at which the volume of  $\text{conv}(X_j)$  appears in  $\mathcal{V}$ . The volume of  $\text{conv}(X_i)$  appears at a position  $\ell > k$ . As the fixing operation removes a volume equal to  $V_k$  and inserts two volumes smaller than  $V_k$ , we have that  $\mathcal{V}$  and  $\mathcal{V}'$  agree in the first  $k - 1$  positions, but  $V'_k < V_k$ . This exactly defines the relation “ $\mathcal{V}' < \mathcal{V}$ ” in decreasing lexicographical order, and as this is a total order, fixing must eventually terminate.

► **Remark.** Instead of volume, we may use the number of points from  $X$  inside  $\text{conv}(X_j)$  as a measure.

Since applications of Lemma 7 allow to keep the Tverberg point and the sizes of the parts in the (Tverberg) partition, we actually have the following strengthening of Theorem 3.

► **Theorem 8.** *Let  $X$  be a set of points in  $d$ -space. Suppose that there is a Tverberg partition of  $X$  into  $r$  parts of sizes  $s_1 \dots, s_r$ , for which  $o$  is a Tverberg point, and  $s_i \leq d + 1, i = 1, \dots, r$ . Then there is also a crossing Tverberg partition of  $X$  into  $r$  parts of sizes  $s_1 \dots, s_r$ , for which a Tverberg point is  $o$ .*

In the next section, we prove Lemma 7. We remark that for each of the other theorems and corollaries in this section, we have already explained how to derive them from the Tverberg theorem, or from our main Theorem 3. In Section 3, we discuss possible generalizations as well as some algorithmic aspects of the problem.

## 2 Proof of Lemma 7

We employ the following parity lemma that we prove in the next two subsections (in a geometric way for  $d = 2$ , and in a combinatorial way for all dimensions).

► **Lemma 9 (Parity lemma).** *Let  $V \subset \mathbb{R}^d, |V| = 2(d + 1)$ , such that  $V \cup \{0\}$  is in general position. Then*

$$\left| \left\{ \{F, G\} : F, G \in \binom{V}{d+1}, F \cap G = \emptyset, 0 \in \text{conv } F \cap \text{conv } G \right\} \right| \text{ is even.}$$

In particular, if there is one such pair  $\{F, G\}$ , then there is another.

Before we move on to the proof, let us derive Lemma 7 from this. It suffices to handle the case where the set  $T \cup T' \cup \{0\}$  is in general position. To justify this, we make two observations. The first is that general position is achieved by *some* arbitrarily small (for example, random) perturbation of  $T \cup T'$ . The second is that the property of *not* having sets  $S, S'$  as required is maintained under *any* sufficiently small perturbation. Putting the two observations together, we see that if Lemma 7 holds for  $T \cup T' \cup \{0\}$  in general position, then it holds for all  $T, T'$ .

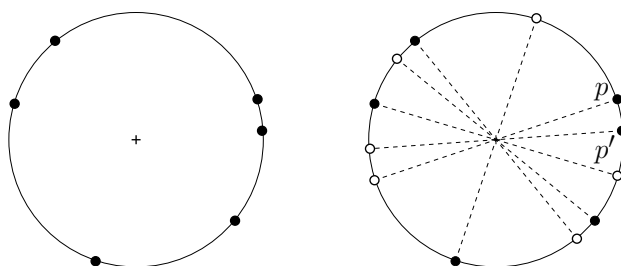
Then, by Lemma 9 with  $V = T \cup T'$ , there is another pair  $\{S, S'\}$  such that  $S \cup S' = T \cup T', 0 \in \text{conv}(S) \cap \text{conv}(S')$ . At most one of these pairs can yield nested simplices: the outer simplex of a nested pair coincides with  $\text{conv}(T \cup T')$  and is therefore uniquely determined. Hence, one of the two pairs yields crossing simplices.

► Remark. We note that this statement and its application is quite unusual. Typically, one proves that the number of objects of a certain type is always odd and thus at least one object of the type exists.

### 2.1 Geometric proof of the parity Lemma 9 for $d = 2$

The purely combinatorial proof that we give below for all  $d$  is not difficult, but does not provide any geometric intuition. We therefore start with a simple proof in the plane.

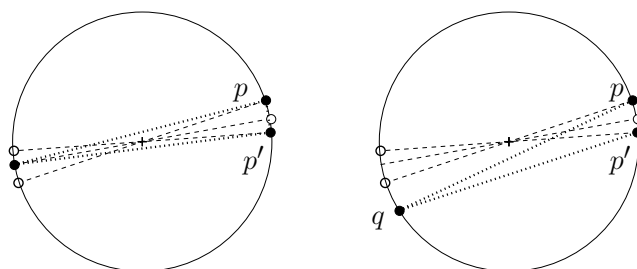
Consider a set  $V$  of  $2(d + 1) = 6$  points in the plane. We remark that the statement is invariant under scaling points, and thus we may assume that all points lie on a circle with the center in the origin. Due to general position, no two points from  $V$  lie on a line passing through the origin; see Figure 3 (left).



■ **Figure 3** 6 points in the plane (black) and their mirror images (white); there must be two consecutive black points  $p, p'$ .

Now we reflect each point at the origin and obtain another 6 points, drawn in white in Figure 3 (right). We observe that in the circular order of points, there must be two consecutive ones of the same color. Indeed, an alternating pattern (black, white, black, white, ...) would lead to pairs of antipodal points having the *same* color. Let  $p, p'$  be two consecutive points of the same color; by going to the antipodal points if necessary, we may assume that they are black and hence belong to  $V$ .

We make two observations (actually, just one). (i)  $p$  and  $p'$  cannot belong to a triple  $F \subset V$  such that  $\mathbf{0} \in \text{conv}(F)$ , as otherwise, the third point would get reflected to a white point between  $p$  and  $p'$ ; see Figure 4 (left). (ii) For any  $q \in V \setminus \{p, p'\}$ , the two segments  $\text{conv}(\{p, q\})$  and  $\text{conv}(\{p', q\})$  have the origin on the same side; see Figure 4 (right).



■ **Figure 4** Consecutive black points are combinatorially indistinguishable.

This implies the following: if  $\{F, G\}$  is a partition of  $V$  that we count in Lemma 9, then (i)  $p$  and  $p'$  are in different parts, and (ii) swapping  $p$  and  $p'$  between the parts leads to a different partition  $\{F', G'\}$  that we also count. In other words,  $p$  and  $p'$  are “combinatorially indistinguishable” with respect to the relevant properties, and the operation of swapping them between parts establishes a matching between the partitions that we want to count. Hence, their number is even.



## 2.2 Combinatorial proof of the parity Lemma 9

Recall that for  $|V| = 2(d + 1)$ , we need to show that there is an even number of partitions  $\{F, G\}$  of  $V$  into parts of equal size  $d + 1$  such that  $\mathbf{0} \in \text{conv}(F) \cap \text{conv}(G)$ . We show that this follows from the fact that the  $(d + 1)$ -element subsets  $F$  with  $\mathbf{0} \in \text{conv}(F)$  form a *cocycle*, a concept borrowed from topology.

► **Definition 10.** Let  $n \geq k \geq 1$  be integers, and let  $V$  be a set with  $n$  elements. A family  $\mathcal{C} \subset \binom{V}{k}$  of  $k$ -element subsets of  $V$  is a cocycle if

$$|\{F \in \mathcal{C} : F \subset M\}| \text{ is even for every } M \subset V \text{ with } |M| = k + 1.$$

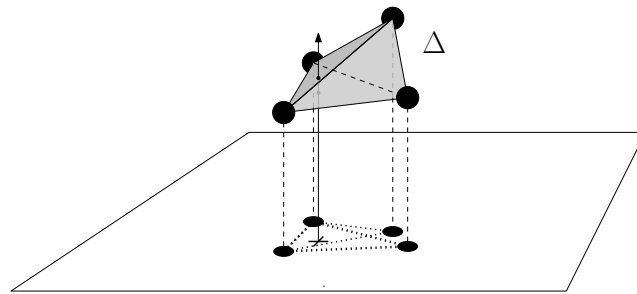
► **Example 11.** Fix a set  $D \in \binom{V}{k-1}$ . Then  $\delta D := \{F \in \binom{V}{k} : D \subset F\}$  is a cocycle. Indeed, a  $(k + 1)$ -element subset  $M \subset V$  either contains no sets in  $\delta D$  (if  $D \not\subset M$ ) or exactly two sets in  $\delta D$  (if  $D = M \setminus \{p, q\}$ ).

► **Lemma 12.** Let  $V \subset \mathbb{R}^d$  be such that  $V \cup \{\mathbf{0}\}$  is in general position. Then

$$\mathcal{C}(V) := \left\{ F \in \binom{V}{d+1} : \mathbf{0} \in \text{conv } F \right\}$$

is a cocycle.

**Proof.** Let  $M := \{v_1, \dots, v_{d+2}\} \subset V$ . Lift the points to dimension  $d + 1$  such that the convex hull of the lifted point set  $\hat{M}$  is a full-dimensional simplex  $\Delta$ ; see Figure 5.



■ **Figure 5** Illustration of the geometric proof of Lemma 12 when  $d = 2$ .

Any set  $\hat{F} \subset \hat{M}$  of size  $d + 1$  spans a facet of  $\Delta$ , and we have  $\mathbf{0} \in \text{conv}(F)$  if and only if the vertical line through  $\mathbf{0}$  intersects that facet. As  $V \cup \{\mathbf{0}\}$  is in general position, this line intersects the convex set  $\Delta$  in a line segment, if it intersects it at all; hence, the line intersects the boundary of  $\Delta$  in zero or two points, each lying in the interior of some facet. Thus,  $|\{F \in \mathcal{C}(V) : F \subset M\}| \in \{0, 2\}$ .

We remark that there is also an elementary linear algebra version of this “proof by picture” (omitted in this extended abstract). ◀

By Lemma 12, Lemma 9 is now simply a special case of the following main result of this section.

► **Theorem 13.** Let  $k \geq 1$ ,  $|V| = 2k$  and let  $\mathcal{C} \subset \binom{V}{k}$  be a cocycle. Set

$$\mathcal{P}_{\mathcal{C}} := \{ \{F, G\} : F, G \in \mathcal{C}, F \cap G = \emptyset \}.$$

Then  $|\mathcal{P}_{\mathcal{C}}|$  is even.



### 38:8 The Crossing Tverberg Theorem

For the proof, we need one more concept. For a family  $\mathcal{D} \subset \binom{V}{k-1}$ , we define the *coboundary*

$$\delta\mathcal{D} = \left\{ F \in \binom{V}{k} : F \text{ contains an odd number of sets } D \in \mathcal{D} \right\}.$$

This naturally extends the definition  $\delta D = \{F \in \binom{V}{k} : D \subset F\}$  for  $D \in \binom{V}{k-1}$  from our previous example. We call  $\delta D$  the *elementary cocycle* associated with  $D$ .

To prove Theorem 13, we use the following basic fact.

► **Lemma 14.** *For every cocycle  $\mathcal{C} \subset \binom{V}{k}$  there exists a family  $\mathcal{D} \subset \binom{V}{k-1}$  such that  $\mathcal{C} = \delta\mathcal{D}$ . Equivalently,*

$$\mathcal{C} = \delta D_1 \oplus \delta D_2 \oplus \dots \oplus \delta D_m,$$

where  $\mathcal{D} = \{D_1, \dots, D_m\}$  and  $\oplus$  denotes symmetric difference.

**Proof of Theorem 13.** The statement is trivially true when  $\mathcal{C} = \emptyset$  is the empty cocycle.

Thus, by Lemma 14, it suffices to show that the parity of  $|\mathcal{P}_{\mathcal{C}}|$  does not change when we take the symmetric difference with an elementary cocycle, i.e.,

$$|\mathcal{P}_{\mathcal{C}}| \equiv |\mathcal{P}_{\mathcal{C} \oplus \delta D}| \pmod{2} \tag{1}$$

for any  $D \in \binom{V}{k-1}$ .

▷ **Claim 15.**  $\mathcal{P}_{\mathcal{C} \oplus \delta D} = \mathcal{P}_{\mathcal{C}} \oplus \mathcal{R}$ , where

$$\mathcal{R} := \left\{ \{F, G\} : F \in \delta D, G \in \mathcal{C}, F \cap G = \emptyset \right\}.$$

*Proof.* To see this, note that for pairs  $\{F, G\}$  with  $D \not\subset F$  and  $D \not\subset G$  nothing changes, i.e., such a pair is either in both  $\mathcal{P}_{\mathcal{C}}$  and  $\mathcal{P}_{\mathcal{C} \oplus \delta D}$  or in neither. Thus, it suffices to consider pairs  $\{F, G\}$ ,  $F \cap G = \emptyset$ , such that  $D$  is contained in one of the two sets. Without loss of generality,  $D \subset F$  and  $G \subset V \setminus D$ .

Since  $D \not\subset G$ , we have  $G \in \mathcal{C}$  if and only if  $G \in \mathcal{C} \oplus \delta D$ . Since  $D \subset F$ , we have  $F \in \delta D$ , and hence,  $F \in \mathcal{C}$  if and only if  $F \notin \mathcal{C} \oplus \delta D$ . Thus,  $\{F, G\} \in \mathcal{P}_{\mathcal{C} \oplus \delta D}$  if and only if  $\{F, G\} \in \mathcal{R}$  and  $\{F, G\} \notin \mathcal{P}_{\mathcal{C}}$ . This proves the claim. ◀

Since  $\mathcal{C}$  is a cocycle and  $|V \setminus D| = k+1$ , there is an even number of  $G \in \mathcal{C}$  with  $G \subset V \setminus D$ . Since  $|V| = 2k$ , each such  $G$  determines a unique  $F = V \setminus G$ ,  $F \in \delta D$ , with  $\{F, G\} \in \mathcal{R}$ . Thus,

$$|\mathcal{R}| = |\{G \in \mathcal{C} : G \subset V \setminus D\}| \equiv 0 \pmod{2}.$$

Combined with the claim, we get that (1) holds, which concludes the proof of the theorem. ◀

For the sake completeness, we also prove Lemma 14.

**Proof of Lemma 14.** Let  $\mathcal{C} \subset \binom{V}{k}$  be a cocycle. Choose an arbitrary element  $v \in V$ . Define

$$\mathcal{D} := \mathcal{D}_v := \left\{ D \in \binom{V \setminus v}{k-1} : D \cup \{v\} \in \mathcal{C} \right\}.$$

We claim that

$$\mathcal{C} = \delta\mathcal{D}.$$

Let  $F \in \binom{V}{k}$ . We distinguish two cases:

- (1) If  $v \in F$  then, by the definition of  $\mathcal{D} \subset \binom{V \setminus \{v\}}{k-1}$ , we have  $F \setminus \{v\} \in \mathcal{D}$  if and only if  $F \in \mathcal{C}$ . Moreover,  $F \setminus \{v\}$  is the only set in  $\binom{V \setminus \{v\}}{k-1}$  that  $F$  contains. Thus,

$$F \in \mathcal{C} \Leftrightarrow F \in \delta\mathcal{D} \quad \text{in this case.}$$

- (2) Assume  $v \notin F$ . Then  $M := F \cup \{v\}$  has  $k + 1$  elements. Since  $\mathcal{C}$  is a cocycle,  $M$  contains an even number of sets from  $\mathcal{C}$ . Thus,

$$\begin{aligned} F \in \mathcal{C} &\Leftrightarrow M \text{ contains an odd number of } G \in \mathcal{C} \text{ with } v \in G \\ &\Leftrightarrow F \text{ contains an odd number of sets } D = G \setminus \{v\} \in \mathcal{D} \\ &\Leftrightarrow F \in \delta\mathcal{D}. \end{aligned}$$

### 3 Discussion

#### 3.1 A Topological version of Theorem 3

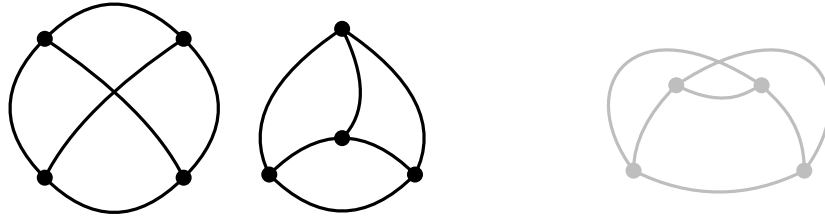
If  $r$  is a prime power, then the Tverberg theorem admits a topological generalization, known as the *Topological Tverberg theorem*:

► **Theorem 16.** *If  $d \in \mathbb{N}$  and if  $r$  is a prime power, then for every continuous map from the  $(d + 1)(r - 1)$ -dimensional simplex  $\Delta_{(d+1)(r-1)}$  to  $\mathbb{R}^d$ , there exist  $r$  pairwise disjoint faces of  $\Delta_{(d+1)(r-1)}$  whose images intersect in a common point.*

This result was first proved in the case when  $r$  is a prime by Bárány, Shlosman and Szücs [5] and later extended to prime powers by Özaydin [18]. On the other hand, Theorem 16 is false if  $r$  is not a prime power: By work of Mabillard and Wagner [14] and a result of Özaydin [18], a closely related result (the *generalized Van Kampen–Flores theorem*) is false whenever  $r$  is not a prime power and, as observed by Frick [12], the failure of Theorem 16 for  $r$  not a prime power follows from this by a reduction due to Gromov [13] and to Blagojević, Frick, and Ziegler [7]. The lowest dimension in which counterexamples are known to exist is  $d = 2r$  [3, 14]. We refer to the recent surveys [6, 8, 23, 27] for more background on the Topological Tverberg theorem and its history.

In the same vein, it is natural to wonder if our Theorem 3 also extends to the topological setting. The straightforward approach to generalize our result would be to use the Topological Tverberg theorem instead of the Tverberg theorem and then keep fixing the pairs of simplices whose boundaries do not mutually intersect. To this end we need an adaptation of Lemma 12 and the fixing procedure to the topological setting. The rest of our argument is free of any geometry except for the use of Carathéodory’s theorem which is not really crucial. While extending Lemma 12 is easy, showing the termination of the fixing procedure appears to be quite difficult except under the scenario which we discuss below. First, we discuss planar extensions of Theorem 3. An *arrangement of pseudolines*  $\mathbb{P}$  is a finite set of not self-intersecting open arcs, called *pseudolines*, in  $\mathbb{R}^2$  such that (i) For every pair  $P_1, P_2 \in \mathbb{P}$  of two distinct pseudolines,  $P_1$  and  $P_2$  intersect transversely in a single point, and (ii)  $\mathbb{R}^2 \setminus P$  is not connected for every  $P \in \mathbb{P}$ . A drawing of a complete graph on  $n$  vertices  $K_n$  in the plane is *pseudolinear* if the edges can be extended to an arrangement of pseudolines.

In the plane, it is not hard to see that Theorem 3 and its proof almost extends to the setting of pseudolinear drawings of complete graphs. Since the Topological Tverberg theorem is only valid for prime powers  $r$ , the number of pairwise crossing triangles is slightly smaller than the number of vertices divided by 3.



■ **Figure 6** The allowed drawings of  $K_4$  in a pseudolinear drawing of  $K_n$  (left). The forbidden drawing of  $K_4$  in a pseudolinear drawing of  $K_n$  (right).

► **Theorem 17.** *In a pseudolinear drawing of a complete graph  $K_{3n}$  we can find  $m = (1 - o(1))n$  vertex-disjoint and pairwise crossing triangles. Moreover, the topological discs bounded by these triangles intersect in a common point.*

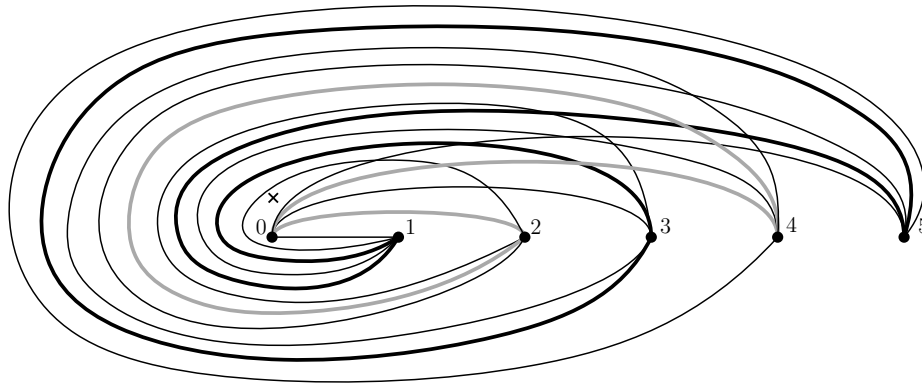
**Proof.** Let  $\mathcal{D}$  be a pseudolinear drawing of  $K_{3n}$ . We put  $m$  to be the largest prime power not larger than  $n$ . By the asymptotic law of distribution of prime numbers  $m = (1 - o(1))n$ . Next, apply the Topological Tverberg theorem with  $r = m$  and  $d = 2$  to a map  $\mu : \Delta_{3m-3} \rightarrow \mathbb{R}^2$  which extends  $\mathcal{D}$  as follows. We define  $\mu$  on the 1-dimensional skeleton of  $\Delta_{3m-3}$  as a restriction of  $\mathcal{D}$  to some  $K_{3m-2}$ . Note that every triangle of  $K_{3m-2}$  is drawn by  $\mathcal{D}$  as a closed arc without self intersections. The map  $\mu$  extends to the 2-dimensional skeleton of  $\Delta_{3m-3}$  so that every 2-dimensional face is mapped homeomorphically in  $\mathbb{R}^2$ . We define the map  $\mu$  on the rest of  $\Delta_{3m-3}$  arbitrarily while maintaining continuity.

Analogously to the proof of Theorem 3, an application of the Topological Tverberg theorem gives us  $m - 1$  disjoint 2-dimensional faces  $F_1, \dots, F_{m-1}$  of  $\Delta_{3m-3}$  whose images under  $\mu$  intersect in a common point. To this end we apply Carathéodory’s theorem for drawings of complete graphs [4, Lemma 4.7] instead of the original version of Carathéodory’s theorem. Let  $T_i$  denote the boundary of  $F_i$ , for  $i = 1, \dots, m - 1$ . Note that each  $T_i$  is a triangle in  $K_{3m-2}$ . If all pairs  $T_i$  and  $T_j$  are crossing then we are done. Otherwise, we perform the fixing operations, which can be done since Lemma 12 easily extends to the setting in which we replace simplices by images of 2-dimensional faces of  $\Delta_{3m-3}$  under  $\mu$ .

The procedure of applying successively the fixing operation terminates due to the following argument. First, observe that every four vertices in a pseudolinear drawing of a complete graph induce either a crossing free drawing of  $K_4$  or a drawing of  $K_4$  with exactly one pair of crossing edges in the interior of a disc bounded by a crossing free 4-cycle, see Figure 6 for an illustration. It follows that if  $\mu(T_i) \cap \mu(T_j) = \emptyset$ , and let’s say  $\mu(F_i) \subset \mu(F_j)$ , then the restriction of  $\mu$  to the subgraph of  $K_{3m-2}$  induced by  $V(T_i) \cup V(T_j)$  is contained in  $\mu(F_j)$ . Indeed, otherwise there exists a vertex  $u \in V(T_i)$  and  $v \in V(T_j)$  such that  $\mu(uv)$  crosses an edge of  $T_j$ , let us denote it by  $wz$ , that is not incident to  $v$ . Then the restriction of  $\mu$  to the subgraph of  $K_{3m-2}$  induced by  $\{u, v, w, z\}$  is a drawing of  $K_4$  that is not pseudolinear (contradiction). Therefore the volume argument goes through if we consider, say, volumes of  $\mu(F_i)$ ’s. ◀

A drawing of a graph in the plane is *simple* if every pair of edges intersect at most once either at a common end point or in a proper crossing. Clearly, all pseudolinear drawings of complete graphs are also simple, but not vice-versa, as illustrated in the last drawing in Figure 6. Hence, it might be worthwhile to extend Theorem 17 to simple drawings of complete graphs.

If we want the interiors of the triangles to be pairwise intersecting, we only know that we can take  $m$  at least  $\Omega(\log^{1/6} n)$  which is easily derived from the following result of Pach, Solymosi and Tóth [20]. Every simple drawing of  $K_n$  contains a drawing of  $K_m$  that is weakly



■ **Figure 7** The twisted drawing of  $K_6$ . The bold grey and black edges form boundaries of two crossing triangles. The symbol  $x$  marks a point contained in the interior of these two triangles.

isomorphic to a so-called convex complete graph or a twisted complete graph, see Figure 7, for which Theorem 17 holds. We omit the proof of the latter which is rather straightforward. For example, if in a twisted drawing of  $K_{3n}$  the vertices are labeled as indicated in the figure,  $\{\{0 + i, n + i, 2n + i\} \mid i = 0, \dots, n - 1\}$  is a crossing Tverberg partition.

If we do not insist on the interiors of the triangles to be pairwise intersecting, we know that  $m$  can be taken to be at least  $\Omega(n^\varepsilon)$  for some small  $\varepsilon > 0$  by the following result of Fox and Pach [11]. Every simple drawing of  $K_n$  contains  $\Omega(n^\varepsilon)$  pairwise crossing edges for some  $\varepsilon > 0$ .

Theorem 17 could be generalized to hold for an appropriate high dimensional analog of pseudolinear drawings. Since this would require introducing many technical terms and would not offer substantially interesting content we refrain from doing so.

### 3.2 Stronger conditions on crossings

A pair of vertex disjoint  $(\lceil d/2 \rceil - 1)$ -dimensional simplices in general position in  $d$ -space does not intersect. Hence, the pairwise intersection of the boundaries of  $\text{conv}(X_i)$ 's in the conclusion of Theorem 3, cannot be strengthened to the pairwise intersection of lower than  $\lceil d/2 \rceil$ -dimensional skeleta of the boundaries.

Nevertheless, for  $d = 3$  one may ask if in the setting of Lemma 7, we get a stronger property along the following lines. Can we guarantee the existence of a pair of vertex-disjoint tetrahedra  $\{S, S'\}$  that both contain the origin and such that the boundary of a 2-dimensional face  $F$  of  $S$  is linked with the boundary of a 2-dimensional face  $G$  of  $S'$ , meaning that the boundary of  $F$  intersects  $G$  and the boundary of  $G$  intersects  $F$ ? Again, the answer to this question is negative. Stefan Felsner and Manfred Scheucher (personal communication) found the following set of 8 points:

$$\begin{aligned} (3, -2, 2), \quad (2, -5, 3), \quad (-3, 0, -4), \quad (-1, 2, 0), \\ (1, -5, -4), \quad (4, 1, -2), \quad (-2, -5, -4), \quad (-3, 1, 3). \end{aligned}$$

This set determines a pair of disjoint tetrahedra both containing the origin  $(0, 0, 0)$ , but no two disjoint linked tetrahedra both containing the origin.

The example was found using a SAT solver who found an abstract order type with the required property. A realization of the order type with actual points was obtained with a randomized procedure.

### 3.3 Computational complexity of finding a crossing Tverberg partition

A natural question is whether we can find the partition of the point set given by Theorem 3 efficiently, i.e., in polynomial time in the size of  $X$ . A straightforward way to construct an algorithm is to make the proof of Theorem 3 algorithmic. To this end we first need an algorithm for finding a Tverberg partition and also a Tverberg point. Unfortunately, it is neither known whether this can be done in polynomial time, nor are there hardness results [9, Section 3.4]. Since a Tverberg partition always exists, the decision problem is trivial, so NP-completeness cannot apply. It might still be possible to prove completeness of the problem for a suitable subclass of TFNP (containing search problems for which a solution is guaranteed to exist). It has been shown that the problem is contained in two such subclasses that have complete problems, namely PPAD and PLS [15, Theorem 4.9]. While this makes it unlikely that the problem is complete for either of them, it could well be contained in and turn out to be complete for a subclass of  $\text{PPAD} \cap \text{PLS}$ , a number of which have been introduced and studied in recent years; see [10] and the references therein.

The only closely related hardness result we are aware of is the one by Teng [24, Theorem 8.14], who proved that checking whether a given point is a Tverberg point of a given point set is NP-complete.

A line of research on finding an approximate Tverberg partition efficiently was initiated by Miller and Sheehy [16] and further developed in [17, 22]. In particular, Mulzer and Werner [17] showed that it is possible to find in time  $d^{O(\log d)|X|}$  an approximate Tverberg partition of size  $\left\lceil \frac{|X|}{4(d+1)^3} \right\rceil$ , whereas Theorem 1 guarantees the partition of size  $\left\lceil \frac{|X|}{d+1} \right\rceil$ .

If we aim only at an approximate algorithmic version of our Theorem 3 along the lines of the result of Mulzer and Werner, we face the problem of efficiently fixing an (approximate) Tverberg partition to make it crossing. Due to the fact that our termination argument for the iterated *Fixing Pairs* procedure relies on progress in the lexicographical ordering of the simplex volumes, we may potentially need exponentially many (in the size of  $X$ ) iterations before we arrive at a crossing partition. We leave it as an interesting open problem to prove or disprove that there is always a way to invoke the *Fixing Pairs* operation only polynomially (or least subexponentially) many times in order to arrive at a partition required by Theorem 3.

---


#### References

- 1 José Luis Alvarez-Rebollar, Jorge Cravioto Lagos, and Jorge Urrutia. Crossing families and self crossing Hamiltonian cycles. *XVI Encuentros de Geometría Computacional*, page 13, 2015.
- 2 B. Aronov, P. Erdős, W. Goddard, D. J. Kleitman, M. Klugerman, J. Pach, and L. J. Schulman. Crossing families. *Combinatorica*, 14(2):127–134, 1994.
- 3 Sergey Avvakumov, Isaac Mabillard, Arkadiy Skopenkov, and Uli Wagner. Eliminating higher-multiplicity intersections, III. Codimension 2. Preprint, [arXiv:1511.03501](https://arxiv.org/abs/1511.03501), 2015.
- 4 Martin Balko, Radoslav Fulek, and Jan Kynčl. Crossing Numbers and Combinatorial Characterization of Monotone Drawings of  $K_n$ . *Discrete & Computational Geometry*, 53(1):107–143, 2015.
- 5 Imre Bárány, Senya B Shlosman, and András Szücs. On a topological generalization of a theorem of Tverberg. *Journal of the London Mathematical Society*, 2(1):158–164, 1981.
- 6 Imre Bárány and Pablo Soberón. Tverberg’s theorem is 50 years old: A survey. *Bulletin of the American Mathematical Society*, 55(4):459–492, June 2018. doi:10.1090/bull/1634.
- 7 Pavle V. M. Blagojević, Florian Frick, and Günter M. Ziegler. Tverberg plus constraints. *Bull. Lond. Math. Soc.*, 46(5):953–967, 2014.

- 8 Pavle V. M. Blagojević and Günter M. Ziegler. Beyond the Borsuk–Ulam Theorem: The Topological Tverberg Story. In: *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek* (M. Loeb, J. Nešetřil, and R. Thomas, eds.), Springer, pp. 273–341, 2017.
- 9 Jesus A. de Loera, Xavier Goaoc, Frédéric Meunier, and Nabil Mustafa. The discrete yet ubiquitous theorems of Carathéodory, Helly, Sperner, Tucker, and Tverberg. Preprint, [arxiv:1706.05975](https://arxiv.org/abs/1706.05975), 2018.
- 10 John Fearnley, Spencer Gordon, Ruta Mehta, and Rahul Savani. Unique End of Potential Line. *CoRR*, abs/1811.03841, 2018. [arXiv:1811.03841](https://arxiv.org/abs/1811.03841).
- 11 Jacob Fox and János Pach. Coloring  $K_k$ -free intersection graphs of geometric objects in the plane. *European Journal of Combinatorics*, 33(5):853–866, 2012.
- 12 Florian Frick. Counterexamples to the topological Tverberg conjecture. *Oberwolfach Reports*, 12(1):318–321, 2015.
- 13 Mikhail Gromov. Singularities, expanders and topology of maps. Part 2: From combinatorics to topology via algebraic isoperimetry. *Geom. Funct. Anal.*, 20(2):416–526, 2010.
- 14 Isaac Mabillard and Uli Wagner. Eliminating Higher-Multiplicity Intersections, I. A Whitney Trick for Tverberg-Type Problems. Preprint, [arXiv:1508.02349](https://arxiv.org/abs/1508.02349), 2015. An extended abstract appeared (under the title *Eliminating Tverberg points, I. An analogue of the Whitney trick*) in Proc. 30th Ann. Symp. Comput. Geom., 2014, pp. 171–180.
- 15 Frédéric Meunier, Wolfgang Mulzer, Pauline Sarrazolles, and Yannik Stein. The rainbow at the end of the line—a PPAD formulation of the colorful Carathéodory theorem with applications. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1342–1351. SIAM, 2017.
- 16 Gary L Miller and Donald R Sheehy. Approximate centerpoints with proofs. *Computational Geometry*, 43(8):647–654, 2010.
- 17 Wolfgang Mulzer and Daniel Werner. Approximating Tverberg points in linear time for any fixed dimension. *Discrete & Computational Geometry*, 50(2):520–535, 2013.
- 18 Murad Özaydin. Equivariant maps for the symmetric group. Preprint, 1987. Available at <http://minds.wisconsin.edu/handle/1793/63829>.
- 19 János Pach, Natan Rubin, and Gábor Tardos. Planar point sets determine many pairwise crossing segments. In *Proceedings of the 51st Annual ACM Symposium on the Theory of Computing*. ACM, to appear, 2019.
- 20 János Pach, József Solymosi, and Géza Tóth. Unavoidable configurations in complete topological graphs. *Discrete & Computational Geometry*, 30(2):311–320, 2003.
- 21 Johann Radon. Mengen konvexer Körper, die einen gemeinsamen Punkt enthalten. *Mathematische Annalen*, 83(1-2):113–115, 1921.
- 22 David Rolnick and Pablo Soberón. Algorithms for Tverberg’s theorem via centerpoint theorems. Preprint, [arXiv:1601.03083](https://arxiv.org/abs/1601.03083), 2016.
- 23 Arkadiy B. Skopenkov. A user’s guide to the topological Tverberg conjecture. *Russian Mathematical Surveys*, 73(2):323, 2018.
- 24 Shang-Hua Teng. *Points, spheres, and separators: a unified geometric approach to graph partitioning*. PhD thesis, Carnegie Mellon University, 1992.
- 25 László Fejes Tóth. Eine Kennzeichnung des Kreises. *Elemente der Mathematik*, 22:25–48, 1967.
- 26 Helge Tverberg. A generalization of Radon’s theorem. *Journal of the London Mathematical Society*, 1(1):123–128, 1966.
- 27 Rade T. Živaljević. Topological Methods in Discrete Geometry. In Jacob E. Goodman, Joseph O’Rourke, and Csaba D. Tóth, editors, *Handbook of discrete and computational geometry*, CRC Press Ser. Discrete Math. Appl., chapter 21. CRC, Boca Raton, FL, 2018.



# $\mathbb{Z}_2$ -Genus of Graphs and Minimum Rank of Partial Symmetric Matrices

Radoslav Fulek 

IST Austria, Am Campus 1, Klosterneuburg 3400, Austria  
radoslav.fulek@ist.ac.at

Jan Kynčl 

Department of Applied Mathematics, Charles University, Faculty of Mathematics and Physics,  
Malostranské nám. 25, 118 00 Praha 1, Czech Republic  
kyncl@kam.mff.cuni.cz

---

## Abstract

---

The *genus*  $g(G)$  of a graph  $G$  is the minimum  $g$  such that  $G$  has an embedding on the orientable surface  $M_g$  of genus  $g$ . A drawing of a graph on a surface is *independently even* if every pair of nonadjacent edges in the drawing crosses an even number of times. The  $\mathbb{Z}_2$ -genus of a graph  $G$ , denoted by  $g_0(G)$ , is the minimum  $g$  such that  $G$  has an independently even drawing on  $M_g$ .

By a result of Battle, Harary, Kodama and Youngs from 1962, the graph genus is additive over 2-connected blocks. In 2013, Schaefer and Štefankovič proved that the  $\mathbb{Z}_2$ -genus of a graph is additive over 2-connected blocks as well, and asked whether this result can be extended to so-called 2-amalgamations, as an analogue of results by Decker, Glover, Huneke, and Stahl for the genus. We give the following partial answer. If  $G = G_1 \cup G_2$ ,  $G_1$  and  $G_2$  intersect in two vertices  $u$  and  $v$ , and  $G - u - v$  has  $k$  connected components (among which we count the edge  $uv$  if present), then  $|g_0(G) - (g_0(G_1) + g_0(G_2))| \leq k + 1$ . For complete bipartite graphs  $K_{m,n}$ , with  $n \geq m \geq 3$ , we prove that  $\frac{g_0(K_{m,n})}{g(K_{m,n})} = 1 - O(\frac{1}{n})$ . Similar results are proved also for the Euler  $\mathbb{Z}_2$ -genus.

We express the  $\mathbb{Z}_2$ -genus of a graph using the minimum rank of partial symmetric matrices over  $\mathbb{Z}_2$ ; a problem that might be of independent interest.

**2012 ACM Subject Classification** Mathematics of computing  $\rightarrow$  Graphs and surfaces; Mathematics of computing  $\rightarrow$  Computations on matrices

**Keywords and phrases** graph genus, minimum rank of a partial matrix, Hanani–Tutte theorem, graph amalgamation

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.39

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1903.08637>.

**Funding** *Radoslav Fulek*: The author gratefully acknowledges support from Austrian Science Fund (FWF): M2281-N35.

*Jan Kynčl*: Supported by project 19-04113Y of the Czech Science Foundation (GAČR), by the PRIMUS/17/SCI/3 project of Charles University and by Charles University project UNCE/SCI/004.

**Acknowledgements** We are grateful to anonymous referees for comments that helped us to improve the presentation of the results.

## 1 Introduction

The *genus*  $g(G)$  of a graph  $G$  is the minimum  $g$  such that  $G$  has an embedding on the orientable surface  $M_g$  of genus  $g$ . Similarly, the *Euler genus*  $eg(G)$  of  $G$  is the minimum  $g$  such that  $G$  has an embedding on a surface of Euler genus  $g$ . We say that two edges in a graph are *independent* (also *nonadjacent*) if they do not share a vertex. The  $\mathbb{Z}_2$ -genus  $g_0(G)$  and *Euler  $\mathbb{Z}_2$ -genus*  $eg_0(G)$  of  $G$  are defined as the minimum  $g$  such that  $G$  has a drawing on  $M_g$  and a surface of Euler genus  $g$ , respectively, with every pair of independent edges crossing an even number of times. Clearly,  $g_0(G) \leq g(G)$  and  $eg_0(G) \leq eg(G)$ .



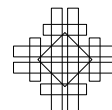
© Radoslav Fulek and Jan Kynčl;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 39; pp. 39:1–39:16

Leibniz International Proceedings in Informatics



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





The definition of the  $\mathbb{Z}_2$ -genus and Euler  $\mathbb{Z}_2$ -genus is motivated by the strong Hanani–Tutte theorem [17, 35] stating that a graph is planar if and only if its  $\mathbb{Z}_2$ -genus is 0. Many variants and extensions of the theorem have been proved [6, 14, 23, 29, 31], and they found various applications in combinatorial and computational geometry; see the survey by Schaefer [28].

It had been a long-standing open problem whether the strong Hanani–Tutte theorem extends to surfaces other than the plane and projective plane, although the problem was first explicitly stated in print by Schaefer and Štefankovič [30] in 2013. They conjectured that the strong Hanani–Tutte theorem extends to every orientable surface, that is,  $g_0(G) = g(G)$  for every graph  $G$ . They proved that a minimal counterexample to their conjecture must be 2-connected; this is just a restatement of their block additivity result, which we discuss later in this section. In a recent manuscript [13], we provided an explicit construction of a graph  $G$  for which  $g(G) = 5$  and  $g_0(G) \leq 4$ , thereby refuting the conjecture. Nevertheless, the conjecture by Schaefer and Štefankovič [30] that  $eg_0(G) = eg(G)$  for every graph  $G$  might still be true.

The conjecture has been verified only for graphs  $G$  with  $eg(G) \leq 1$ : Pelsmajer, Schaefer and Stasi [24] proved that the strong Hanani–Tutte theorem extends to the projective plane, using the characterization of projective planar graphs by an explicit list of forbidden minors. Recently, Colin de Verdière et al. [8] gave a constructive proof of the same result.

Schaefer and Štefankovič [30] also formulated a weaker form of their conjecture about the  $\mathbb{Z}_2$ -genus, stating that there exists a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  such that  $g(G) \leq f(g_0(G))$  for every graph  $G$ . Assuming the validity of an unpublished Ramsey-type result by Robertson and Seymour, the existence of such  $f$  follows as a corollary from our recent result [15] stating that  $g_0(G) = g(G)$  for the graphs  $G$  in the so-called family of Kuratowski minors. Regarding the asymptotics of  $f$ , we do not have any explicit upper bound on  $f$ , and the existence of  $G$  with  $g(G) = 5$  and  $g_0(G) \leq 4$  implies that  $f(k) \geq 5k/4$  [13, Corollary 11].

As the next step towards a good understanding of the relation between the (Euler) genus and the (Euler)  $\mathbb{Z}_2$ -genus we provide further indication of their similarity. We will build upon techniques introduced in [30] and [15], and reduce the problem of estimating the (Euler)  $\mathbb{Z}_2$ -genus to the problem of estimating the minimum rank of partial symmetric matrices over  $\mathbb{Z}_2$ .

First, we extend our recent result determining the  $\mathbb{Z}_2$ -genus of  $K_{3,n}$  [15, Proposition 18] in a weaker form to all complete bipartite graphs. A classical result by Ringel [5, 26, 27], [22, Theorem 4.4.7], [16, Theorem 4.5.3] states that for  $m, n \geq 2$ , we have  $g(K_{m,n}) = \left\lceil \frac{(m-2)(n-2)}{4} \right\rceil$  and  $eg(K_{m,n}) = \left\lceil \frac{(m-2)(n-2)}{2} \right\rceil$ .

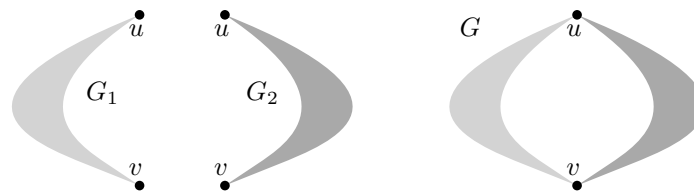
► **Theorem 1.** *If  $n \geq m \geq 3$ , then*

$$g_0(K_{m,n}) \geq \frac{(n-2)(m-2)}{4} - \frac{m-3}{2} \quad \text{and}$$

$$eg_0(K_{m,n}) \geq \frac{(n-2)(m-2)}{2} - (m-3).$$

Our second result is a  $\mathbb{Z}_2$ -variant of the results of Stahl [32], Decker, Glover and Huneke [11, 12], Miller [20] and Richter [25] showing that the genus and Euler genus of graphs are almost additive over 2-amalgamations, which we now describe in detail.

We say that a graph  $G$  is a  $k$ -amalgamation of graphs  $G_1$  and  $G_2$  (with respect to vertices  $x_1, \dots, x_k$ ) if  $G = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$  and  $V(G_1) \cap V(G_2) = \{x_1, \dots, x_k\}$ , and we write  $G = \amalg_{x_1, \dots, x_k}(G_1, G_2)$ . See Figure 1 for an illustration.



■ **Figure 1** A 2-amalgamation  $G = \Pi_{u,v}(G_1, G_2)$ .

An old result of Battle, Harary, Kodama and Youngs [4], [22, Theorem 4.4.2], [16, Theorem 3.5.3] states that the genus of a graph is additive over its 2-connected blocks. In other words, if  $G$  is a 1-amalgamation of  $G_1$  and  $G_2$  then  $g(G) = g(G_1) + g(G_2)$ . Stahl and Beinecke [33, Corollary 2], [22, Theorem 4.4.3] and Miller [20, Theorem 1] proved that the same holds for the Euler genus, that is,  $eg(G) = eg(G_1) + eg(G_2)$ . Neither the genus nor the Euler genus are additive over 2-amalgamations: for example, the nonplanar graph  $K_5$  can be expressed as a 2-amalgamation of two planar graphs in several ways. Nevertheless, the additivity in this case fails only by at most 1 for the genus and by at most 2 for the Euler genus. Formally, Stahl [32] and Decker, Glover and Huneke [11, 12] proved that if  $G$  is a 2-amalgamation of  $G_1$  and  $G_2$  then  $|g(G) - (g(G_1) + g(G_2))| \leq 1$ . For the Euler genus, Miller [20] proved its additivity over edge-amalgamations, which implies  $eg(G_1) + eg(G_2) \leq eg(G) \leq eg(G_1) + eg(G_2) + 2$ . Richter [25] later proved a more precise formula for the Euler genus of 2-amalgamations with respect to a pair of nonadjacent vertices.

Schaefer and Štefankovič [30] showed that the  $\mathbb{Z}_2$ -genus and Euler  $\mathbb{Z}_2$ -genus are additive over 2-connected blocks and they asked whether  $|g_0(G) - (g_0(G_1) + g_0(G_2))| \leq 1$  if  $G$  is a 2-amalgamation of  $G_1$  and  $G_2$ , as an analogue of the result by Stahl [32] and Decker, Glover and Huneke [11, 12]. We prove a slightly weaker variant of almost-additivity over 2-amalgamations for both the  $\mathbb{Z}_2$ -genus and the Euler  $\mathbb{Z}_2$ -genus.

► **Theorem 2.** *Let  $G$  be a 2-amalgamation  $\Pi_{v,u}(G_1, G_2)$ . Let  $l$  be the total number of connected components of  $G - u - v$  in  $G$ . Let  $k = l$  if  $uv \notin E(G)$  and  $k = l + 1$  if  $uv \in E(G)$ . Then*

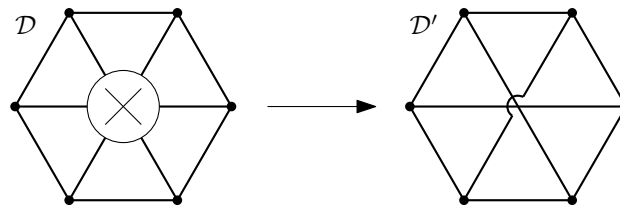
- a)  $g_0(G_1) + g_0(G_2) - (k + 1) \leq g_0(G) \leq g_0(G_1) + g_0(G_2) + 1$ , and
- b)  $eg_0(G_1) + eg_0(G_2) - (2k - 1) \leq eg_0(G) \leq eg_0(G_1) + eg_0(G_2) + 2$ .

## Organization

We give basic definitions and tools in Sections 2 and 3. In Section 4, we present linear-algebraic results lying at the heart of our arguments. In Section 5 and 6, we prove Theorem 1 and Theorem 2, respectively. In Section 6, in order to illustrate our techniques in a simpler setting, we first reprove the block additivity result for the Euler  $\mathbb{Z}_2$ -genus. We finish with concluding remarks in Section 7. Omitted proofs are in the full version.

## 2 Graphs on surfaces

We refer to the monograph by Mohar and Thomassen [22] for a detailed introduction into surfaces and graph embeddings. By a *surface* we mean a connected compact 2-dimensional topological manifold. Every surface is either *orientable* (has two sides) or *nonorientable* (has only one side). Every orientable surface  $S$  is obtained from the sphere by attaching  $g \geq 0$  *handles*, and this number  $g$  is called the *genus* of  $S$ . Similarly, every nonorientable surface  $S$



■ **Figure 2** An embedding  $\mathcal{D}$  of  $K_{3,3}$  in the plane with a single crosscap (left) and its planarization  $\mathcal{D}'$  (right).

is obtained from the sphere by attaching  $g \geq 1$  *crosscaps*, and this number  $g$  is called the (*nonorientable*) *genus* of  $S$ . The simplest orientable surfaces are the sphere (with genus 0) and the torus (with genus 1). The simplest nonorientable surfaces are the projective plane (with genus 1) and the Klein bottle (with genus 2). We denote the orientable surface of genus  $g$  by  $M_g$ , and the nonorientable surface of genus  $g$  by  $N_g$ . The *Euler genus* of  $M_g$  is  $2g$  and the Euler genus of  $N_g$  is  $g$ .

Let  $G = (V, E)$  be a graph or a multigraph with no loops, and let  $S$  be a surface. A *drawing* of  $G$  on  $S$  is a representation of  $G$  where every vertex is represented by a unique point in  $S$  and every edge  $e$  joining vertices  $u$  and  $v$  is represented by a simple curve in  $S$  joining the two points that represent  $u$  and  $v$ . If it leads to no confusion, we do not distinguish between a vertex or an edge and its representation in the drawing and we use the words “vertex” and “edge” in both contexts. We assume that in a drawing no edge passes through a vertex, no two edges touch, every edge has only finitely many intersection points with other edges and no three edges cross at the same inner point. In particular, every common point of two edges is either their common endpoint or a crossing. Let  $\mathcal{D}$  be a drawing of a graph  $G$ . We denote by  $cr_{\mathcal{D}}(e, f)$  the number of crossings between the edges  $e$  and  $f$  in  $\mathcal{D}$ . A drawing of  $G$  on  $S$  is an *embedding* if no two edges cross.

A drawing of a graph is *independently even* if every pair of independent edges in the drawing crosses an even number of times. In the literature, the notion of  $\mathbb{Z}_2$ -*embedding* is used to denote an independently even drawing [30], but also an *even drawing* [6] in which all pairs of edges cross evenly.

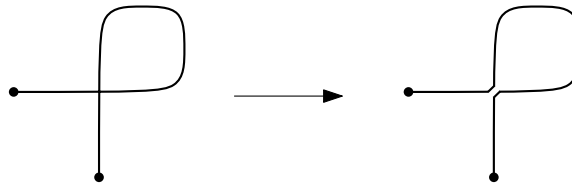
### 3 Topological and algebraic tools

#### 3.1 Combinatorial representation of drawings

Schaefer and Štefankovič [30] used the following combinatorial representation of drawings of graphs on  $M_g$  and  $N_g$ . First, every drawing of a graph on  $M_g$  can be considered as a drawing on the nonorientable surface  $N_{2g+1}$ , since  $M_g$  minus a point is homeomorphic to an open subset of  $N_{2g+1}$ . The surface  $N_h$  minus a point can be represented combinatorially as the plane with  $h$  *crosscaps*. A crosscap at a point  $x$  is a combinatorial representation of a Möbius strip whose boundary is identified with the boundary of a small circular hole centered in  $x$ . Informally, the main “objective” of a crosscap is to allow a set of curves intersect transversally at  $x$  without counting it as a crossing.

Let  $\mathcal{D}$  be a drawing of a graph  $G$  in the plane with  $h$  crosscaps. To every edge  $e \in E(G)$  we assign a vector  $y_e^{\mathcal{D}}$  (or simply  $y_e$ ) from  $\mathbb{Z}_2^h$  such that  $(y_e^{\mathcal{D}})_i = 1$  if and only if  $e$  passes an odd number of times through the  $i$ th crosscap.

Given a drawing  $\mathcal{D}$  of a graph  $G$  in the plane with  $h$  crosscaps, the *planarization* of  $\mathcal{D}$  is a drawing  $\mathcal{D}'$  of  $G$  in the plane, obtained from  $\mathcal{D}$  as follows; see Figure 2 for an illustration. We turn the crosscaps into holes, fill the holes with discs, reconnect the severed edges of



■ **Figure 3** Removing a self-crossing of an edge.

$G$  by simple curves drawn across the filling discs while avoiding creating common crossing points of three and more edges, and finally we eliminate self-crossings of edges by cutting and rerouting the edges at such crossings; see Figure 3. Since  $\mathcal{D}$  represents a drawing  $\mathcal{D}_h$  on  $N_h$ , we denote by  $\text{cr}_{\mathcal{D}}^*(e, f)$  the number of crossings between the edges  $e$  and  $f$  that occur outside crosscaps in  $\mathcal{D}$ , which is equal to  $\text{cr}_{\mathcal{D}_h}(e, f)$ . Writing  $y_e^\top y_f$  for the scalar product of  $y_e$  and  $y_f$ , we have

$$\text{cr}_{\mathcal{D}}^*(e, f) \equiv \text{cr}_{\mathcal{D}'}(e, f) + y_e^\top y_f \pmod{2}, \quad (1)$$

since  $y_e^\top y_f$  has the same parity as the number of new crossings between  $e$  and  $f$  introduced during the construction of the planarization. If  $\mathcal{D}$  represents a drawing on  $M_g$  (in the plane with  $h = 2g + 1$  crosscaps), we say that  $\mathcal{D}$  is *orientable*. This is equivalent with every cycle passing through the crosscaps an even number of times.

We will use the first two of the following three lemmata by Schaefer and Štefankovič [30].

► **Lemma 3** ([30, Lemma 5]). *Let  $G$  be a graph that has an independently even drawing  $\mathcal{D}$  on a surface  $S$  and let  $F$  be a forest in  $G$ . Let  $h = 2g + 1$  if  $S = M_g$  and  $h = g$  if  $S = N_g$ . Then  $G$  has a drawing  $\mathcal{E}$  in the plane with  $h$  crosscaps, such that*

- 1) *for every pair of independent edges  $e, f$  the number  $\text{cr}_{\mathcal{E}}^*(e, f)$  is even, and*
- 2) *every edge  $f$  of  $F$  passes through each crosscap an even number of times; that is,  $y_f^{\mathcal{E}} = 0$ .*

We will be using Lemma 3 when  $G$  is connected and  $F$  is a spanning tree of  $G$ .

► **Lemma 4** ([30, Lemma 3]). *Let  $G$  be a graph that has an orientable drawing  $\mathcal{D}$  in the plane with finitely many crosscaps such that for every pair of independent edges  $e, f$  the number  $\text{cr}_{\mathcal{D}}^*(e, f)$  is even. Let  $d$  be the dimension of the vector space generated by the set  $\{y_e^{\mathcal{D}}; e \in E(G)\}$ . Then  $G$  has an independently even drawing on  $M_{\lfloor d/2 \rfloor}$ .*

► **Lemma 5** ([30, Lemma 4]). *Let  $G$  be a graph that has a drawing in the plane with finitely many crosscaps such that for every pair of independent edges  $e, f$  the number  $\text{cr}_{\mathcal{D}}^*(e, f)$  is even. Let  $d$  be the dimension of the vector space generated by the set  $\{y_e^{\mathcal{D}}; e \in E(G)\}$ . Then  $G$  has an independently even drawing on a surface of Euler genus  $d$ .*

### 3.2 Bounding $\mathbb{Z}_2$ -genus by matrix rank

In Proposition 10 we will strengthen Lemma 4 and Lemma 5. An immediate corollary of Proposition 10 (Corollary 11 below) can be thought of as a  $\mathbb{Z}_2$ -variant of a result of Mohar [21, Theorem 3.1]. Roughly speaking, Proposition 10 says that we can upper bound the  $\mathbb{Z}_2$ -genus and Euler  $\mathbb{Z}_2$ -genus of a graph  $G$  in terms of the rank of a symmetric matrix  $A$  encoding the parity of crossings between independent edges. The entries in  $A$  representing the parity of crossings between adjacent edges, and in the case of the Euler  $\mathbb{Z}_2$ -genus also diagonal elements, can be chosen arbitrarily. The choice of such undetermined entries minimizing the rank of  $A$  will play a crucial role in the proof of Theorem 2.

We use the theory of symmetric matrices over the two-element field  $\mathbb{F}_2$ , developed by Albert [1]. Our goal will be to express a symmetric  $n \times n$  matrix  $A$  over  $\mathbb{F}_2$  as a Gram matrix of  $n$  vectors spanning a vector space of minimum possible dimension. This is equivalent to finding an  $m \times n$  matrix  $B$  of minimum rank such that  $A = B^\top B$ .

A symmetric matrix over  $\mathbb{F}_2$  is *alternate* if its diagonal contains only 0-entries<sup>1</sup>. Two square matrices  $A$  and  $B$  are *congruent* if there exists an invertible matrix  $C$  such that  $B = C^\top AC$ . We use the following two results by Albert [1], which hold over an arbitrary field.

► **Lemma 6** ([1, Theorem 3]). *The rank of an alternate matrix is even.*

► **Lemma 7** ([1, Theorem 6]). *Every non-alternate symmetric matrix is congruent to a diagonal matrix.*

MacWilliams [19] gave a concise exposition of the following result of Albert [1].

► **Lemma 8** ([19, Theorem 1]). *An invertible symmetric matrix  $A$  over  $\mathbb{F}_2$  can be factored as  $B^\top B$  for some square matrix  $B$  if and only if  $A$  is not alternate.*

We need to extend the factorization from Lemma 8 to alternate and to non-invertible matrices. In the case of non-alternate matrices we again obtain their rank factorization. We use Lemma 7 to achieve this.

► **Lemma 9.** *Let  $A$  be a symmetric  $n \times n$  matrix over  $\mathbb{F}_2$  and let  $r$  be the rank of  $A$ . If  $A$  is non-alternate, then there is an  $r \times n$  matrix  $B$  of rank  $r$  such that  $A = B^\top B$ . If  $A$  is alternate, then there is an  $(r + 1) \times n$  matrix  $B$  of rank  $r$  or  $r + 1$  such that  $A = B^\top B$ .*

**Proof.** If  $A$  is not alternate, let  $A' = A$ ; otherwise let  $A' = (a'_{ij})$  be a symmetric matrix obtained from  $A$  by adding a single row and single column, as the first row and the first column of  $A'$ , with  $a'_{11} = 1$  and  $a'_{1i} = a'_{i1} = 0$  for  $i > 1$ . Let  $r'$  be the rank of  $A'$ . Clearly, if  $A$  is alternate then  $r' = r + 1$ .

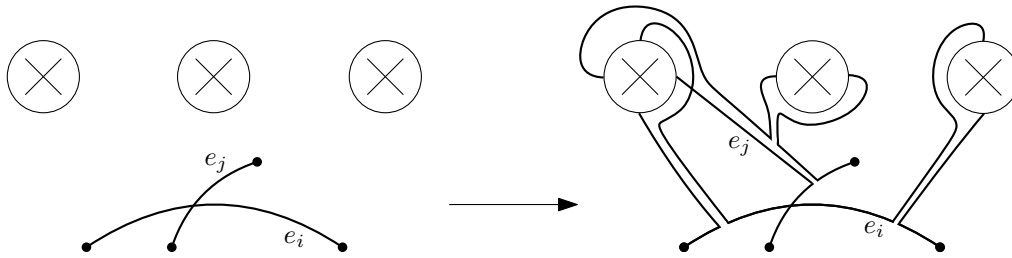
By Lemma 7, there are an invertible matrix  $C$  and a diagonal matrix  $D$  of rank  $r'$  such that  $A' = C^\top DC$ . Since every element of  $\mathbb{F}_2$  is a square of itself, we have  $D = D^\top D$ . Let  $E$  be the  $r' \times n$  matrix obtained from  $D$  by removing all the zero rows from  $D$ . Then  $D = E^\top E$  is a rank factorization of  $D$ , and hence  $A' = C^\top E^\top EC = (EC)^\top EC$  is a rank factorization of  $A'$ . If  $A$  is not alternate, we choose  $B$  as  $EC$ .

If  $A$  is alternate, we obtain  $B$  from  $EC$  by deleting the first column. By the definition of  $A'$ , we have  $B^\top B = A$ . Clearly, we have  $r \leq \text{rank}(B) \leq \text{rank}(EC) = r' = r + 1$ . ◀

Let  $\mathcal{D}$  be a drawing of a graph  $G$  in the plane and let  $E(G) = \{e_1, \dots, e_m\}$ . We say that a symmetric  $m \times m$  matrix  $A = (a_{ij})$  over  $\mathbb{F}_2$  *represents*  $\mathcal{D}$  if for every independent pair of edges  $e_i, e_j$  we have  $a_{ij} = \text{cr}_{\mathcal{D}}(e_i, e_j) \pmod 2$ . In particular, if  $G$  has a vertex of degree at least 2 then a matrix representing  $\mathcal{D}$  is not unique.

► **Proposition 10.** *Let  $\mathcal{D}$  be a drawing of a graph  $G$  in the plane. If a matrix  $A$  represents  $\mathcal{D}$  then  $\text{eg}_0(G) \leq \text{rank}(A)$ . If additionally  $A$  has only zeros on the diagonal then  $\text{g}_0(G) \leq \text{rank}(A)/2$ .*

<sup>1</sup> Over an arbitrary field,  $A$  is an alternate matrix if  $A^\top = -A$  and all the diagonal entries of  $A$  are 0. The diagonal condition is redundant for the fields of characteristic other than 2. Alternate matrices are precisely coordinate matrices of *alternating* bilinear forms.



■ **Figure 4** Pulling the edges  $e_i$  and  $e_j$  with the crosscap vectors  $y_{e_i}^\top = (1, 0, 1)$  and  $y_{e_j}^\top = (1, 1, 0)$ , respectively, over crosscaps.

**Proof.** Let  $r$  be the rank of  $A$ . If  $r = 0$ , then  $\mathcal{D}$  is independently even, and hence  $eg_0(G) = g_0(G) = 0$ . Now assume that  $r > 0$ . Let  $A = B^\top B$  be the factorization from Lemma 9. The matrix  $B$  is an  $h \times m$  matrix of rank  $r$  or  $r + 1$ , and  $r \leq h \leq r + 1$ . Moreover, if  $h = r + 1$ ,  $A$  is an alternate matrix. Write  $B$  as  $(y_{e_1} \ \dots \ y_{e_m})$ . We will do the following. For every  $i \in [m]$ , we interpret  $y_{e_i} \in \mathbb{Z}_2^h$  as a crosscap vector of the edge  $e_i$  of  $G$ . Then we construct a drawing  $\mathcal{D}_0$  of  $G$  in the plane with  $h$  crosscaps in which  $cr_{\mathcal{D}_0}^*(e, f)$  is even for every pair of independent edges  $e, f$ , and such that  $y_{e_i}^{\mathcal{D}_0} = y_{e_i}$  for every  $i \in [m]$ .

Now we describe the construction of  $\mathcal{D}_0$  in more detail. In the complement of  $\mathcal{D}$  in the plane we introduce  $h$  crosscaps. For every  $i \in [m]$  and  $j \in [h]$ , if  $(y_{e_i})_j = 1$ , we pull  $e_i$  over the  $j$ th crosscap (in an arbitrary order). Since every edge is pulled over each crosscap at most once, we can easily avoid creating self-crossings of the edges. See [30, Fig. 1] or Figure 4 for an illustration. Let  $\mathcal{D}_0$  be the resulting drawing in the plane with  $h$  crosscaps. For every  $e_j \in E(G)$ , the parity of  $cr_{\mathcal{D}_0}^*(e_i, e_j)$  differs from the parity of  $cr_{\mathcal{D}}(e_i, e_j)$  if and only if  $y_{e_j}^\top y_{e_i}$  is odd. By the definition of  $A$ , if  $e_i$  and  $e_j$  are independent, the parity of  $cr_{\mathcal{D}}(e_i, e_j)$  is the same as the parity of  $y_{e_j}^\top y_{e_i}$ , and so  $cr_{\mathcal{D}_0}^*(e_i, e_j)$  is even as required.

The drawing  $\mathcal{D}_0$  represents an independently even drawing on  $N_h$ . If  $h = r$ , the first part of the proposition follows. If  $h = r + 1$  then  $A$  is alternate, and Lemma 6 implies that  $r$  is even. The decomposition  $A = B^\top B$  now also implies that  $y_{e_i}^\top y_{e_i} \pmod 2 = 0$  for every  $i \in [m]$ , and hence the drawing  $\mathcal{D}_0$  is orientable. Therefore, by Lemma 4 we have  $g_0(G) \leq \lfloor (r + 1)/2 \rfloor = r/2$ . Since  $eg_0(G) \leq 2g_0(G)$ , we also get  $eg_0(G) \leq r$ . ◀

An almost immediate corollary of Proposition 10 is the following.

► **Corollary 11.** *We have  $eg_0(G) = \min_{A, \mathcal{D}} \text{rank}(A)$ , where we minimize over symmetric matrices  $A$  representing a drawing  $\mathcal{D}$  of  $G$  in the plane, and  $g_0(G) = \min_{A, \mathcal{D}} \text{rank}(A)/2$ , where we minimize over alternate matrices  $A$  representing a drawing  $\mathcal{D}$  of  $G$  in the plane.*

#### 4 Minimum rank of partial symmetric matrices

In this section we prove linear-algebraic results that we use to establish Theorem 1 and Theorem 2. We write  $I_n$  and  $J_n$  for the  $n \times n$  identity matrix and all-one matrix, respectively.

It is a basic fact that the matrix rank is subadditive over an arbitrary field, that is, for any two matrices  $A_1$  and  $A_2$  of the same dimensions we have

$$\text{rank}(A_1 + A_2) \leq \text{rank}(A_1) + \text{rank}(A_2). \tag{2}$$

### 4.1 Tournament matrices

All matrices in this subsection are  $\{0, 1\}$ -matrices and all matrix computations are performed over  $\mathbb{F}_2$ . An  $n \times n$  matrix  $A = (a_{ij})$  is a *tournament matrix* if  $a_{ij} = a_{ji} + 1$  whenever  $i \neq j$ .

The aim of this subsection is to extend de Caen's [10] lower bound on the rank of tournament matrices to certain block matrices. This extension lies at the heart of the proof of Theorem 1.

De Caen [10] proved that every  $n \times n$  tournament matrix  $A$  satisfies

$$\text{rank}(A) \geq \left\lceil \frac{n-1}{2} \right\rceil, \tag{3}$$

which can be seen as follows. We have  $A + A^\top = J_n + I_n$ , and (2) implies that  $\text{rank}(I_n + J_n) \geq n - 1$ . Using (2) again, we get  $n - 1 \leq \text{rank}(A) + \text{rank}(A^\top) = 2 \cdot \text{rank}(A)$ .

► **Lemma 12.** *Let  $m, n \geq 2$ . Let  $A = (A_{ij})$  be an  $m \times m$  block matrix, where each block  $A_{ij}$  is an  $n \times n$  matrix. Let  $B$  be an  $n \times n$  tournament matrix. Assume that  $A$  is symmetric and that for each off-diagonal block  $A_{ij}$ ,  $i \neq j$ , one of the matrices  $A_{ij} + B$  or  $A_{ij} + B + J_n$  is a diagonal  $n \times n$  matrix. Then  $\text{rank}(A) \geq \left\lceil \frac{(m-1)(n-1)}{2} \right\rceil - (m - 2)$ .*

### 4.2 Block symmetric matrices

In this section, we prove minimum rank formulas for certain partial block symmetric matrices that play an important role in the proof of Theorem 2. The study of the rank of partial block matrices was initiated by Cohen et al. [7], Davis [9], and Woerdeman [36]. We adapt previous results to the setting of symmetric matrices.

Let  $A(X) = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & X \end{pmatrix}$  be a block matrix over an arbitrary field in which the block  $X$  is treated as a variable. Woerdeman [36] and Davis [9] proved that

$$\min_X \text{rank}(A(X)) = \text{rank} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} \\ A_{12} \end{pmatrix} - \text{rank}(A_{11}). \tag{4}$$

The following lemma shows that for symmetric  $A(X)$ , the minimum in (4) is achieved for a symmetric matrix  $X$ .

► **Lemma 13.** *Let  $A_{21} = A_{12}^\top$  and let  $A_{11}$  be symmetric. Then*

$$\min_X \text{rank}(A(X)) = 2 \cdot \text{rank} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} - \text{rank}(A_{11}),$$

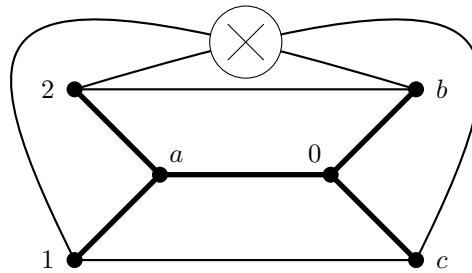
where we minimize over symmetric  $X$ .

Let  $A(X_2, X_3) = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & X_2 & A_{23} \\ A_{31} & A_{32} & X_3 \end{pmatrix}$  be a block matrix over an arbitrary field in which

the blocks  $X_2$  and  $X_3$  are treated as variables. For matrices over fields of characteristic different from 2, Cohen et al. [7], see also [34], proved that

$$\begin{aligned} \min_{X_2, X_3} \text{rank}(A(X_2, X_3)) &= \text{rank} \begin{pmatrix} A_{11} & A_{12} & A_{13} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} \\ A_{21} \\ A_{31} \end{pmatrix} \\ &+ \min \left\{ \text{rank} \begin{pmatrix} A_{11} & A_{12} \\ A_{31} & A_{32} \end{pmatrix} - \left( \text{rank} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} \\ A_{31} \end{pmatrix} \right), \right. \\ &\quad \left. \text{rank} \begin{pmatrix} A_{11} & A_{13} \\ A_{21} & A_{23} \end{pmatrix} - \left( \text{rank} \begin{pmatrix} A_{11} & A_{13} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} \right) \right\}, \end{aligned} \tag{5}$$





■ **Figure 5** An embedding of  $K_{3,3}$  in the plane with a single crosscap. The edges of the spanning tree  $T$  are thickened.

In the following lemma, we prove an upper bound on  $\min_{X_2, X_3} \text{rank}(A(X_2, X_3))$ , which is equal to the right-hand side of (5), if we restrict ourselves to symmetric matrices  $A(X_2, X_3)$ . The lemma is valid for the symmetric matrices over an arbitrary field.

► **Lemma 14.** *Let  $A_{21} = A_{21}^\top, A_{31} = A_{13}^\top, A_{32} = A_{23}^\top$ , and let  $A_{11}$  be symmetric. Then*

$$\begin{aligned} \min_{X_2, X_3} \text{rank}(A(X_2, X_3)) &\leq 2 \cdot \text{rank} \begin{pmatrix} A_{11} & A_{12} & A_{13} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} & A_{12} \\ A_{31} & A_{32} \end{pmatrix} \\ &\quad - (\text{rank} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{11} & A_{13} \end{pmatrix}) \end{aligned}$$

where we minimize over symmetric matrices  $X_2$  and  $X_3$ .

## 5 Estimating the $\mathbb{Z}_2$ -genus and the Euler $\mathbb{Z}_2$ -genus of $K_{m,n}$

We prove Theorem 1, whose proof is based on our previous result [15, Lemma 17], which we present next. All matrices and vectors in this subsection are  $\{0, 1\}$ -matrices and all matrix and vector computations are performed over  $\mathbb{F}_2$ .

In 1976, Kleitman [18] proved that every drawing of  $K_{3,3}$  in the plane contains an odd number of unordered pairs of independent edges crossings an odd number of times. Let  $\{a, b, c\}$  and  $\{0, 1, 2\}$  be the two maximal independent sets in  $K_{3,3}$  and let  $T$  be the spanning tree of  $K_{3,3}$  containing all the edges incident to  $a$  and  $0$ . Let  $\mathcal{D}$  be a drawing of  $K_{3,3}$  in the plane with finitely many crosscaps in which  $\text{cr}_{\mathcal{D}}^*(e, f)$  is even for every pair of independent edges  $e, f$ , and  $y_e = 0$  for every  $e \in E(T)$ ; see Figure 5 for an illustration. The result of Kleitman implies the following lemma, restating [15, Lemma 17].

► **Lemma 15.** *In the drawing  $\mathcal{D}$ ,  $y_{b1}^\top y_{c2} + y_{c1}^\top y_{b2} = 1$ .*

**Proof.** Let  $\mathcal{D}'$  be the planarization of  $\mathcal{D}$ . By (1),  $\text{cr}_{\mathcal{D}'}(e, f) = y_e^\top y_f$  for every pair of independent edges  $e$  and  $f$  in  $K_{3,3}$ , since  $\mathcal{D}$  is independently even. Using Kleitman’s result,  $1 = \sum_{e,f} \text{cr}_{\mathcal{D}'}(e, f) = \sum_{e,f} y_e^\top y_f$ , where we sum over unordered independent pairs. Hence,  $\sum_{e,f} y_e^\top y_f = y_{b1}^\top y_{c2} + y_{c1}^\top y_{b2}$  concludes the proof. ◀

**Proof of Theorem 1.** We denote the vertices of  $K_{m,n}$  in one part by  $u_0, \dots, u_{m-1}$  and in the other part by  $v_0, \dots, v_{n-1}$ .

Let  $\mathcal{D}$  be the combinatorial representation of an independently even drawing of  $K_{m,n}$  on a surface  $S$  in the plane with finitely many crosscaps (see Section 3.1). Let  $y_e = y_e^{\mathcal{D}}$  be the crosscap vector of  $e \in E(K_{m,n})$  associated with  $\mathcal{D}$ . For  $i_1, i_2 \in \{1, \dots, m-1\} = [m-1]$ , let  $A_{i_1 i_2} = (a_{j_1 j_2})$  be the  $(n-1) \times (n-1)$  matrix with entries  $a_{j_1 j_2} = y_{i_1 j_1}^\top y_{i_2 j_2}$ . Let  $A = (A_{i_1 i_2})$  be the  $(m-1) \times (m-1)$  block matrix composed of the previously defined  $A_{i_1 i_2}$ ’s. By Lemma 3,



we assume that  $y_e = 0$  for  $e \in E' = \{u_0v_0, \dots, u_0v_{n-1}, v_0u_1, \dots, v_0u_{m-1}\}$ . Hence, we can let  $A' = \begin{pmatrix} A & 0 \\ 0 & 0 \end{pmatrix}$  be a matrix representing  $\mathcal{D}$ , where the rows and columns of the three all-zero blocks correspond to the edges in  $E'$ . For every  $i_1, i_2, j_1$  and  $j_2$ , where  $i_1 \neq i_2$  and  $j_1 \neq j_2$  we then apply Lemma 15 to the drawing of  $K_{3,3}$  induced by the vertices  $u_0, v_0, u_{i_1}, u_{i_2}, v_{j_1}, v_{j_2}$  in  $\mathcal{D}$  and obtain that  $a_{j_1j_2} + a_{j_2j_1} = 1$ . In other words,  $A_{i_1i_2}$  is a tournament matrix. We show that either  $A_{i_1i_2} = B + D_{i_1i_2}$  or  $A_{i_1i_2} = B + J_{n-1} + D_{i_1i_2}$ , where  $B$  is a fixed tournament matrix and  $D_{i_1i_2}$  is a diagonal matrix.

If the previous claim holds then Lemma 12 applies to  $A$ . Thus,  $\text{rank}(A') = \text{rank}(A) \geq \left\lceil \frac{(m-2)(n-2)}{2} \right\rceil - (m-3)$ . By Corollary 11 or just by observing that  $\text{rank}(A')$  is upper bounded by the dimension of the space generated by the crosscap vectors associated with  $\mathcal{D}$ ,  $\text{eg}_0(K_{m,n}) \geq \left\lceil \frac{(m-2)(n-2)}{2} \right\rceil - (m-3)$  as desired. Similarly,  $2 \cdot \text{g}_0(K_{m,n}) \geq \left\lceil \frac{(m-2)(n-2)}{2} \right\rceil - (m-3)$  and the claimed lower bound for  $\text{g}_0(K_{m,n})$  follows as well.

It remains to prove the claim. To this end we apply the argument that was used to prove [15, Lemma 17]. In the drawing of  $K_{3,3}$  induced by the vertices  $u_0, v_0, u_{i_1}, u_{i_2}, v_{j_1}, v_{j_2}$  in  $\mathcal{D}$  we locally deform  $\mathcal{D}$  in a close neighborhood of  $u_0$ , so that the edges  $u_0v_0, u_0v_{j_1}$  and  $u_0v_{j_2}$  cross one another an even number of times, while keeping  $\mathcal{D}$  independently even. It is easy to see that this is indeed possible. Similarly, we adjust the drawing in a close neighborhood of  $v_0$ , so that the edges  $v_0u_0, v_0u_{i_1}$  and  $v_0u_{i_2}$  cross one another an even number of times. Let  $\mathcal{D}'$  be the resulting modification of  $\mathcal{D}$ .

We will prove below that

- (\*) In the block  $A_{i_1, i_2}$ , for  $j_1 \neq j_2$ , the value  $a_{j_1, j_2} = 1$  if and only if up to the choice of orientation the edges  $u_0v_0, u_0v_{j_1}, u_0v_{j_2}$  and  $v_0u_0, v_0u_{i_1}, v_0u_{i_2}$  appear in the rotation at  $u_0$  and  $v_0$ , respectively, in this order clockwise.

Hence, suppose that (\*) holds and that  $v_0u_0, v_0u_{i_1}, v_0u_{i_2}$  appear in the rotation at  $v_0$  in  $\mathcal{D}'$  in this order clockwise. For  $i'_1, i'_2 \in [m-1]$ ,  $i'_1 \neq i'_2$ , we adjust the drawing in a close neighborhood of  $v_0$ , so that the edges  $v_0u_0, v_0u_{i'_1}$  and  $v_0u_{i'_2}$  cross one another an even number of times. Let  $\mathcal{D}''$  be the resulting drawing. By (\*),  $A_{i'_1i'_2} = A_{i_1i_2} + D_{i'_1i'_2}$ , where  $D_{i_1i_2}$  is a diagonal matrix, if  $v_0u_0, v_0u_{i'_1}, v_0u_{i'_2}$  in  $\mathcal{D}''$  appear in the rotation at  $v_0$  in this order clockwise; and  $A_{i'_1i'_2} = A_{i_1i_2} + D_{i'_1i'_2} + J_{n-1}$ , if  $v_0u_0, v_0u_{i'_1}, v_0u_{i'_2}$  appear in the rotation at  $v_0$  in  $\mathcal{D}''$  in this order counterclockwise. It remains to prove (\*).

Let  $\gamma_{i,j}$  be the closed curve representing the cycle traversing vertices  $u_0, v_0, u_i$  and  $v_j$  in  $\mathcal{D}$ . The condition that characterizes when  $a_{j_1j_2} = 1$ , for  $j_1 \neq j_2$ , follows by considering a slightly perturbed drawing of  $\gamma_{i_1, j_1}$  and  $\gamma_{i_2, j_2}$ , in which all their intersections become proper edge crossings. Note that  $a_{j_1j_2} = 1$  if and only if  $u_{i_1}v_{j_1}$  and  $u_{i_2}v_{j_2}$  have an odd number of intersections at crosscaps. Furthermore,  $\gamma_{i_1, j_1}$  and  $\gamma_{i_2, j_2}$  must have an even number of intersections in total. Therefore as  $\mathcal{D}$  is an independently even drawing,  $a_{j_1j_2} = 1$  if and only if in  $\mathcal{D}'$  the edge  $u_0v_0$  is a transversal intersection of  $\gamma_{i_1, j_1}$  and  $\gamma_{i_2, j_2}$ ; in other words, up to the choice of orientation  $u_0v_0, u_0v_{j_1}, u_0v_{j_2}$  and  $v_0u_0, v_0u_{i_1}, v_0u_{i_2}$  appear in the rotation at  $u_0$  in this order clockwise. ◀

## 6 Amalgamations

All matrices and vectors in this subsection are  $\{0, 1\}$ -matrices and all matrix and vector computations are performed over  $\mathbb{F}_2$ .

### 6.1 1-amalgamations

In order to ease up the readability, as a warm-up we first reprove the result of Schaefer and Štefankovič for the Euler genus. The proof of our result for 2-amalgamations follows the same blueprint, but the argument gets slightly more technical.

► **Theorem 16** ([30]). *Let  $G_1$  and  $G_2$  be graphs. Let  $G = \Pi_v(G_1, G_2)$ . Then  $\text{eg}_0(G_1) + \text{eg}_0(G_2) = \text{eg}_0(G)$ .*

**Proof.** The Euler  $\mathbb{Z}_2$ -genus of a graph is the sum of the Euler  $\mathbb{Z}_2$ -genera of its connected components [30, Lemma 7]. Thus, we assume that both  $G_1$  and  $G_2$  are connected.

We start the argument similarly as in [30] by choosing an appropriate spanning tree  $T$  in  $G$  and fixing an independently even drawing of  $G$  on  $N_g$ , in which each edge in  $E(T)$  passes an even number of times through each crosscap. Nevertheless, the rest of the proof differs considerably, one of the key differences being the use of Proposition 10 rather than Lemma 5 to bound the Euler  $\mathbb{Z}_2$ -genus of involved graphs.

The following claim is rather easy to prove.

▷ **Claim 17.** We have

$$\text{eg}_0(G) \leq \text{eg}_0(G_1) + \text{eg}_0(G_2). \tag{6}$$

It remains to prove the opposite inequality. We first choose a spanning tree  $T$  of  $G$  with the following property. Recall that  $v$  is a fixed cut vertex. For every  $e \in E(G) \setminus E(T)$  it holds that if  $v \notin e$  then the unique cycle in  $T \cup e$  does not pass through  $v$ . The desired spanning tree  $T$  is obtained as the exploration tree of a Depth-First-Search in  $G$  starting at  $v$ . We consider an independently even drawing of  $G$  on a surface  $S$  witnessing its Euler genus. By Lemma 3, we obtain a drawing  $\mathcal{D}$  of  $G$  in the plane with finitely many crosscaps in which  $\text{ct}_{\mathcal{D}}^*(e, f)$  is even for every pair of independent edges  $e, f$ , and every edge of  $T$  passes through each crosscap an even number of times. In the following we will write  $y_e$  for  $y_e^{\mathcal{D}}$ .

First, a few words on the strategy of the rest of the proof. Let  $B' = \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix}$  be a matrix representing the planarization of  $\mathcal{D}$ , where the rows and columns of the three all-zero blocks correspond to the edges in  $T$ . For a pair of edges  $e$  and  $f$  in  $G$  this parity is given by  $y_e^{\top} y_f$ . By introducing an appropriate block structure on  $B$ , and using Lemma 13 we show that the rank of  $B$  can be lower bounded by  $\text{eg}_0(G_1) + \text{eg}_0(G_2)$ . This will conclude the proof since the rank of  $B$  is easily upper bounded by  $\text{eg}_0(G)$ .

Let  $E_1$  and  $E_2$  be the set of edges in  $E(G_1) \setminus E(T)$  and  $E(G_2) \setminus E(T)$ , respectively, that are not incident to  $v$ . Let  $F_1$  and  $F_2$  be the set of edges in  $E(G_1) \setminus E(T)$  and  $E(G_2) \setminus E(T)$ , respectively, that are incident to  $v$ .

Let  $\alpha, \beta \in \{E_1, E_2, F_1, F_2\}$ . Let  $\alpha = \{e_1, \dots, e_{|\alpha|}\}$ . Let  $\beta = \{e'_1, \dots, e'_{|\beta|}\}$ . Let  $A_{\alpha, \beta} = (a_{ij})$  be the  $|\alpha| \times |\beta|$  matrix over  $\mathbb{Z}_2$  such that  $a_{ij} = y_{e_i}^{\top} y_{e'_j}$ . Let  $B = (B_{ij})$  be a  $4 \times 4$  block matrix such that  $B_{ij} = A_{\alpha_i, \alpha_j}$ , where  $\alpha_1 = E_1, \alpha_2 = F_1, \alpha_3 = F_2$  and  $\alpha_4 = E_2$ . Clearly,  $B'$  represents the planarization of  $\mathcal{D}$ .

In what follows we collect some properties of  $B$  and its submatrices, whose combination establishes the result. The rank of  $B'$ , and therefore also  $B$ , is at most the dimension of the space generated by the crosscap vectors of  $\mathcal{D}$ . The latter is at most  $\text{eg}_0(G)$  since crosscap vectors have  $\text{eg}_0(G)$  or  $\text{eg}_0(G) + 1$  coordinates depending on whether the original drawing of  $G$  is on  $N_g$  or  $M_g$ , but in the latter we lose one dimension since every crosscap vector has an even number of ones. Hence, we have

$$\text{eg}_0(G) \geq \text{rank}(B). \tag{7}$$

If we arbitrarily change blocks  $A_{F_1, F_1}$  and  $A_{F_2, F_2}$  of  $B$ ,  $B'$  will still represent the planarization of  $\mathcal{D}$ . Let  $B_1(X) = \begin{pmatrix} A_{E_1, E_1} & A_{E_1, F_1} \\ A_{F_1, E_1} & X \end{pmatrix}$  and  $B_2(X) = \begin{pmatrix} X & A_{F_2, E_2} \\ A_{E_2, F_2} & A_{E_2, E_2} \end{pmatrix}$ . Then by Proposition 10,

$$\text{eg}_0(G_i) \leq \min_X \begin{pmatrix} \text{rank}(B_i(X)) & 0 \\ 0 & 0 \end{pmatrix} = \min_X \text{rank}(B_i(X)), \quad (8)$$

where we minimize over symmetric matrices  $X$ . By Lemma 13,

$$\min_X \text{rank}(B_i(X)) = 2 \cdot \text{rank} \begin{pmatrix} A_{E_i, E_i} & A_{E_i, F_i} \\ A_{F_i, E_i} & X \end{pmatrix} - \text{rank}(A_{E_i, E_i}). \quad (9)$$

The last ingredient in the proof is the following claim which holds due to the careful choice of the spanning tree  $T$ .

▷ **Claim 18.** We have

$$\begin{aligned} & 2 \cdot (\text{rank} \begin{pmatrix} A_{E_1, E_1} & A_{E_1, F_1} \\ A_{F_1, E_1} & X \end{pmatrix} + \text{rank} \begin{pmatrix} A_{E_2, E_2} & A_{E_2, F_2} \\ A_{F_2, E_2} & X \end{pmatrix}) - (\text{rank}(A_{E_1, E_1}) + \text{rank}(A_{E_2, E_2})) \\ & \leq \text{rank}(B). \end{aligned}$$

We are done by the following chain of inequalities.

$$\begin{aligned} \text{eg}_0(G) & \stackrel{(6)}{\leq} \text{eg}_0(G_1) + \text{eg}_0(G_2) \stackrel{(8)}{\leq} \min_X \text{rank}(B_1(X)) + \min_X \text{rank}(B_2(X)) \\ & \stackrel{(9)}{=} 2 \cdot (\text{rank} \begin{pmatrix} A_{E_1, E_1} & A_{E_1, F_1} \\ A_{F_1, E_1} & X \end{pmatrix} + \text{rank} \begin{pmatrix} A_{E_2, E_2} & A_{E_2, F_2} \\ A_{F_2, E_2} & X \end{pmatrix}) \\ & \quad - (\text{rank}(A_{E_1, E_1}) + \text{rank}(A_{E_2, E_2})) \\ & \leq \text{rank}(B) \stackrel{(7)}{\leq} \text{eg}_0(G) \quad \blacktriangleleft \end{aligned}$$

## 6.2 2-amalgamations

**Proof of Theorem 2.** We will prove the parts a) and b) in parallel. We assume that  $G - u - v$  has precisely 2 connected components and that  $uv \notin E(G)$  (the general case is treated in the full version). By the block additivity result [30], we assume that none of  $u$  and  $v$  is a cut vertex of  $G$ , and by the additivity over connected components [30, Lemma 7] that  $G$  is connected. We follow the line of thought analogous to the proof of Theorem 16.

It is easy to prove the following claim.

▷ **Claim 19.**

$$g_0(G) \leq g_0(G_1) + g_0(G_2) + 1 \quad \text{and} \quad \text{eg}_0(G) \leq \text{eg}_0(G_1) + \text{eg}_0(G_2) + 2 \quad (10)$$

It remains to prove the opposite inequalities of a) and b). We choose an appropriate spanning tree  $T$  of  $G$  and fix an independently even drawing of  $G$  on  $N_g$ , in which each edge of  $T$  passes an even number of times through each crosscap. To this end we first choose a spanning tree  $T'$  of  $G - v$  with the following property. For every  $e \in E(G - v) \setminus E(T')$ , if  $u \notin e$  then the unique cycle in  $T' \cup e$  does not pass through  $u$ . The desired spanning  $T'$  is obtained as the exploration tree of a Depth-First-Search in  $G - v$  starting at  $u$ . Let  $u_i$  be an arbitrary vertex such that  $vu_i \in E(G_i)$ , for  $i = 1, 2$ . We obtain  $T$  as  $T' \cup vu_1$ .

We consider an independently even drawing of  $G$  on a surface  $S$  witnessing its genus (respectively, Euler genus). By Lemma 3, we obtain a drawing  $\mathcal{D}$  in the plane with finitely many crosscaps in which  $\text{cr}_{\mathcal{D}}^*(e, f)$  is even for every pair of independent edges  $e, f$ , and every edge of  $T$  passes through each crosscap an even number of times. In the following we will write  $y_e$  for  $y_e^{\mathcal{D}}$ .

First, a few words on the strategy of the rest of the proof. Let  $B' = \begin{pmatrix} B & 0 \\ 0 & 0 \end{pmatrix}$  be a matrix representing the planarization of  $\mathcal{D}$ , where the rows and columns of the three all-zero blocks correspond to the edges in  $T$ . For a pair of edges  $e$  and  $f$  in  $G$  this parity is given by  $y_e^\top y_f$ . By introducing an appropriate block structure on  $B$ , and using Lemma 14 we show that the rank of  $B$  can be lower bounded by  $g_0(G_1) + g_0(G_2) - 7/2$  (respectively,  $eg_0(G_1) + eg_0(G_2) - 3$ ). This will conclude the proof in this case, since the rank of  $B$  is easily upper bounded by  $2 \cdot g_0(G)$  (respectively,  $eg_0(G)$ ).

Let  $E_1$  and  $E_2$  be the set of edges in  $E(G_1) \setminus E(T)$  and  $E(G_2) \setminus E(T)$ , respectively, that are incident neither to  $v$  nor to  $u$ . Let  $F_1$  and  $F_2$  be the set of edges in  $E(G_1) \setminus E(T)$  and  $E(G_2) \setminus E(T)$ , respectively, that are incident to  $u$ . Let  $H_1$  and  $H_2$  be the set of edges in  $E(G_1) \setminus E(T)$  and  $E(G_2) \setminus E(T)$ , respectively, that are incident to  $v$ . Thus, we have that  $E(T), E_1, E_2, F_1, F_2, H_1$  and  $H_2$  form a partition of  $E(G)$ .

Let  $\alpha, \beta \in \{E_1, E_2, F_1, F_2, H_1, H_2\}$ . Let  $\alpha = \{e_1, \dots, e_{|\alpha|}\}$ . Let  $\beta = \{e'_1, \dots, e'_{|\beta|}\}$ . Let  $A_{\alpha, \beta} = (a_{ij})$  be the  $|\alpha| \times |\beta|$  matrix over  $\mathbb{Z}_2$  such that  $a_{ij} = y_{e_i}^\top y_{e'_j}$ . Let  $B = (B_{ij})$  be a 6 by 6 block matrix such that  $B_{ij} = A_{\alpha_i, \alpha_j}$ , where  $\alpha_1 = E_1, \alpha_2 = F_1, \alpha_3 = H_1, \alpha_4 = E_2, \alpha_5 = F_2$  and  $\alpha_6 = H_2$ . Clearly,  $B'$  represents the planarization of  $\mathcal{D}$ .

In what follows we collect some properties of  $B$  and its submatrices, whose combination establishes the result. Since the rank of  $B'$ , and therefore also  $B$ , is at most the dimension of the space generated by the crosscap vectors of  $\mathcal{D}$ , we have the following

$$2 \cdot g_0(G) \geq \text{rank}(B) \quad (\text{respectively, } eg_0(G) \geq \text{rank}(B)). \tag{11}$$

Let  $B_1(X_2, X_3) = \begin{pmatrix} A_{E_1, E_1} & A_{E_1, F_1} & A_{E_1, H_1} \\ A_{F_1, E_1} & X_2 & A_{F_1, H_1} \\ A_{H_1, E_1} & A_{H_1, F_1} & X_3 \end{pmatrix}$ , and  
 let  $B_2(X_1, X_2) = \begin{pmatrix} X_1 & A_{H_2, F_2} & A_{H_2, E_2} \\ A_{F_2, H_2} & X_2 & A_{F_2, E_2} \\ A_{E_2, H_2} & A_{E_2, F_2} & A_{E_2, E_2} \end{pmatrix}$ .

Since changing blocks  $A_{F_i, F_i}$  and  $A_{H_i, H_i}$  in  $B$ , for  $i = 1, 2$ , except for the diagonal, does not affect the property that  $B'$  represents the planarization of  $\mathcal{D}$ , by Proposition 10,

$$2 \cdot g_0(G_i) \leq \min_{X_2, X_3} \text{rank}(B_i(X_2, X_3)) + 2 \quad (\text{respectively, } eg_0(G_i) \leq \min_{X_2, X_3} \text{rank}(B_i(X_2, X_3))), \tag{12}$$

where we minimize over symmetric matrices. We add 2 on the right hand side in the first inequality due to the orientability. In particular, it can happen that  $X_2$  or  $X_3$  minimizing the rank has a 1-entry on the diagonal. If this is the case, in the corresponding independently even drawing, as constructed in the proof of Proposition 10, there exists an edge  $e$  incident to  $u$  or  $v$  such that  $y_e^\top y_e = 1$ . In order to make  $y_e^\top y_e = 0$ , we introduce a crosscap, and pull the edge  $e$  over it. The introduced crosscap can be shared by the edges incident to  $v$  and by the edges incident to  $u$ . Therefore adding 2 crosscaps is sufficient. By Lemma 14,

$$\begin{aligned} \min_{X_2, X_3} \text{rank}(B_i(X_2, X_3)) &\leq 2 \cdot \text{rank} \begin{pmatrix} A_{E_i, E_i} & A_{E_i, F_i} & A_{E_i, H_i} \\ A_{H_i, E_i} & A_{H_i, F_i} \end{pmatrix} \\ &\quad - (\text{rank} \begin{pmatrix} A_{E_i, E_i} & A_{E_i, F_i} \end{pmatrix} + \text{rank} \begin{pmatrix} A_{E_i, E_i} & A_{E_i, H_i} \end{pmatrix}) \end{aligned} \tag{13}$$

The inequality (13) implies the last ingredient in the proof which is stated next. The claim holds due to the careful choice of the spanning tree  $T$ .

▷ Claim 20.  $\min_{X_2, X_3} \text{rank}(B_1(X_2, X_3)) + \min_{X_1, X_2} \text{rank}(B_2(X_1, X_2)) \leq \text{rank}(B) + 3$ , where we minimize over symmetric matrices.

We are done by the following two chains of (in)equalities.

$$\begin{aligned} -2 + 2 \cdot g_0(G) &\stackrel{(10)}{\leq} 2 \cdot g_0(G_1) + 2 \cdot g_0(G_2) \\ &\stackrel{(12)}{\leq} \min_{X_2, X_3} \text{rank}(B_1(X_2, X_3)) + \min_{X_1, X_2} \text{rank}(B_2(X_1, X_2)) + 4 \\ &\leq \text{rank}(B) + 7 \\ &\stackrel{(11)}{\leq} 2 \cdot g_0(G) + 7, \end{aligned}$$

$$\begin{aligned} -2 + \text{eg}_0(G) &\stackrel{(10)}{\leq} \text{eg}_0(G_1) + \text{eg}_0(G_2) \\ &\stackrel{(12)}{\leq} \min_{X_2, X_3} \text{rank}(B_1(X_2, X_3)) + \min_{X_1, X_2} \text{rank}(B_2(X_1, X_2)) \\ &\leq \text{rank}(B) + 3 \\ &\stackrel{(11)}{\leq} \text{eg}_0(G) + 3. \end{aligned} \quad \blacktriangleleft$$

## 7 Conclusion

Theorem 1 does not determine the (Euler)  $\mathbb{Z}_2$ -genus of  $K_{m,n}$  precisely for  $m \geq 4$ , and we find the problem of computing the precise values interesting already for  $m = 4$ . We also leave as an open problem whether in Theorem 2, the dependence of the upper bounds on  $k$  can be removed. Let  $G$  be a  $k$ -amalgamation of  $G_1$  and  $G_2$  for some  $k \geq 3$ . On the one hand, the result of Miller [20] and Richter [25] was extended by Archdeacon [2] to  $k$ -amalgamations, for  $k \geq 3$ , with the error term  $\text{eg}(G_1) + \text{eg}(G_2) - \text{eg}(G)$  being at most quadratic in  $k$ . On the other hand, in a follow-up paper [3] Archdeacon showed that for  $k \geq 3$ , the genus of a graph is not additive over  $k$ -amalgamations, in a very strong sense. In particular, the value of  $g(G_1) + g(G_2) - g(G)$  can be arbitrarily large even for  $k = 3$ . We wonder if the  $\mathbb{Z}_2$ -genus and the Euler  $\mathbb{Z}_2$ -genus behave in a similar way.

---

## References

- 1 A. Adrian Albert. Symmetric and alternate matrices in an arbitrary field, I. *Trans. Amer. Math. Soc.*, 43(3):386–436, 1938. doi:10.2307/1990068.
- 2 Dan Archdeacon. The nonorientable genus is additive. *J. Graph Theory*, 10(3):363–383, 1986. doi:10.1002/jgt.3190100313.
- 3 Dan Archdeacon. The orientable genus is nonadditive. *J. Graph Theory*, 10(3):385–401, 1986. doi:10.1002/jgt.3190100314.
- 4 Joseph Battle, Frank Harary, Yukihiko Kodama, and J. W. T. Youngs. Additivity of the genus of a graph. *Bull. Amer. Math. Soc.*, 68:565–568, 1962. doi:10.1090/S0002-9904-1962-10847-7.
- 5 André Bouchet. Orientable and nonorientable genus of the complete bipartite graph. *J. Combin. Theory Ser. B*, 24(1):24–33, 1978. doi:10.1016/0095-8956(78)90073-4.
- 6 G. Cairns and Y. Nikolayevsky. Bounds for generalized thrackles. *Discrete Comput. Geom.*, 23(2):191–206, 2000. doi:10.1007/PL00009495.
- 7 Nir Cohen, Charles R. Johnson, Leiba Rodman, and Hugo J. Woerdeman. Ranks of completions of partial matrices. In *The Gohberg anniversary collection, Vol. I (Calgary, AB, 1988)*, volume 40 of *Oper. Theory Adv. Appl.*, pages 165–185. Birkhäuser, Basel, 1989.

- 8 Éric Colin de Verdière, Vojtěch Kaluža, Pavel Paták, Zuzana Patáková, and Martin Tancer. A direct proof of the strong Hanani–Tutte theorem on the projective plane. *J. Graph Algorithms Appl.*, 21(5):939–981, 2017. doi:10.7155/jgaa.00445.
- 9 Chandler Davis. Completing a matrix so as to minimize the rank. In *Topics in operator theory and interpolation*, volume 29 of *Oper. Theory Adv. Appl.*, pages 87–95. Birkhäuser, Basel, 1988.
- 10 D. de Caen. The ranks of tournament matrices. *Amer. Math. Monthly*, 98(9):829–831, 1991. doi:10.2307/2324270.
- 11 R. W. Decker, H. H. Glover, and J. P. Huneke. The genus of the 2-amalgamations of graphs. *J. Graph Theory*, 5(1):95–102, 1981. doi:10.1002/jgt.3190050107.
- 12 R. W. Decker, H. H. Glover, and J. P. Huneke. Computing the genus of the 2-amalgamations of graphs. *Combinatorica*, 5(4):271–282, 1985. doi:10.1007/BF02579241.
- 13 R. Fulek and J. Kynčl. Counterexample to an extension of the Hanani–Tutte theorem on the surface of genus 4. Accepted to *Combinatorica*, 2017. arXiv:1709.00508.
- 14 Radoslav Fulek and Jan Kynčl. Hanani–Tutte for approximating maps of graphs. In *34th International Symposium on Computational Geometry*, volume 99 of *LIPICs. Leibniz Int. Proc. Inform.*, pages 39:1–39:15. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. doi:10.4230/LIPICs.SocG.2018.39.
- 15 Radoslav Fulek and Jan Kynčl. The  $\mathbb{Z}_2$ -genus of Kuratowski minors. In *34th International Symposium on Computational Geometry*, volume 99 of *LIPICs. Leibniz Int. Proc. Inform.*, pages 40:1–40:14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018. doi:10.4230/LIPICs.SocG.2018.40.
- 16 Jonathan L. Gross and Thomas W. Tucker. *Topological graph theory*. Dover Publications, Inc., Mineola, NY, 2001. Reprint of the 1987 original [Wiley, New York; MR0898434 (88h:05034)] with a new preface and supplementary bibliography.
- 17 Haim Hanani. Über wesentlich unplättbare Kurven im drei-dimensionalen Raume. *Fundamenta Mathematicae*, 23:135–142, 1934.
- 18 D. J. Kleitman. A note on the parity of the number of crossings of a graph. *J. Combinatorial Theory Ser. B*, 21(1):88–89, 1976. doi:10.1016/0095-8956(76)90032-0.
- 19 Jessie MacWilliams. Orthogonal matrices over finite fields. *Amer. Math. Monthly*, 76:152–164, 1969. doi:10.2307/2317262.
- 20 Gary L. Miller. An additivity theorem for the genus of a graph. *J. Combin. Theory Ser. B*, 43(1):25–47, 1987. doi:10.1016/0095-8956(87)90028-1.
- 21 Bojan Mohar. An obstruction to embedding graphs in surfaces. *Discrete Math.*, 78(1-2):135–142, 1989. doi:10.1016/0012-365X(89)90170-2.
- 22 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2001.
- 23 János Pach and Géza Tóth. Which crossing number is it anyway? *J. Combin. Theory Ser. B*, 80(2):225–246, 2000. doi:10.1006/jctb.2000.1978.
- 24 Michael J. Pelsmayer, Marcus Schaefer, and Despina Stasi. Strong Hanani-Tutte on the projective plane. *SIAM J. Discrete Math.*, 23(3):1317–1323, 2009. doi:10.1137/08072485X.
- 25 R. Bruce Richter. On the Euler genus of a 2-connected graph. *J. Combin. Theory Ser. B*, 43(1):60–69, 1987. doi:10.1016/0095-8956(87)90030-X.
- 26 Gerhard Ringel. Das Geschlecht des vollständigen paaren Graphen. *Abh. Math. Sem. Univ. Hamburg*, 28:139–150, 1965. doi:10.1007/BF02993245.
- 27 Gerhard Ringel. Der vollständige paare Graph auf nichtorientierbaren Flächen. *J. Reine Angew. Math.*, 220:88–93, 1965. doi:10.1515/crll.1965.220.88.
- 28 Marcus Schaefer. Hanani-Tutte and related results. In *Geometry—intuitive, discrete, and convex*, volume 24 of *Bolyai Soc. Math. Stud.*, pages 259–299. János Bolyai Math. Soc., Budapest, 2013. doi:10.1007/978-3-642-41498-5\_10.
- 29 Marcus Schaefer. Toward a theory of planarity: Hanani-Tutte and planarity variants. *J. Graph Algorithms Appl.*, 17(4):367–440, 2013. doi:10.7155/jgaa.00298.

- 30 Marcus Schaefer and Daniel Štefankovič. Block additivity of  $\mathbb{Z}_2$ -embeddings. In *Graph drawing*, volume 8242 of *Lecture Notes in Comput. Sci.*, pages 185–195. Springer, Cham, 2013. doi:10.1007/978-3-319-03841-4\_17.
- 31 Mikhail Skopenkov. On approximability by embeddings of cycles in the plane. *Topology Appl.*, 134(1):1–22, 2003. doi:10.1016/S0166-8641(03)00069-5.
- 32 Saul Stahl. Permutation-partition pairs: a combinatorial generalization of graph embeddings. *Trans. Amer. Math. Soc.*, 259(1):129–145, 1980. doi:10.2307/1998149.
- 33 Saul Stahl and Lowell W. Beineke. Blocks and the nonorientable genus of graphs. *J. Graph Theory*, 1(1):75–78, 1977. doi:10.1002/jgt.3190010114.
- 34 Yongge Tian. The minimum rank of a  $3 \times 3$  partial block matrix. *Linear Multilinear Algebra*, 50(2):125–131, 2002. doi:10.1080/03081080290019531.
- 35 W. T. Tutte. Toward a theory of crossing numbers. *J. Combinatorial Theory*, 8:45–53, 1970. doi:10.1016/S0021-9800(70)80007-2.
- 36 H. J. Woerdeman. The lower order of lower triangular operators and minimal rank extensions. *Integral Equations Operator Theory*, 10(6):859–879, 1987. doi:10.1007/BF01196124.



# An Experimental Study of Forbidden Patterns in Geometric Permutations by Combinatorial Lifting

**Xavier Goaoc**

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France  
xavier.goaoc@loria.fr

**Andreas Holmsen**

Department of Mathematical Sciences, KAIST, Daejeon, South Korea  
andreash@kaist.edu

**Cyril Nicaud**

Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM,  
F-77454, Marne-la-Vallée, France  
cyril.nicaud@u-pem.fr

---

## Abstract

We study the problem of deciding if a given triple of permutations can be realized as geometric permutations of disjoint convex sets in  $\mathbb{R}^3$ . We show that this question, which is equivalent to deciding the emptiness of certain semi-algebraic sets bounded by cubic polynomials, can be “lifted” to a purely combinatorial problem. We propose an effective algorithm for that problem, and use it to gain new insights into the structure of geometric permutations.

**2012 ACM Subject Classification** Theory of computation → Computational geometry; Computing methodologies → Combinatorial algorithms

**Keywords and phrases** Geometric permutation, Emptiness testing of semi-algebraic sets, Computer-aided proof

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.40

**Related Version** A full version of this paper, available at <https://arxiv.org/abs/1903.03014>, contains some details omitted from this version due to space constraints.

**Funding** *Xavier Goaoc*: Supported by Institut Universitaire de France.

*Andreas Holmsen*: Supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03930998).

## 1 Introduction

Consider pairwise disjoint convex sets  $C_1, C_2, \dots, C_n$  and lines  $\ell_1, \ell_2, \dots, \ell_k$  in  $\mathbb{R}^d$ , where every line intersects every set. Each line  $\ell_i$  defines two orders on the sets, namely the orders in which the two orientations of  $\ell_i$  meet the sets; this pair of orders, one the reverse of the other, are identified to form the *geometric permutation* realized by  $\ell_i$  on  $C_1, C_2, \dots, C_n$ . Going in the other direction, one may ask if a given family of permutations can occur as geometric permutations of a family of pairwise disjoint convex sets in  $\mathbb{R}^d$ , *i.e.* whether it is *geometrically realizable* in  $\mathbb{R}^d$ .

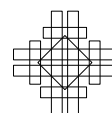
In  $\mathbb{R}^2$  there exist pairs of permutations that are unrealizable, while in  $\mathbb{R}^3$ , every pair of permutations is realizable by a family of segments with endpoints on two skew lines. The simplest non-trivial question is therefore to understand which triples of permutations are geometrically realizable. This question is equivalent to testing the non-emptiness of certain semi-algebraic sets bounded by cubic polynomials. We show that the structure of these



© Xavier Goaoc, Andreas Holmsen, and Cyril Nicaud;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 40; pp. 40:1–40:16  
Leibniz International Proceedings in Informatics



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





polynomials allow to “lift” this algebraic question to a purely combinatorial one, then propose an algorithm for that combinatorial problem, and present some new results on geometric permutations obtained with its assistance.

**Conventions.** To simplify the discussion, we work with *oriented* lines and thus with *permutations*, in place of the non-oriented lines and geometric permutations customary in this line of inquiry. We represent permutations by words such as 1423 or *badc*, to be interpreted as follows. The letters of the word are the elements being permuted and they come with a natural order, namely  $<$  for integer and the alphabetical order for letters. The word gives the sequence of images of the elements by increasing order; for example, 312 codes the permutation mapping 1 to 3, 2 to 1 and 3 to 2, and *badc* codes the permutation exchanging *a* with *b* and *c* with *d*. The *size* of a permutation is the number of elements being permuted, that is the length of this word. We say that a triple of permutations is *realizable* (resp. *forbidden*) to mean that it is realizable (resp. not realizable) in  $\mathbb{R}^3$ .

## 1.1 Contributions

Our results are of two types, methodological and geometrical.

**Combinatorial lifting.** Our first contribution is a new approach for deciding the emptiness of a semi-algebraic set with a special structure. We describe it for the geometric realizability problem, here and in Section 3, but stress that it applies more broadly.

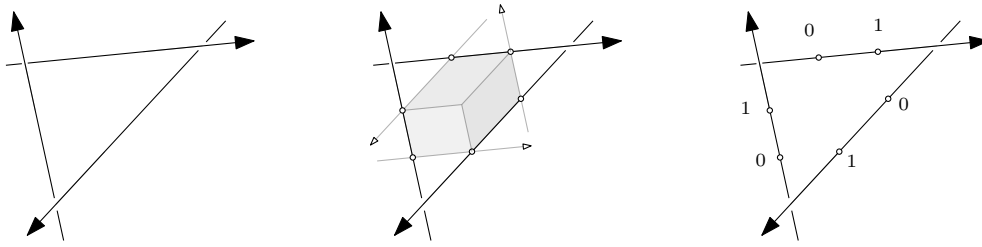
As spelled out in Section 2, deciding if a triple of permutations is realizable amounts to testing the emptiness of a semi-algebraic set  $R \subseteq \mathbb{R}^n$ . Let  $u_1, u_2, \dots, u_n$  denote the variables and  $P_1, P_2, \dots, P_m$  the polynomials used in a Boolean formula defining  $R$ . The structure we take advantage of is that here, each  $P_k$  can be written as a product of terms, each of which is of the form  $u_i - u_j$ ,  $u_i - 1$ , or  $u_i - f(u_j)$ , with  $f(t) = \frac{1}{1-t}$ . If only terms of the form  $u_i - u_j$  or  $u_i - 1$  occur, then we can test the emptiness of  $R$  by examining the possible orders on  $(1, u_1, u_2, \dots, u_n)$ . We propose to handle the terms of the form  $u_i - f(u_j)$  in the same way, by exploring the orders that can arise on  $(1, u_1, f(u_1), u_2, f(u_2), \dots, u_n, f(u_n))$ .

The main difficulty in this approach is to restrict the exploration to the orders that can be realized by a sequence of the form  $(1, u_1, f(u_1), u_2, f(u_2), \dots, u_n, f(u_n))$ . This turns out to be easier if we extend the lifting to

$$\Lambda : \begin{cases} (\mathbb{R} \setminus \{0, 1\})^n & \rightarrow \mathbb{R}^{3n} \\ (u_1, u_2, \dots, u_n) & \mapsto (u_1, f(u_1), f^{(2)}(u_1), \dots, u_n, f(u_n), f^{(2)}(u_n)). \end{cases}$$

This extended lifting allows to take advantage of the facts that  $f^{(3)} = f \circ f \circ f$  is the identity, that  $f$  permutes circularly the intervals  $(-\infty, 0)$ ,  $(0, 1)$  and  $(1, \infty)$ , and that  $f$  is increasing on each of them. In Proposition 7, we essentially show that *an order on the  $3n$  lifted variables can be realized by a point of  $\Lambda(\mathbb{R}^{3n})$  if and only if it is compatible with the action of  $f$ , as captured by these properties.*

**Algorithm.** Our second contribution is an algorithm that puts the combinatorial lifting in practice, and decides if a triple of permutations is realizable in  $O(6^n n^{10})$  time and  $O(n^2)$  space in the worst-case. We provide an implementation in Python, see the full version of this paper.



■ **Figure 1** Three skew lines (left), the parallelotope (middle) and the marked points (right).

**New geometric results.** Our remaining contributions are new geometric results obtained with the aid of our implementation. A first systematic exploration reveals:

► **Theorem 1.** *Every triple of permutations of size 5 is geometrically realizable in  $\mathbb{R}^3$ .*

The smallest known triple of geometric permutations forbidden in  $\mathbb{R}^3$  has size 6 (see Section 1.2), so Theorem 1 proves that it is minimal. We also obtained the complete list of forbidden triples of size 6 (see the full version). Interestingly, although everything is realizable up to size 5, something can be said on geometric permutations of size 4. Recall that the *side operator*  $(pq) \odot (rs)$  of the two lines  $(pq)$  and  $(rs)$ , oriented respectively from  $p$  to  $q$  and from  $r$  to  $s$ , is the orientation of the tetrahedron  $pqrs$ ; it captures the mutual disposition of the two lines. We prove:

► **Theorem 2.** *Let  $\ell_1$  and  $\ell_2$  be two oriented lines intersecting four pairwise disjoint convex sets in the order 1234. Any oriented line  $\ell_3$  that intersects those four sets in the order 2143 satisfies  $\ell_1 \odot \ell_3 = \ell_2 \odot \ell_3$ .*

The pattern (1234, 2143) is known to be forbidden in some cases (see Section 1.2), but this is the first condition valid for arbitrary disjoint convex sets. We could prove Theorem 2 because our algorithm solves a more constrained problem than just realizability of permutations. Given three lines in general position in  $\mathbb{R}^3$ , there is a unique parallelotope with three disjoint edges supported on these three lines (see Figure 1). Combinatorial lifting, and therefore our algorithm, can decide whether three permutations can be realized *with the vertices of that parallelotope in prescribed positions in the permutations*.

We label the vertices of the parallelotope with  $0$  and  $1$  as in Figure 1 and work with permutations where two extra elements,  $0$  and  $1$ , are inserted; we call them *tagged permutations*. We examine triples of tagged permutations realizable on a canonical system of lines (see Equation (1)), and characterize those minimally unrealizable up to size 4 (Proposition 10); for size 2 and 3, we provide independent, direct, geometric proofs of unrealizability (Section 7). We conjecture that no other minimally unrealizable triples of tagged permutations exist, and verified this experimentally up to size 6 (not counting  $0$  and  $1$ ). A weaker conjecture is:

► **Conjecture 3.** *There exists a polynomial time algorithm that decides the geometric realizability of a triple of permutations of size  $n$  in  $\mathbb{R}^3$ .*

## 1.2 Discussion and related work

We now put our contribution in context, starting with motivations for studying geometric permutations.

**Geometric transversals.** In the 1950's, Grünbaum [7] conjectured that, given a family of disjoint translates of a convex figure in the plane, if every five members of the family can be met by a line, then there exists a line that meets the entire family. (Such a statement, if true, is an example of *Helly-type theorem*.) Progress on Grünbaum's conjecture was slow until the 1980's, when the notion of geometric permutations of families of convex sets was introduced [13, 14]. Their systematic study in the plane was refined by Katchalski [12] in order to prove a weak version of Grünbaum's conjecture (with 128 in place of 5). Tverberg [24] soon followed up with a proof of the conjecture, again using a careful analysis of planar geometric permutations. This initial success and further conjectures about Helly-type theorems stimulated a more systematic study of geometric permutations realizable under various geometric restrictions; cf. [10] and the references therein.

Another motivation to study geometric permutations comes from computational geometry, more precisely the study of geometric structures such as *arrangements*. There, geometric permutations appear as a coarse measure of complexity of the space of line transversals to families of sets, and relates to various algorithmic problems such as ray-shooting or smallest enclosing cylinder computation [18, §7.6]. From this point of view, the main question is to estimate the maximum number  $g_d(n)$  of distinct geometric permutations of  $n$  pairwise disjoint convex sets in  $\mathbb{R}^d$ . Broadly speaking, while  $g_2(n)$  is known to equal  $2n - 2$  [6], even the order of magnitude of  $g_d(n)$  as  $n \rightarrow \infty$  is open for every  $d \geq 3$ ; the gap is between  $\Omega(n^{d-1})$  and  $O(n^{2d-3} \log n)$ . Bridging this gap has been identified as an important problem in discrete geometry [18, §7.6], yet, over the last fifteen years, the only progress has been an improvement of the upper bound from  $O(n^{2d-2})$  down to  $O(n^{2d-3} \log n)$ ; moreover, while the former bound follows from a fairly direct argument, the latter is a technical tour de force [20]. We hope that a better understanding of small forbidden configurations will suggest new approaches to this question.

**Geometric realizability problems.** Combinatorial structures that arise from geometric configurations such as arrangements, polytopes, or intersection graphs are classical objects of enquiry in discrete and computational geometry (see *e.g.* [23, § 1, 5, 6, 10, 15, 17, 28]). We are concerned here with the *membership testing problem*: given an instance of a combinatorial structure, decide if there exists a geometric configuration that induces it. Such problems can be difficult: for instance, deciding whether a given graph can be obtained as intersection graph of segments in the plane is NP-hard [15].

A natural approach to membership testing is to parameterize the candidate geometric configuration and express the combinatorial structure as conditions on these parameters. This often results in a semi-algebraic set. In the real-RAM model<sup>1</sup>, the emptiness of a semi-algebraic set in  $\mathbb{R}^d$  with real coefficients can be tested in time  $(nD)^{O(d)}$  [19, Prop. 4.1], where  $n$  is the number of polynomials and  $D$  their maximum degree. (Other approaches exist but have worse complexity bounds, see [5, 16, 11, 4]). Given three permutations of size  $n$ , we describe their realizations as a semi-algebraic set defined by  $O(n^2)$  cubic polynomials in  $n$  variables; the above method thus has complexity  $n^{O(n)}$ , making our  $O(6^n n^{10})$  solution competitive in theory. Practical effectiveness is usually difficult to predict as it depends on the geometry of the underlying algebraic surfaces; for example, deciding if two geometric permutations of *size four* are realizable by disjoint unit balls in  $\mathbb{R}^3$  was recently checked to be out of reach [9].

---

<sup>1</sup> See the same reference for a similar bound in the bit model.

Some geometric realizability problems, for example the recognition of unit disk graphs, are  $\exists\mathbb{R}$ -hard [21] and therefore as difficult from a complexity point of view as deciding the emptiness of a general semi-algebraic set. We do not know whether deciding the emptiness of semi-algebraic sets amenable to our combinatorial lifting remains  $\exists\mathbb{R}$ -hard; we believe, however, that deciding if a triple of permutations is realizable is not (cf. Conjecture 3).

**Forbidden patterns.** A dimension count shows that any  $k$  permutations are realizable in  $\mathbb{R}^{2k-1}$ . Our most direct predecessor is the work of Asinowski and Katchalski [2] who proved that this argument is sharp by constructing, for every  $k$ , a set of  $k$  permutations that are not realizable in  $\mathbb{R}^{2k-2}$ . They also showed that the triple  $(123456, 321654, 246135)$  is not realizable in  $\mathbb{R}^3$ , a fact that follows easily from our list of obstructions.

In a sense, our work tries to generalize some arguments previously used to analyze geometric permutations in the plane. For example, the (standard) proof that the pair  $(1234, 2143)$  is non-realizable as geometric permutations in  $\mathbb{R}^2$  essentially analyzes tagged permutations. Indeed, if we augment the permutations by an additional label  $^0$  marking the intersection of the lines realizing the two orders, we get that  $(^0ab, ^0ba)$ ,  $(^0ab, ab^0)$ ,  $(ba^0, ^0ba)$  and  $(ba^0, ab^0)$  are forbidden, and there is nowhere to place  $^0$  in  $(1234, 2143)$ . We should, however, emphasize that already in  $\mathbb{R}^3$  the geometry is much more subtle.

Forbidden patterns were used to bound the number of geometric permutations for certain restricted families of convex sets. For pairwise disjoint translates of a convex planar figure [12, 24, 25], it is known that a given family can have at most three geometric permutations, and the possible sets of realizable geometric permutations have been characterized. The situation is similar for families of pairwise disjoint unit balls in  $\mathbb{R}^d$ . Here, an analysis of forbidden patterns in geometric permutations showed that a given family can have at most a constant number of geometric permutations (in fact only two if the family is sufficiently large) [22, 3, 9]. Another example is [1], where it is shown that the maximum number of geometric permutations for convex objects in  $\mathbb{R}^d$  induced by lines that pass through the origin, is in  $\Theta(n^{d-1})$ . The restriction that the lines pass through the origin, allows them to deal with permutations augmented by one additional label, and their argument relies on the forbidden tagged pattern  $(^0ab, ^0ba)$  [1, Lemma 2.1].

In these examples, the bounds use highly structured sets of forbidden patterns. In general, one cannot expect polynomial bounds on the sole basis of excluding a handful of patterns; for instance it is not hard to construct an exponential size family of permutations of  $[n]$  which avoids the pattern  $(1234, 2143)$ . Such questions are well-studied in the area of “pattern-avoidance” and usually the best one could hope for is an exponential upper bound on the size of the family [17].

## 2 Semi-algebraic parameterization

Let  $P = (\pi_1, \pi_2, \pi_3)$  denote a triple of permutations of  $\{1, 2, \dots, n\}$ . We now describe a semi-algebraic set that is nonempty if and only if  $P$  has a geometric realization in  $\mathbb{R}^3$ .

**Canonical realizations.** We say that a geometric realization of  $P$  is *canonical* if the oriented line transversals are

$$\ell_x = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \mathbb{R} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \ell_y = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + \mathbb{R} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad \ell_z = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \mathbb{R} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad (1)$$

and if the convex sets are triangles with vertices on  $\ell_x$ ,  $\ell_y$  and  $\ell_z$ .

► **Lemma 4.** *If  $P$  is geometrically realizable in  $\mathbb{R}^3$ , then it has a canonical realization, possibly after reversing some of the permutations  $\pi_i$ .*

**Proof.** Consider a realization of  $P$  by three lines and  $n$  pairwise disjoint sets. For each convex set we select a point from the intersection with each of the lines and replace it by the (possibly degenerate) triangle spanned by these points. This realizes  $P$  by compact convex sets. By taking the Minkowski sum of each set with a sufficiently small ball, the sets remain disjoint and the lines intersect the sets in their interior. We may now perturb the lines into three lines  $L_x, L_y$  and  $L_z$  that are pairwise skew and not all parallel to a common plane. We then, again, crop each set to a triangle with vertices on  $L_x, L_y$  and  $L_z$ .

We now use an affine map to send our three lines to  $\ell_x, \ell_y$  and  $\ell_z$ . An affine transform is defined by 12 parameters and fixing the image of one line amounts to four linear conditions on these parameters; these constraints determine a unique transform because the lines are in general position. Note, however, that the oriented line  $L_x$  is mapped to either  $\ell_x$  or  $-\ell_x$ , so  $\pi_1$  may have to be reversed; the same applies to the permutations  $\pi_2$  and  $\pi_3$ . ◀

We equip the line  $\ell_x$  (resp.  $\ell_y, \ell_z$ ) with the coordinate system obtained by projecting the  $x$ -coordinate (resp.  $y$ -coordinate,  $z$ -coordinate) of  $\mathbb{R}^3$ . This parameterizes the space of canonical realizations by  $\mathbb{R}^{3n}$ . Specifically, we equip  $\mathbb{R}^{3n}$  with a coordinate system  $(O, x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, z_1, z_2, \dots, z_n)$  and for any point  $c \in \mathbb{R}^{3n}$  we put

$$\mathcal{T}(c) = \{\text{conv}\{X_i, Y_i, Z_i\}\}_{1 \leq i \leq n}, \quad \text{where } X_i = \begin{pmatrix} x_i \\ 1 \\ 0 \end{pmatrix}, \quad Y_i = \begin{pmatrix} 0 \\ y_i \\ 1 \end{pmatrix}, \quad \text{and } Z_i = \begin{pmatrix} 1 \\ 0 \\ z_i \end{pmatrix}.$$

Each element of  $\mathcal{T}(c)$  is thus a triangle with a vertex on each of  $\ell_x, \ell_y$  and  $\ell_z$ . We define:

$$R = \{c \in \mathbb{R}^{3n} : \mathcal{T}(c) \text{ consists of disjoint triangles and realizes } P\}.$$

The triple  $P$  is realizable if and only if  $R$  is non-empty.

**Triangle disjointness.** We now review an algorithm of Guigue and Devillers [8] to decide if two triangles are disjoint, and use it to formulate the condition that two triangles  $X_i Y_i Z_i$  and  $X_j Y_j Z_j$  be disjoint as a semi-algebraic condition on  $x_i, \dots, z_j$ .

The algorithm and our description are expressed in terms of orientations, where the *orientation* of four points  $p, q, r, s \in \mathbb{R}^3$  is

$$[p, q, r, s] \stackrel{\text{def}}{=} \text{sign det} \begin{pmatrix} x_p & x_q & x_r & x_s \\ y_p & y_q & y_r & y_s \\ z_p & z_q & z_r & z_s \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Intuitively, the orientation indicates whether point  $s$  is “above” (+1), on (0), or “below” (−1) the plane spanned by  $p, q, r$ , where above and below refer to the orientation of the plane that makes the directed triangle  $pqr$  positively oriented. We only consider orientations of non-coplanar quadruples of points, so orientations take values in  $\{\pm 1\}$ .

If one triangle is on one side of the plane spanned by the other, then the triangles are disjoint. We check this by computing

$$v(i, j) \stackrel{\text{def}}{=} \begin{pmatrix} [X_i, Y_i, Z_i, X_j] \\ [X_i, Y_i, Z_i, Y_j] \\ [X_i, Y_i, Z_i, Z_j] \\ [X_j, Y_j, Z_j, X_i] \\ [X_j, Y_j, Z_j, Y_i] \\ [X_j, Y_j, Z_j, Z_i] \end{pmatrix} \in \{-1, 1\}^6$$

and testing if  $v(i, j)_1 = v(i, j)_2 = v(i, j)_3$  or  $v(i, j)_4 = v(i, j)_5 = v(i, j)_6$ . If this fails, then we rename  $\{X_i, Y_i, Z_i\}$  into  $\{A_i, B_i, C_i\}$  and  $\{X_j, Y_j, Z_j\}$  into  $\{A_j, B_j, C_j\}$  so that

$$\begin{pmatrix} [A_i, B_i, C_i, A_j] \\ [A_i, B_i, C_i, B_j] \\ [A_i, B_i, C_i, C_j] \\ [A_j, B_j, C_j, A_i] \\ [A_j, B_j, C_j, B_i] \\ [A_j, B_j, C_j, C_i] \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \end{pmatrix}.$$

Then, the triangles are disjoint if and only if  $[A_i, B_i, A_j, B_j] = 1$  or  $[A_i, C_i, C_j, A_j] = 1$  [8]. The renaming is done as follows. Since the first test is inconclusive, the plane spanned by a triple of points separates the other triple of points. We let  $(A_i, B_i, C_i)$  be the circular permutation of  $(X_i, Y_i, Z_i)$  such that  $A_i$  is separated from  $B_i$  and  $C_i$  by the plane spanned by  $X_j, Y_j$ , and  $Z_j$ . We let  $(A_j, B_j, C_j)$  be the circular permutation of  $(X_j, Y_j, Z_j)$  such that  $A_j$  is separated from  $B_j$  and  $C_j$  by the plane spanned by  $A_i, B_i$ , and  $C_i$ . If  $[A_i, B_i, C_i, A_j] = -1$  then we exchange  $B_i$  and  $C_i$ . If  $[A_j, B_j, C_j, A_i] = -1$  then we exchange  $B_j$  and  $C_j$ .

**Semi-algebraicity.** Every step in the Guigue-Devillers algorithm can be expressed as a logical proposition in terms of orientation predicates which are, when specialized to our parameterization, conditions on the sign of polynomials in the coordinates of  $c$ . Checking that each of  $\ell_x, \ell_y$  and  $\ell_z$  intersects the triangles in the prescribed order amounts to comparing coordinates of  $c$ . Altogether, the set  $R$  is a semi-algebraic subset of  $\mathbb{R}^{3n}$ .

### 3 Combinatorial lifting

We now explain how to test combinatorially the emptiness of our semi-algebraic set  $R$ .

**Definitions.** We start by decomposing each orientation predicate used in the definition of  $R$  as indicated in Table 1. For the last three rows, this is not a factorization since one of the factors is of the form  $u - f(v)$  where  $f : t \mapsto \frac{1}{1-t}$ .

In light of the third column of Table 1, it may seem natural to “linearize” the problem by considering the map  $(x_1, x_2, \dots, z_n) \mapsto (x_1, f(x_1), x_2, f(x_2), \dots, z_n, f(z_n))$  from  $\mathbb{R}^{3n}$  to  $\mathbb{R}^{6n}$ . Indeed, the order on the lifted coordinates and 1 determines the sign of all polynomials defining  $R$ . We must, however, identify the orders on the coordinates in  $\mathbb{R}^{6n}$  that can be realized by lifts of points from  $\mathbb{R}^{3n}$ . Perhaps surprisingly, the task gets easier if we lift to even higher dimension. For convenience we let  $\mathbb{R}_* \stackrel{\text{def}}{=} \mathbb{R} \setminus \{0, 1\}$ . The lifting map we use is:

$$\Lambda : \begin{cases} \mathbb{R}_*^{3n} & \rightarrow \mathbb{R}^{9n} \\ (x_1, x_2, \dots, z_n) & \mapsto (x_1, f(x_1), f^{(2)}(x_1), x_2, \dots, z_n, f(z_n), f^{(2)}(z_n)) \end{cases}$$

To determine the image of  $\Lambda(\mathbb{R}_*^{3n})$ , we will use the following properties of  $f$ :

■ **Table 1** Orientation predicates used in the Guigue-Devillers algorithm when specialized to points from  $\ell_x, \ell_y$  and  $\ell_z$ .

| Orientation            | Determinant                      | Decomposition                                                  |
|------------------------|----------------------------------|----------------------------------------------------------------|
| $[X_a, X_b, Y_c, Y_d]$ | $(x_a - x_b)(y_c - y_d)$         | $(x_a - x_b)(y_c - y_d)$                                       |
| $[X_a, X_b, Z_c, Z_d]$ | $(x_a - x_b)(z_c - z_d)$         | $(x_a - x_b)(z_c - z_d)$                                       |
| $[Y_a, Y_b, Z_c, Z_d]$ | $(y_a - y_b)(z_c - z_d)$         | $(y_a - y_b)(z_c - z_d)$                                       |
| $[X_a, X_b, Y_c, Z_d]$ | $(x_a - x_b)(y_c z_d - z_d + 1)$ | $(x_a - x_b)(y_c - 1) \left( z_d - \frac{1}{1 - y_c} \right)$  |
| $[X_a, Y_b, Y_c, Z_d]$ | $(y_b - y_c)(x_a - x_a z_d - 1)$ | $-(y_b - y_c)(z_d - 1) \left( x_a - \frac{1}{1 - z_d} \right)$ |
| $[X_a, Y_b, Z_c, Z_d]$ | $(z_c - z_d)(x_a y_b + 1 - y_b)$ | $(z_c - z_d)(x_a - 1) \left( y_b - \frac{1}{1 - x_a} \right)$  |

▷ **Claim 5.**  $f^{(3)} = f \circ f \circ f$  is the identity on  $\mathbb{R}_*$ ,  $f$  permutes the intervals  $(-\infty, 0)$ ,  $(0, 1)$  and  $(1, +\infty)$  circularly, and  $f$  is monotone on each of these intervals.

Let us denote the points of  $\mathbb{R}^{9n}$  by vectors  $(\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_{9n})$ . We next “lift” the semi-algebraic description of  $R$ :

1. We pick a Boolean formula  $\phi$  describing  $R$  in terms of orientations (for the triangle disjointedness) and comparisons of coordinates (for the geometric permutations).
2. We decompose every orientation predicate occurring in  $\phi$  as in the third row of Table 1.
3. We then construct another Boolean formula  $\psi$  by substituting<sup>2</sup> in  $\phi$  every  $f(x_1)$  by the variable  $\mathbf{t}_2$  (to which it is mapped under  $\Lambda$ ). We similarly substitute every  $f(x_i)$ ,  $f(y_i)$  and  $f(z_i)$ , then every remaining  $x_i$ ,  $y_i$  and  $z_i$  by the corresponding variable  $\mathbf{t}_*$ .
4. We let  $S \subset \mathbb{R}^{9n}$  be the (semi-algebraic) set of points that satisfy  $\psi$ .

We finally let  $\mathcal{H}$  denote the arrangement in  $\mathbb{R}^{9n}$  of the set of hyperplanes:

$$\{\mathbf{t}_i = \mathbf{t}_j\}_{1 \leq i < j \leq 9n} \cup \{\mathbf{t}_i = 0\}_{1 \leq i \leq 9n} \cup \{\mathbf{t}_i = 1\}_{1 \leq i \leq 9n}.$$

Note that the full-dimensional (open) cells in  $\mathcal{H}$  are in bijection with the total orders on  $\{0, 1, \mathbf{t}_1, \dots, \mathbf{t}_{9n}\}$  in which 0 comes before 1. We write  $\prec_A$  for the order associated with a full-dimensional cell  $A$  of  $\mathcal{H}$ .

► **Lemma 6.** *Every full-dimensional cell of  $\mathcal{H}$  is disjoint from or contained in  $S$ . Moreover,  $R$  is nonempty if and only if there exists a full-dimensional cell of  $\mathcal{H}$  that is contained in  $S$  and intersects  $\Lambda(\mathbb{R}_*^{3n})$ .*

**Proof.** The set  $S$  is defined by the positivity or negativity of polynomials, each of which is a product of terms of the form  $(\mathbf{t}_i - \mathbf{t}_j)$  or  $(\mathbf{t}_i - 1)$ . The first statement thus follows from the fact the coordinates of all points in a full-dimensional cell realize the same order on  $\{0, 1, \mathbf{t}_1, \dots, \mathbf{t}_{9n}\}$ . By the perturbation argument used in the proof of Lemma 4, if  $R$  is non-empty, then it contains a point with no coordinate in  $\{0, 1\}$ . Thus,  $R$  is non-empty if and only if  $\Lambda(R)$  is non-empty. The construction of  $S$  ensures that  $\Lambda(R) = S \cap \Lambda(\mathbb{R}_*^{3n})$ . Again, a perturbation argument ensures that if  $\Lambda(R)$  is nonempty, it contains a point outside of the union of the hyperplanes of  $\mathcal{H}$ . The second statement follows. ◀

<sup>2</sup> For example, with  $n = 3$ , the product  $(x_1 - x_2)(y_2 - 1) \left( z_3 - \frac{1}{1 - y_2} \right) = (x_1 - x_2)(y_2 - 1)(z_3 - f(y_2))$  appearing in  $\phi$  is translated in  $\psi$  as  $(\mathbf{t}_1 - \mathbf{t}_4)(\mathbf{t}_{13} - 1)(\mathbf{t}_{25} - \mathbf{t}_{14})$ .

**Zone characterization.** Inspired by Lemma 6, we now characterize the orders  $\prec_A$  such that  $A$  intersects  $\Lambda(\mathbb{R}_*^{3n})$ . We split the  $9n$  variables  $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_{9n}$  into  $3n$  blocks of three consecutive variables  $\mathbf{t}_{3i+1}, \mathbf{t}_{3i+2}, \mathbf{t}_{3i+3}$  (representing  $x_i, f(x_i), f^{(2)}(x_i)$  for  $0 \leq i < n$ ,  $y_i, f(y_i), f^{(2)}(y_i)$  for  $n \leq i < 2n$ , and  $z_i, f(z_i), f^{(2)}(z_i)$  for  $2n \leq i < 3n$ ). We also define an operator  $\mathbf{f}$  that shifts the variables cyclically within each individual block:

$$\mathbf{f}(\mathbf{t}_{3i+1}) = \mathbf{t}_{3i+2}, \quad \mathbf{f}(\mathbf{t}_{3i+2}) = \mathbf{t}_{3i+3} \quad \text{and} \quad \mathbf{f}(\mathbf{t}_{3i+3}) = \mathbf{t}_{3i+1}.$$

By convention,  $\mathbf{f}^0$  means the identity. The fact that  $\mathbf{f}$  mimicks, symbolically, the action of  $f$  yields the following characterization.

► **Proposition 7.** *A full-dimensional cell  $A$  of  $\mathcal{H}$  intersects  $\Lambda(\mathbb{R}_*^{3n})$  if and only if*

(i) *For any  $0 \leq i < 3n$ , there exists  $j \in \{0, 1, 2\}$  s. t.*

$$\mathbf{f}^{(j)}(\mathbf{t}_{3i+1}) \prec_A 0 \prec_A \mathbf{f}^{(j+1)}(\mathbf{t}_{3i+1}) \prec_A 1 \prec_A \mathbf{f}^{(j+2)}(\mathbf{t}_{3i+1}).$$

(ii) *For any  $1 \leq i, j \leq 9n$ ,  $\{\mathbf{t}_i \prec_A \mathbf{t}_j \text{ and } \mathbf{f}(\mathbf{t}_j) \prec_A \mathbf{f}(\mathbf{t}_i)\} \Rightarrow \mathbf{t}_i \prec_A 1 \prec_A \mathbf{t}_j$ .*

**Proof.** Let us first see why the conditions are necessary. Let  $c = (x_1, x_2, \dots, z_n) \in \mathbb{R}_*^{3n}$  such that  $\Lambda(c) \in A$ . Fix some  $0 \leq i < n$ . As  $j$  ranges over  $\{0, 1, 2\}$ , the coordinate  $\mathbf{f}^{(j)}(\mathbf{t}_{3i+1})$  of  $\Lambda(c)$  ranges over  $\{x_i, f(x_i), f^{(2)}(x_i)\}$ , and Condition (i) holds because  $f$  permutes the intervals  $(-\infty, 0)$ ,  $(0, 1)$  and  $(1, +\infty)$  circularly. The cases  $n \leq i < 3n$  are similar. Condition (ii) follows in a similar manner from the fact that  $f$  permutes the intervals  $(-\infty, 0)$ ,  $(0, 1)$  and  $(1, +\infty)$  circularly and is increasing on each of them.

To examine sufficiency we need some notations. We let  $\mathcal{V} = \{0, 1, \mathbf{t}_1, \dots, \mathbf{t}_{9n}\}$ . Given an order  $\prec$  on  $\mathcal{V}$  and two elements  $a, b \in \mathcal{V}$  we write  $(a, b)_\prec \stackrel{\text{def}}{=} \{c \in \mathcal{V} : a \prec c \prec b\}$ . We also write  $(\cdot, a)_\prec$  for the set of elements smaller than  $a$ , and  $(a, \cdot)_\prec$  for the set of elements larger than  $a$ , and  $[a, b)_\prec, [a, b]_\prec$  or  $[a, b]_\prec$  to include one or both bounds in the interval.

Let  $\prec_*$  be an order on  $\{0, 1, \mathbf{t}_1, \dots, \mathbf{t}_{9n}\}$  such that  $0 \prec_* 1$ . By Condition (i),  $(1, \cdot)_{\prec_*}$  has size  $3n$ , so let us write  $(1, \cdot)_{\prec_*} = \{b_1, b_2, \dots, b_{3n}\}$  with  $1 \prec_* b_1 \prec_* b_2 \prec_* \dots \prec_* b_{3n}$ . Condition (i) also ensures that for every  $0 \leq i < 3n$ , exactly one of  $\{\mathbf{t}_{3i+1}, \mathbf{f}(\mathbf{t}_{3i+1}), \mathbf{f}^{(2)}(\mathbf{t}_{3i+1})\}$  belongs to  $(1, \cdot)_{\prec_*}$ . Hence, for every  $0 \leq i < 3n$  there are uniquely defined integers  $0 \leq \alpha(i) \leq 2$  and  $1 \leq \beta(i) \leq 3n$  such that  $b_{\beta(i)} = \mathbf{f}^{\alpha(i)}(\mathbf{t}_{3i+1})$ .

We next pick  $3n$  real numbers  $1 < r_1 < r_2 < \dots < r_{3n}$ , put

$$\begin{aligned} x_i &= f^{3-\alpha(i)}(r_{\beta(i)}), & \text{for } 0 \leq i < n \\ y_i &= f^{3-\alpha(i)}(r_{\beta(i)}), & \text{for } n \leq i < 2n \\ z_i &= f^{3-\alpha(i)}(r_{\beta(i)}), & \text{for } 2n \leq i < 3n, \end{aligned}$$

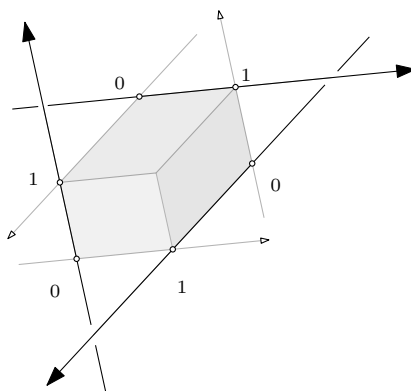
and let  $p = (x_1, \dots, x_n, y_1, \dots, y_n, z_1, \dots, z_n) \in \mathbb{R}^{3n}$ . Note that  $\Lambda(p)$  lies in a full-dimensional cell of the arrangement  $\mathcal{H}$ ; let us denote it by  $A$ .

Now,  $0$  precedes  $1$  in both  $\prec_*$  and  $\prec_A$ . Also,  $[1, \cdot)_{\prec_*} = [1, \cdot)_{\prec_A}$  and the two orders coincide on that interval by construction of  $p$ . Remark that  $\mathbf{f}$  acts similarly for both orders:

- $\mathbf{f}$  maps  $[1, \cdot)_{\prec_*}$  to  $(\cdot, 0]_{\prec_*}$  increasingly for  $\prec_*$  by Conditions (i) and (ii).
- $\mathbf{f}$  maps  $[1, \cdot)_{\prec_A}$  to  $(\cdot, 0]_{\prec_A}$  increasingly for  $\prec_A$  by definition of  $p$  and Claim 5.

We therefore also have  $(\cdot, 0]_{\prec_*} = (\cdot, 0]_{\prec_A}$  and the orders coincide on that interval as well. The same argument applied to  $\mathbf{f}^2$  shows that  $[0, 1]_{\prec_*} = [0, 1]_{\prec_A}$  and that the two orders coincide on that interval as well. Altogether,  $\prec_*$  and  $\prec_A$  coincide. ◀





■ **Figure 2** Tagging permutations by adding the  $^0$  and  $^1$ .

#### 4 Tagged patterns and their canonical realization

Let us clarify the geometric problem that is really captured by our combinatorial lifting.

As it appears from Table 1, the combinatorial lift searches only for a *canonical* realization. By Lemma 4, however, this is not a restriction. More importantly, the lift compares the lifted variable to the constants 0 and 1. In the coordinate systems of  $\ell_x$ ,  $\ell_y$  and  $\ell_z$ , the 0 and 1 are at certain corners of the unique parallelotope with three disjoint edges supported by these lines. Let us label these points  $^0$  and  $^1$  as Figure 2. Note that fixing the comparisons of 0 and each  $x_i$  is equivalent to specifying the position of the point  $^0$  of  $\ell_x$  in the first permutation. Other coordinates behave similarly.

Formally, we define a *tagged permutation* as a permutation of  $\{^0, ^1, 1, 2, \dots, n\}$  in which  $^0$  precedes  $^1$ . We call a triple of tagged permutations a *tagged pattern*. A *canonical realization* of a tagged pattern is a set of triangles, with vertices on  $\ell_x$ ,  $\ell_y$  and  $\ell_z$ , such that  $\ell_x$  (resp.  $\ell_y$ ,  $\ell_z$ ) intersects the triangles in the first (resp. second, third) permutation and such that the tagged corners of the parallelotope appear in the right position on each line.

Our experiments will use two more notions. Two tagged patterns are *equivalent* for canonical realizability if one can be transformed into the other by (i) relabeling the symbols other than  $^0$  and  $^1$  bijectively, and (ii) applying a circular permutation to the triple. A tagged pattern is *minimally forbidden* if it has no canonical realization, and deleting any symbol other than  $^0$  and  $^1$  from the three tagged permutations produces a tagged pattern which has a canonical realization.

#### 5 Algorithm

We now present an algorithm that takes a tagged pattern as input and decides if it admits a canonical realization. Our initial problem of testing the geometric realizability of a triple of permutations of size  $n$  reduces to  $8 \binom{n+2}{2}^3$  instances of that problem.

##### 5.1 Outline

Following Sections 2 and 3, we search for an order on  $\{0, 1, \mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_{9n}\}$  satisfying the conditions of Proposition 7 and the formula  $\psi$  (which defines  $S$ ). To save breath, we call such an order *good*. We say that *triangles  $i$  and  $j$  are disjoint in a partial order  $\mathcal{P}$*  if for every  $c \in \mathbb{R}^{3n}$  such that the order on  $\Lambda(c)$  is a linear extension of  $\mathcal{P}$ , the triangles  $i$  and  $j$  of  $\mathcal{T}(c)$  are disjoint.

Our algorithm gradually refines a set of partial orders on  $\{0, 1, \mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_{9n}\}$  with the constraint that, at any time, every good order is a linear extension of at least one of these partial orders. (Note that we do not need to make  $\psi$  explicit.) Every partial order is refined until all or none of its extensions are good, so that we can report success or discard that partial order. Refinements are done in two ways:

- *branching* over an uncomparable pair, meaning duplicating the partial order and adding the comparison in one copy, and its reverse in the other copy,
- *forcing* a comparison when it is required for the formula  $S$  to be satisfiable.

We keep our algorithm as simple as possible to facilitate the verification of the algorithm, its implementation, and the geometric results proven with their aid. This comes at the cost of some efficiency, but we discuss some possible improvements in Section 5.3.

## 5.2 Description

Our poset representation stores (i) for each lifted variable the interval  $(\cdot, 0)$ ,  $(0, 1)$  or  $(1, \cdot)$  that contains it, and (ii) a directed graph over the variables contained in the interval  $(1, \cdot)$ . The graph has  $3n$  vertices, by Lemma 6. To compare two variables, we first retrieve the intervals containing them. If they differ, we can return the comparison readily. If they agree, then up to composing by  $f$  or  $f^{(2)}$  we can assume that both variables are in  $(1, \cdot)$  and we use the graph to reply. We ensure throughout that the graph is saturated, *i.e.* is its own transitive closure. In our implementation, initialization takes  $O(n^3)$  time, elements comparison takes  $O(1)$  time, and edge addition to the graph takes  $O(n^2)$  time.

We start with the poset of the comparisons forced by the tagged pattern: all pairs  $(x_i, x_j)$ ,  $(f(x_i), f(x_j))$ ,  $\dots$ ,  $(f^{(2)}(z_i), f^{(2)}(z_j))$  as well as pairs separated by 0 or 1. We next collect in a set  $U$  the comparisons missing to compute the vectors  $v(i, j)$ .

► **Lemma 8.**  $U$  contains only pairs of the form  $z_k - f(y_k)$ ,  $x_k - f(z_k)$ , or  $y_k - f(x_k)$ .

**Proof.** Every orientation predicate considered involves three points of the same index. Consider for instance  $[X_i, Y_i, Z_i, X_j]$ . Following Table 1, this decomposes into  $(x_i - x_j)(y_i - 1)(z_i - f(y_i))$  and only the sign of the last term may be undecided. Other cases are similar and show that  $U$  can only contain terms the form  $z_k - f(y_k)$ ,  $x_k - f(z_k)$ , or  $y_k - f(x_k)$ . ◀

Every pair in  $U$  corresponds to two variables *with same index*, so  $|U| \leq 3n$ . If  $U$  contains the three pairs with a given index, then two of the eight choices for these three comparisons are cyclic, and can thus be ignored. We thus have at most  $6^n$  ways to decide the order of the undetermined pairs of  $U$ ; call them *candidates*. For each candidate, we make a separate copy of our current graph and perform the following operations on that copy:

1. We add the  $|U|$  edges ordering the undecided pairs as fixed by the candidate and compute its transitive closure. We check that the result is acyclic; if not, we discard that candidate (as it makes contradictory choices) and move to the next candidate.
2. Let  $\mathcal{P}$  denote the resulting partial order. We consider every  $1 \leq i < j \leq n$  in turn. (Note that  $v(i, j)$  is determined and equal for all linear extensions of  $\mathcal{P}$ .)
  - 2a If  $v(i, j)_1 = v(i, j)_2 = v(i, j)_3$  or  $v(i, j)_4 = v(i, j)_5 = v(i, j)_6$  then triangles  $i$  and  $j$  are disjoint in  $\mathcal{P}$ . We move on to the next pair  $(i, j)$ .

- 2b** Otherwise, the extensions of  $\mathcal{P}$  in which the triangles  $i$  and  $j$  are disjoint are those in which  $[A_i, B_i, A_j, B_j] = 1$  or  $[A_i, C_i, C_j, A_j] = 1$  (in the notations of Section 2). Lemma 9 asserts that  $\mathcal{P}$  already determines at least one of these two predicates.
- 2b1** If both tests are determined to false, then triangles  $i$  and  $j$  intersect in  $\mathcal{P}$ . We then discard  $\mathcal{P}$  and move on to the next candidate.
- 2b2** If one test is determined to false and the other is undetermined, then that second test must evaluate to true in every good extension of  $\mathcal{P}$ . Again, by Table 1 we are missing exactly one comparison to decide that test. We add it to our graph.
- 2b3** In the remaining cases, at least one test is determined to true, so triangles  $i$  and  $j$  are disjoint in  $\mathcal{P}$ . We move on to the next pair  $(i, j)$ .
3. If we exhaust all  $(i, j)$  for a candidate, then we report “realizable”.
4. If we exhaust the candidates without reaching step 3, then we report “unrealizable”.

This algorithm relies on property whose computer-aided proof is discussed in Section 6:

► **Lemma 9.** *At step 2b, at least one of  $[A_i, B_i, A_j, B_j]$  or  $[A_i, C_i, C_j, A_j]$  is determined.*

### 5.3 Discussion

Let us make a few comments on our algorithm.

**Correctness.** Let  $\mathcal{P}_0$  denote the initial poset. First, remark that we explore the candidates exhaustively, so every good extension of  $\mathcal{P}_0$  is a good extension of  $\mathcal{P}_0$  augmented by (at least) one of the candidates. Next, consider the poset  $\mathcal{P}$  obtained in step 2. When processing a pair  $(i, j)$ , we either discard  $\mathcal{P}$  if we detect that  $i$  and  $j$  intersects in it (2b1) or we move on to the next pair  $(i, j)$  after having checked (2a, 2b3) or ensured (2b2) that triangles  $i$  and  $j$  are disjoint in  $\mathcal{P}$ . If we reach step 3, then all extensions of the current partial order are good and we correctly report feasibility. If a candidate is discarded then no linear extension of  $\mathcal{P}$  augmented by that candidate is a good order. If we reach Step 4, then every candidate has been discarded, so no linear extension of  $\mathcal{P}_0$  was a good order to begin with, and we correctly report unfeasibility.

**Complexity.** Initializing the poset and computing  $U$  take  $O(n^3)$  time. We have at most  $6^n$  candidates to consider. Step 1 takes  $O(n^3)$  time. The steps 2a-2b3 are executed  $O(n^2)$  times, and the bottleneck among them is 2b2, which takes  $O(n^2)$  time. Altogether, our algorithm decides if a tagged pattern is realizable in  $O(6^n n^4)$  time.

**Improvements.** In practice, the algorithm we presented can be sped up in several ways. For example, it is much better to branch over the pairs of  $U$  one by one. Once a branching is done, we can update  $U$  by removing the pairs that have become comparable, and thus avoid examining candidates that would get discarded at Step 1. Also, it pays off to record the forbidden tagged patterns of small size, and, given a larger tagged pattern to test, check first that it does not contain a small forbidden pattern.

**One-sided certificate.** If the algorithm reaches Step 3, we actually know a poset for which every linear extension is good. This means that we can compute an arbitrary linear extension to obtain an order on the variables in  $(1, \cdot)$ . We can then assign to these variables any values that satisfy this order, say by choosing the integers from 2 to  $3n + 1$ , and then propagate these values via  $f$  and  $f^{(2)}$  to all lifted variables. From there, we can extract the values of  $x_1, x_2, \dots, z_n$  of a concrete realization of our tagged pattern. In this way, all computations are done on (relatively small) rationals and are therefore easy to do exactly.

## 6 Experimental results

We now discuss our implementation of the above algorithm as well as its experimental use. Remember that we call a tagged pattern forbidden if it admits no *canonical* realization. We make the raw data available (see the full paper).

**Implementation.** We implemented the algorithm of Section 5 in Python 3. For simplicity, our implementation makes one adjustment to the algorithm: we branch over all  $2^{|U|}$  choices for the pairs of undecided variables; so, we take 8 choice per  $k$ , rather than 6. Altogether, the implementation amounts to  $\sim 470$  lines of (commented) code and is sufficiently effective for our experiments: on a standard desktop computer, finding all realizable triples of size 6 (and a realization when it exists) takes about 40 minutes, whereas verifying that no minimally forbidden tagged pattern of size 6 exists took up about a month of computer time; the difference of course is that in the former, for realizable triples we do not have to look at all positions of tags.

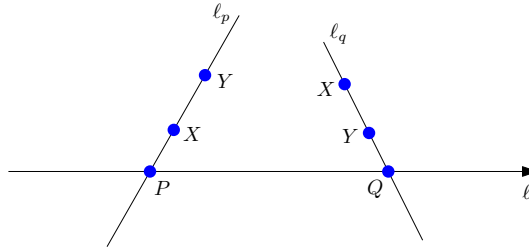
**Proof of Lemma 9.** The statement concerns only two triangles and can be shown by a simple case analysis. Our code sets up an exception that is raised if the statement of the lemma fails (cf line 94 in the code in Appendix B1 in the full version). Checking the realizability of all tagged patterns on two elements exhausts the case analysis, and the exception is not raised.

**Minimally forbidden patterns.** To state the minimally forbidden tagged patterns of size 3 we compress the notation as follows. We use  $\{uv\}$  to mean “ $uv$  or  $vu$ ”. Symbols that are omitted may be placed anywhere (this may include  $^0$  and  $^1$ ). We use  $x_i = y_j$  to mean “any pattern in which the  $i$ th symbol on the 1st tagged permutation equals the  $j$ th symbol of the 2nd tagged permutation”.

► **Proposition 10.** *The equivalence classes of minimally forbidden tagged patterns are:*

- (i) For size 2,  $(ab^0, ^1ab, ab)$ ,  $(^0ab, ab^1, ba)$ ,  $(ab^0, ba^1, ab)$ , and  $(^0ab, ^1ba, ba)$ .
- (ii) For size 3,  $(\{ab\}^0, \{ab\}c^0, z_2 = y_1)$ ,  $(\{ab\}c^0, ^1\{ab\}, z_2 = x_1)$ ,  $(\{ab\}^0, ^1c\{ab\}, z_2 = y_3)$ ,  $(^1c\{ab\}, ^1\{ab\}, z_2 = x_3)$ ,  $(abc^0, b^1ac, ca^0b)$ , and  $(^1abc, b^1ca, ac^0b)$ .
- (iii) For size 4, the taggings of  $(abcd, badc, cdab)$  that contains  $(^0b^1, ^1d, a^0)$  or  $(b^0c, ^1a, a^0)$ , and the taggings of  $(abcd, badc, dcba)$  that contains  $(b^0c, ^0d^1, ^1c)$  or  $(c^0, \{^0ba\}\{^1dc\}, ^1c)$ .
- (iv) None for size 5 and 6.

**Realization database.** For every tagged pattern that our algorithm declared realizable, we computed a realization (as explained in Section 5.3) and checked it independently.



**Geometric permutations.** It remains to prove our statements on geometric permutations:

**Proof of Theorem 1.** For every triple of permutations, we checked that it is realizable by trying all 8 reversals and all  $\binom{7}{2}^3$  possible positions of  $^0$  and  $^1$ , until we find a choice that does not contain any minimally forbidden tagged pattern of Proposition 10. ◀

**Proof of Theorem 2.** We argue by contradiction. Consider four disjoint convex sets met by lines  $l_1, l_2$  in the order  $abcd$  and  $l_3$  in the order  $badc$ ; assume that  $l_1 \odot l_3 = -l_2 \odot l_3$ . By the perturbation argument of Lemma 4, we can assume that the three lines are pairwise skew and that the convex sets are triangles with vertices on these lines. Moreover, there exists a nonsingular affine transform  $A$  that maps the unoriented lines  $l_1$  to  $l_x$ ,  $l_2$  to  $l_y$  and  $l_3$  to  $l_z$ . Remark that  $A$  either preserves or reverses all side operators. Since  $l_x \odot l_z = l_y \odot l_z$  and  $l_1 \odot l_3 = -l_2 \odot l_3$ , the map  $A$  sends the oriented lines  $(l_1, l_2)$  to either  $(l_x, -l_y)$  or  $(-l_x, l_y)$ . We used our program to check that none of  $(abcd, dcba, badc)$ ,  $(dcba, abcd, badc)$ ,  $(abcd, dcba, cdab)$ ,  $(dcba, abcd, cdab)$  admits a canonical realization. The statement follows. ◀

## 7 Geometric analysis (size two)

We present here an independent proof that the tagged patterns of size 2 listed in Proposition 10 do not have a (canonical) realization. We have a similar proof for tagged patterns of size 3 but defer it to the full paper for lack of space. We do not prove the patterns are minimal, nor do we prove that the list is exhaustive; these facts come from the completeness of our computer-aided enumeration.

The following observation was used by Asinowski and Katchalski [2]:

► **Observation 11.** *Let  $X$  and  $Y$  be compact convex sets and let  $P$  and  $Q$  be points in  $\mathbb{R}^3$ . Assume that  $X, Y, P, Q$  are pairwise disjoint and that there exist lines inducing the geometric permutation  $(PXY)$  and  $(QYX)$ . Any oriented line with direction  $\overrightarrow{PQ}$  that intersects  $X$  and  $Y$ , must intersect  $X$  before  $Y$ .*

**Proof.** Refer to the figure. Let  $h$  be a plane that separates  $X$  and  $Y$ . The existence of the geometric permutations  $(PXY)$  and  $(QYX)$  ensure that  $h$  also separates  $P$  and  $Q$ . Moreover, the halfspace bounded by  $h$  that contains  $X$  also contains  $P$ , so any line with direction  $\overrightarrow{PQ}$  traverses  $h$  from the side of  $X$  to the side of  $Y$ . ◀

Observation 11 implies that  $(ab^0, ^1ab, ab)$ ,  $(^0ab, ab^1, ba)$ ,  $(ab^0, ba^1, ab)$ , and  $(^0ab, ^1ba, ba)$ , are forbidden. Indeed, consider, by contradiction, a realization of one of these tagged patterns. Let  $P$  be the point  $^0$  on  $l_x$  and  $Q$  the point  $^1$  on  $l_y$ . In each case, we can map  $X$  and  $Y$  to  $a$  and  $b$  so that some line  $l_P \in \{l_x, -l_x\}$  realizes  $PXY$  and  $l_Q \in \{l_y, -l_y\}$  realizes

$QYX$ . Then, Observation 11 implies that any line with same direction as the line from  $P$  to  $Q$  must intersect  $X$  before  $Y$ ; this applies to the line  $\ell_z$  and contradicts the fact that the configuration realizes the chosen tagged pattern.

---

### References

- 1 B Aronov and S. Smorodinsky. On geometric permutations induced by lines transversal through a fixed point. *Discrete & Computational Geometry*, 34:285–294, 2005.
- 2 Andrei Asinowski and Meir Katchalski. Forbidden Families of Geometric Permutations in  $\mathbb{R}^d$ . *Discrete & Computational Geometry*, 34(1):1–10, 2005. doi:10.1007/s00454-004-1110-x.
- 3 O. Cheong, X. Goaoc, and H.-S. Na. Geometric permutations of disjoint unit spheres. *Computational Geometry: Theory & Applications*, 30:253–270, 2005.
- 4 Felipe Cucker, Peter Bürgisser, and Pierre Lairez. Computing the homology of basic semialgebraic sets in weak exponential time. *arXiv preprint*, 2017. arXiv:1706.07473.
- 5 James H Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1-2):29–35, 1988.
- 6 Herbert Edelsbrunner and Micha Sharir. The maximum number of ways to stab  $n$  convex non-intersecting sets in the plane is  $2n - 2$ . *Discrete & Computational Geometry*, 5:35–42, 1990.
- 7 B. Grünbaum. On common transversals. *Archiv der Mathematik*, 9:465–469, 1958.
- 8 Philippe Guigue and Olivier Devillers. Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of graphics tools*, 8(1):25–32, 2003.
- 9 Jae-Soon Ha, Otfried Cheong, Xavier Goaoc, and Jungwoo Yang. Geometric permutations of non-overlapping unit balls revisited. *Computational Geometry*, 53:36–50, 2016.
- 10 Andreas Holmsen and Rephael Wenger. Helly-type Theorems and geometric transversals. *Handbook of discrete and computational geometry*, CRC Press Ser. Discrete Math. Appl, pages 63–82, 2017.
- 11 Erich Kaltofen, Bin Li, Zhengfeng Yang, and Lihong Zhi. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. *J. Symb. Comput.*, 47(1):1–15, 2012.
- 12 M Katchalski. A conjecture of Grünbaum on common transversals. *Math. Scand.*, 59:192–198, 1986.
- 13 M. Katchalski, T. Lewis, and J. Zaks. Geometric permutations for convex sets. *Discrete Math.*, 54:271–284, 1985.
- 14 M Katchalski, Ted Lewis, and A Liu. Geometric permutations and common transversals. *Discrete & Computational Geometry*, 1(4):371–377, 1986.
- 15 Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *J. Comb. Theory, Ser. B*, 62(2):289–315, 1994.
- 16 Henri Lombardi, Daniel Perrucci, and Marie-Françoise Roy. An elementary recursive bound for effective Positivstellensatz and Hilbert 17-th problem. *arXiv preprint*, 2014. arXiv:1404.2338.
- 17 A. Marcus and G. Tardos. Excluded permutation matrices and the Stanley–Wilf conjecture. *J. Combin. Theory Ser. A*, 107:153–160, 2004.
- 18 János Pach and Micha Sharir. *Combinatorial geometry and its algorithmic applications: The Alcalá lectures*. 152. American Mathematical Soc., 2009.
- 19 James Renegar. On the computational complexity and geometry of the first-order theory of the reals. Part I: Introduction. Preliminaries. The geometry of semi-algebraic sets. The decision problem for the existential theory of the reals. *Journal of symbolic computation*, 13(3):255–299, 1992.
- 20 Natan Rubin, Haim Kaplan, and Micha Sharir. Improved bounds for geometric permutations. *SIAM Journal on Computing*, 41(2):367–390, 2012.
- 21 Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, 2017.

## 40:16 Forbidden Patterns in Geometric Permutations by Combinatorial Lifting

- 22 S. Smorodinsky, J.S.B Mitchell, and M. Sharir. Sharp bounds on geometric permutations for pairwise disjoint balls in  $\mathbb{R}^d$ . *Discrete & Computational Geometry*, 23:247–259, 2000.
- 23 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.
- 24 H. Tverberg. Proof of Grünbaum's conjecture on common transversals for translates. *Discrete & Computational Geometry*, 4:191–203, 1989.
- 25 Helge Tverberg. On geometric permutations and the Katchalski-Lewis conjecture on partial transversals for translates. *DIMACS Ser. Discrete Math. Theor. Comp. Sci.*, 6:351–361, 1991.

# Journey to the Center of the Point Set

**Sariel Har-Peled**

Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA  
<https://sarielhp.org/>  
sariel@illinois.edu

**Mitchell Jones**

Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, USA  
<http://mfjones2.web.engr.illinois.edu/>  
mfjones2@illinois.edu

---

## Abstract

---

We revisit an algorithm of Clarkson et al. [1], that computes (roughly) a  $1/(4d^2)$ -centerpoint in  $\tilde{O}(d^9)$  time, for a point set in  $\mathbb{R}^d$ , where  $\tilde{O}$  hides polylogarithmic terms. We present an improved algorithm that computes (roughly) a  $1/d^2$ -centerpoint with running time  $\tilde{O}(d^7)$ . While the improvements are (arguably) mild, it is the first progress on this well known problem in over twenty years. The new algorithm is simpler, and the running time bound follows by a simple random walk argument, which we believe to be of independent interest. We also present several new applications of the improved centerpoint algorithm.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Random walks and Markov chains

**Keywords and phrases** Computational geometry, Centerpoints, Random walks

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.41

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1712.02949>.

**Funding** *Sariel Har-Peled*: Supported in part by NSF AF awards CCF-1421231 and CCF-1217462.

## 1 Introduction

### Notation

In the following  $O(\cdot)$  hides constants that do not depend on the dimension.  $O_d(\cdot)$  hides constants that depend on the dimension (usually badly – exponential or doubly exponential, or even worse). The notation  $\tilde{O}(\cdot)$  hides polylogarithmic factors, where the power of the polylog is independent of the dimension.

### Computing centerpoints

A classical implication of Helly’s theorem, is that for any set  $P$  of  $n$  points in  $\mathbb{R}^d$ , there is a  $1/(d+1)$ -centerpoint. Specifically, given a constant  $\alpha \in (0, 1)$ , a point  $\bar{c} \in \mathbb{R}^d$  is an  $\alpha$ -centerpoint if all closed halfspaces containing  $\bar{c}$  also contain at least  $\alpha n$  points of  $P$ . It is currently unknown if one can compute a  $\Omega(1/d)$ -centerpoint in polynomial time (in the dimension). A randomized polynomial time algorithm was presented by Clarkson et al. [1], that computes (roughly) a  $1/(4d^2)$ -centerpoint in  $\tilde{O}(d^9)$  time.

### Weak $\varepsilon$ -nets

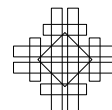
Consider the range space  $(P, \mathcal{C})$ , where  $P$  is a set of  $n$  points in  $\mathbb{R}^d$ , and  $\mathcal{C}$  is the set of all convex shapes in  $\mathbb{R}^d$ . This range space has infinite VC dimension, and as such it is impervious to the standard  $\varepsilon$ -net constructions. *Weak  $\varepsilon$ -nets* bypass this issue by using points outside the point set. While there is significant amount of work on weak  $\varepsilon$ -nets, the constructions known are not easy and result in somewhat large sets. The state of the art is the work by Matoušek



© Sariel Har-Peled and Mitchell Jones;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 41; pp. 41:1–41:14



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





and Wagner [10], which shows a weak  $\varepsilon$ -net construction of size  $O(\varepsilon^{-d}(\log \varepsilon^{-1})^{O(d^2 \log d)})$ . Rubinfeld recently improved this result for points in  $\mathbb{R}^2$ , proving the existence of weak  $\varepsilon$ -nets of size  $O(\varepsilon^{-(1.5+\gamma)})$  for arbitrarily small  $\gamma > 0$  [15]. Such a weak  $\varepsilon$ -net  $\mathcal{W}$  has the guarantee that any convex set  $C$  that contains at least  $\varepsilon n$  points of  $P$ , must contain at least one point of  $\mathcal{W}$ . See [13] for a recent survey of  $\varepsilon$ -nets and related concepts. See also the recent work by Rok and Smorodinsky [14] and references therein.

### Basis of weak $\varepsilon$ -nets

Mustafa and Ray [12] showed that one can pick a random sample  $S$  of size  $c_d \varepsilon^{-1} \log \varepsilon^{-1}$  from  $P$ , and then compute a weak  $\varepsilon$ -net for  $P$  directly from  $S$ , showing that the size of the support needed to compute a weak  $\varepsilon$ -net is (roughly) the size of a regular  $\varepsilon$ -net. Unfortunately, the constant in their sample  $c_d = O(d^d (\log d)^{cd^3 \log d})$  is doubly exponential in the dimension. This constant  $c_d$  is related to the  $((d+1)^2, d+1)$ -Hadwiger-Debrunner number (the best known upper bounds on  $(p, q)$ -Hadwiger-Debrunner numbers can be found in [7, 6]).

In particular, all current results about weak  $\varepsilon$ -nets suffer from the “curse of dimensionality” and have constants that are at least doubly exponential in the dimension.

### Our results

Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ . In addition to the improved algorithm for computing approximate centerpoints, we also suggest two alternatives to weak  $\varepsilon$ -nets as applications, and obtain some related results:

- (a) **Approximating centerpoints.** We revisit the algorithm of Clarkson et al. [1] for approximating a centerpoint. We present an improved algorithm, which is a variant of their algorithm which runs in  $\tilde{O}(d^7)$  time, and computes roughly a  $1/(d+2)^2$ -centerpoint. This improves both the running time, and the quality of centerpoint computed. While the improvements are small (a factor of  $d^2$  roughly in the running time, and a factor of four in the centerpoint quality), we believe that the new algorithm is simpler. The analysis is cleaner, and is of independent interest. In particular, the analysis uses a random walk argument, which is quite uncommon in computational geometry, and (we believe) is of independent interest. See Theorem 18. This is the first improvement of the randomized algorithm of Clarkson et al. [1] in over twenty years. Miller and Sheehy also derandomized the algorithm of Clarkson et al., computing a  $\Omega(1/d^2)$ -centerpoint in time  $n^{O(\log d)}$  [11].
- (b) **Lowerbounding convex functions.** Given a convex function  $f$  in  $\mathbb{R}^d$ , such that one can compute its value and gradient at a point efficiently, we present an algorithm that computes quickly a realizable lower-bound on the value of  $f$  over  $P$ . Formally, the algorithm computes a point  $q \in \mathbb{R}^d$ , such that  $f(q) \leq \min_{p \in P} f(p)$ . The algorithm is somewhat similar in spirit to the ellipsoid algorithm. The running time of the algorithm is  $\tilde{O}(d^9)$ . See Theorem 22.
- (c) **Functional nets.** Let  $C \subseteq \mathbb{R}^d$  be a convex body. Suppose we are only given access to  $C$  via a separation oracle: given a query point  $q$ , the oracle either returns that  $q$  is in  $C$ , or alternatively, the oracle returns a hyperplane separating  $q$  and  $C$ . We show that a random sample of size

$$O(\varepsilon^{-1} d^3 \log d \log^3 \varepsilon^{-1} + \varepsilon^{-1} \log \varphi^{-1}) = \tilde{O}(d^3/\varepsilon),$$

with probability  $\geq 1 - \varphi$ , can be used to decide if a query convex body  $C$  is  $\varepsilon$ -light. Formally, the algorithm, using only the sample, performs  $O(d^2 \log \varepsilon^{-1})$  oracle queries – if

any of the query points generated stabs  $C$ , then  $C$  is considered as (potentially) containing more than  $\varepsilon n$  points. Alternatively, if all the queries missed  $C$ , then  $C$  contains less than  $\varepsilon n$  points of  $P$ . The query points can be computed in polynomial time, and we emphasize that the dependency in the running time and sample size are polynomial in  $\varepsilon$  and  $d$ . See Theorem 28. As such, this result can be viewed as slightly mitigating the curse of dimensionality in the context of weak  $\varepsilon$ -nets.

- (d) **Center nets.** Using the above, one can also construct a weak  $\varepsilon$ -net directly from such a sample – this improves over the result of Mustafa and Ray [12] as far as the dependency on the dimension is concerned. This construction is described in Lemma 33.

Surprisingly, by using ideas from Theorem 28 one can get a stronger form of a weak  $\varepsilon$ -net, which we refer to as an  $(\varepsilon, \alpha)$ -center net. Here  $\alpha = \Omega(1/(d \log \varepsilon^{-1}))$  and one can compute a set  $\mathcal{W}$  of size (roughly)  $\tilde{O}_d(\varepsilon^{-O(d^2)})$ , such that if a convex body  $C$  contains  $\geq \varepsilon n$  points of  $P$ , then  $\mathcal{W}$  contains a point  $q$  which is an  $\alpha$ -centerpoint of  $C \cap P$ . Namely, the net contains a point that stabs  $C$  in the “middle” as far as the point set  $C \cap P$ . See Theorem 38.

## Paper organization

The improved centerpoint approximation algorithm is described in Section 3. Two applications of the improved centerpoint algorithm are presented in Section 4. The construction of center nets is described in Section 5. Background and standard tools used are described in Section 2.

## 2 Background

### 2.1 Ranges spaces, VC dimension, samples and nets

The following is a quick survey of (standard) known results about  $\varepsilon$ -nets,  $\varepsilon$ -samples, and relative approximations [2].

► **Definition 1.** A *range space*  $S$  is a pair  $(\widehat{X}, \mathcal{R})$ , where  $\widehat{X}$  is a **ground set** (finite or infinite) and  $\mathcal{R}$  is a (finite or infinite) family of subsets of  $\widehat{X}$ . The elements of  $\widehat{X}$  are **points** and the elements of  $\mathcal{R}$  are **ranges**.

For technical reasons, it will be easier to consider a finite subset  $X \subseteq \widehat{X}$  as the underlining ground set.

► **Definition 2.** Let  $S = (\widehat{X}, \mathcal{R})$  be a range space, and let  $X$  be a finite (fixed) subset of  $\widehat{X}$ . For a range  $r \in \mathcal{R}$ , its **measure** is the quantity  $\overline{m}(r) = |r \cap X|/|X|$ . For a subset  $S \subseteq X$ , its **estimate** of  $\overline{m}(r)$ , for  $r \in \mathcal{R}$ , is the quantity  $\overline{s}(r) = |r \cap S|/|S|$ .

► **Definition 3.** Let  $S = (\widehat{X}, \mathcal{R})$  be a range space. For  $Y \subseteq \widehat{X}$ , let  $\mathcal{R}_{|Y} = \{r \cap Y \mid r \in \mathcal{R}\}$  denote the **projection** of  $\mathcal{R}$  on  $Y$ . The range space  $S$  projected to  $Y$  is  $S_{|Y} = (Y, \mathcal{R}_{|Y})$ . If  $\mathcal{R}_{|Y}$  contains all subsets of  $Y$  (i.e., if  $Y$  is finite, we have  $|\mathcal{R}_{|Y}| = 2^{|Y|}$ ), then  $Y$  is **shattered** by  $\mathcal{R}$  (or equivalently  $Y$  is shattered by  $S$ ).

The **VC dimension** of  $S$ , denoted by  $\dim_{VC}(S)$ , is the maximum cardinality of a shattered subset of  $\widehat{X}$ . If there are arbitrarily large shattered subsets, then  $\dim_{VC}(S) = \infty$ .

► **Definition 4.** Let  $S = (\widehat{X}, \mathcal{R})$  be a range space, and let  $X$  be a finite subset of  $\widehat{X}$ . For  $0 \leq \varepsilon \leq 1$ , a subset  $S \subseteq X$  is an  **$\varepsilon$ -sample** for  $X$  if for any range  $r \in \mathcal{R}$ , we have  $|\overline{m}(r) - \overline{s}(r)| \leq \varepsilon$ , see Definition 2. Similarly, a set  $S \subseteq X$  is an  **$\varepsilon$ -net** for  $X$  if for any range  $r \in \mathcal{R}$ , if  $\overline{m}(r) \geq \varepsilon$  (i.e.,  $|r \cap X| \geq \varepsilon |X|$ ), then  $r$  contains at least one point of  $S$  (i.e.,  $r \cap S \neq \emptyset$ ).

A generalization of both concepts is relative approximation. Let  $p, \hat{\varepsilon} > 0$  be two fixed constants. A **relative  $(p, \hat{\varepsilon})$ -approximation** is a subset  $S \subseteq X$  that satisfies  $(1 - \hat{\varepsilon})\overline{m}(r) \leq \overline{s}(r) \leq (1 + \hat{\varepsilon})\overline{m}(r)$ , for any  $r \in \mathcal{C}$  such that  $\overline{m}(r) \geq p$ . If  $\overline{m}(r) < p$  then the requirement is that  $|\overline{s}(r) - \overline{m}(r)| \leq \hat{\varepsilon}p$ .

► **Theorem 5** ( $\varepsilon$ -net theorem, [5]). Let  $(\widehat{X}, \mathcal{R})$  be a range space of VC dimension  $\xi$ , let  $X$  be a finite subset of  $\widehat{X}$ , and suppose that  $0 < \varepsilon \leq 1$  and  $\varphi < 1$ . Let  $N$  be a set obtained by  $m$  random independent draws from  $X$ , where  $m \geq \max\left(\frac{4}{\varepsilon} \lg \frac{4}{\varphi}, \frac{8\xi}{\varepsilon} \lg \frac{16}{\varepsilon}\right)$ . Then  $N$  is an  $\varepsilon$ -net for  $X$  with probability at least  $1 - \varphi$ .

The following is a slight strengthening of the result of Vapnik and Chervonenkis [16] – see [2, Theorem 7.13].

► **Theorem 6** ( $\varepsilon$ -sample theorem). Let  $\varphi, \varepsilon > 0$  be parameters and let  $(\widehat{X}, \mathcal{R})$  be a range space with VC dimension  $\xi$ . Let  $X \subset \widehat{X}$  be a finite subset. A sample of size  $O(\varepsilon^{-2}(\xi + \log \varphi^{-1}))$  from  $X$  is an  $\varepsilon$ -sample for  $S = (X, \mathcal{R})$  with probability  $\geq 1 - \varphi$ .

► **Theorem 7** ([8, 3]). A sample  $S$  of size  $O(\hat{\varepsilon}^{-2}p^{-1}(\xi \log p^{-1} + \log \varphi^{-1}))$  from a range space with VC dimension  $\xi$ , is a relative  $(p, \hat{\varepsilon})$ -approximation with probability  $\geq 1 - \varphi$ .

The following is a standard statement on the VC dimension of a range space formed by mixing several range spaces together (see [2]).

► **Lemma 8.** Let  $S_1 = (\widehat{X}_1, \mathcal{C}_1), \dots, S_k = (\widehat{X}_k, \mathcal{C}_k)$  be  $k$  range spaces, which all have VC dimension  $\xi$ . Consider the new set of ranges  $\widehat{C} = \{r_1 \cap \dots \cap r_k \mid r_1 \in R_1, \dots, r_k \in R_k\}$ . Then the range space  $\widehat{S} = (\widehat{X}, \widehat{C})$  has VC dimension  $O(\xi k \log k)$ .

## 2.2 Weak $\varepsilon$ -nets

A convex body  $C \subseteq \mathbb{R}^d$  is  **$\varepsilon$ -heavy** (or just *heavy*) if  $\overline{m}(C) \geq \varepsilon$  (i.e.,  $|C \cap P| \geq \varepsilon|P|$ ). Otherwise,  $C$  is  **$\varepsilon$ -light**.

► **Definition 9** (Weak  $\varepsilon$ -net). Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ . A finite set  $S \subset \mathbb{R}^d$  is a **weak  $\varepsilon$ -net** for  $P$  if for any convex set  $C$  with  $\overline{m}(C) \geq \varepsilon$ , we have  $S \cap C \neq \emptyset$ .

Note, that like (regular)  $\varepsilon$ -nets, weak  $\varepsilon$ -nets have one-sided error – if  $C$  is heavy then the net must stab it, but if  $C$  is light then the net may or may not stab it.

## 2.3 Centerpoints

Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , and a constant  $\alpha \in (0, 1/(d+1)]$ , a point  $\bar{c} \in \mathbb{R}^d$  is an  **$\alpha$ -centerpoint** if for any closed halfspace that contains  $\bar{c}$ , the halfspace also contains at least  $\alpha n$  points of  $P$ . It is a classical consequence of Helly's theorem that a  $1/(d+1)$ -centerpoint always exists. If a point  $\bar{c} \in \mathbb{R}^d$  is a  $1/(d+1)$ -centerpoint for  $P$ , we omit the  $1/(d+1)$  and simply say that  $\bar{c}$  is a centerpoint for  $P$ .

## 3 Approximating the centerpoint via Radon's urn

We revisit the algorithm of Clarkson et al. [1] for approximating a centerpoint. We give a variant of their algorithm, and present a different (and we believe cleaner) analysis of the algorithm. In the process we improve the running time from being roughly  $\tilde{O}(d^9)$  to  $\tilde{O}(d^7)$ , and also improve the quality of centerpoint computed.

### 3.1 Radon's urn

#### 3.1.1 Setup

In the **Radon's urn** game there are  $r$  red balls, and  $b = n - r$  blue balls in an urn, and there is a parameter  $t$ . An iteration of the game goes as follows:

- (a) The player picks a random ball, marks it for deletion, and returns it to the urn.
- (b) The player picks a sample  $S$  of  $t$  balls (with replacement – which implies that we might have several copies of the same ball in the sample) from the urn.
- (c) If at least two of the balls in the sample  $S$  are red, the player inserts a new red ball into the urn. Otherwise, the player inserts a new blue ball.
- (d) The player returns the sample to the urn.
- (e) Finally, the player removes the ball marked for deletion from the urn.

Note that in each stage of the game, the number of balls in the urn remains the same. We are interested in how many rounds of the game one has to play till there are no red balls in the urn with high probability. Here, the initial value of  $r$  (i.e.,  $r_0$ ) is going to be relatively small compared to  $n$ .

### 3.2 Analysis

Let  $P(r)$  be the probability of picking two or more red balls into the sample, assuming that there are  $r$  red balls in the urn. We have that

$$P(r) = \sum_{i=2}^t \binom{t}{i} \left(\frac{r}{n}\right)^i \left(1 - \frac{r}{n}\right)^{t-i} \leq \binom{t}{2} \left(\frac{r}{n}\right)^2 \leq \frac{t^2}{2} \left(\frac{r}{n}\right)^2.$$

Note, that  $P(r) \leq 1/8$  if  $n \geq 2tr$ . Let  $P_+(r)$  be the probability that the number of red balls increased in this iteration. For this to happen, at least two red balls had to be in the sample, and the deleted ball must be blue. Let  $P_-(r)$  be the probability that the number of red balls decreases – the player needs to pick strictly less than two red balls in the sample, and delete a red ball. This implies

$$P_+(r) = P(r)(1 - r/n) \leq P(r) \quad \text{and} \quad P_-(r) = (1 - P(r))(r/n).$$

The probability for a change in the number of red balls at this iteration is

$$P_{\pm}(r) = P_+(r) + P_-(r) = P(r)(1 - r/n) + (1 - P(r))(r/n) = (1 - 2r/n)P(r) + r/n.$$

► **Lemma 10.** *Let  $\xi \in (0, 1/4)$  be a parameter. If  $r \leq R$ , then*

$$P_+(r) \leq \frac{1/2 - \xi}{1/2 + \xi} P_-(r), \quad \text{where} \quad R = (1 - 2\xi) \frac{n}{t^2}.$$

To simplify exposition, we choose  $\xi = 1/6$  so that  $P_+(r) \leq P_-(r)/2$ . In the remaining analysis, the proofs involving tedious calculations are given in the full version of this paper [4], with the modification that the statement of the lemmas and proofs involve the parameter  $\xi$ .

#### What are we analyzing?

The value  $R/n$  is an upper bound on the ratio of red balls that the urn can have and still, with good probability, end with zero red balls at the end of the game. If this ratio is violated anytime during the game, then the urn might end up consisting of only red balls. We want to start the game with an urn initially having close to  $R$  red balls, but still end up with an entirely blue urn with sufficiently high probability.

**The question**

Let  $\vartheta \in (0, 1)$  and  $r_0 = (1 - \vartheta)R$  be the number of red balls in the urn at the start of the game (note that both  $r_0$  and  $R$  are functions of  $n$  and  $t$ ). Let  $\varphi > 0$  be a parameter. The key question is the following: How large does  $n$  need to be so that if we start with  $r_0$  red balls (and  $n - r_0$  blue balls), the game ends with all balls being blue with probability  $\geq 1 - \varphi$ ?

**The game as a random walk**

An iteration of the game where the number of red balls changes is an **effective** iteration. Considering only the effective iterations, this can be interpreted as a random walk starting at  $X_0 = (1 - \vartheta)R$  and at every iteration either decreasing the value by one with probability at least  $2/3$ , and increasing the value with probability at most  $1/3$  (since  $P_+(r) \leq P_-(r)/2$ ). This walk ends when either it reaches 0 or  $R$ . If the walk reaches  $R$ , then the process fails. Otherwise if the walk reaches 0, then there are no red balls in the urn, as desired.

**3.2.1 Analyzing the related walk**

Consider the random walk that starts at  $Y_0 = (1 - \vartheta)R$ . In the  $i$ th iteration,  $Y_i = Y_{i-1} - 1$  with probability  $2/3$  and  $Y_i = Y_{i-1} + 1$  with probability  $1/3$ . Let  $\mathcal{Y} = Y_1, Y_2, \dots$  be the resulting random walk which stochastically dominates the walk  $X_0, X_1, \dots$ . This walk is strongly biased towards going to 0, and as such it does not hang around too long before moving on, as testified by the following lemma.

► **Lemma 11.** *Let  $I$  be any integer number, and let  $\varphi > 0$  be a parameter. The number of times the random walk  $\mathcal{Y}$  visits  $I$  is at most  $9 \ln(9/\varphi)$  times, and this holds with probability  $\geq 1 - \varphi$ .*

We next bound the probability that the walk fails.

► **Lemma 12.** *Let  $\varphi > 0$  be a parameter. If  $R \geq \frac{3}{\vartheta} \ln \frac{9}{\varphi}$  then the probability that the random walk ever visits  $R$  (and thus fails) is bounded by  $\varphi$ .*

**3.2.2 Back to the urn**

The number of red balls in the urn is stochastically dominated by the random walk above. The challenge is that the number of iterations one has to play before an effective iteration happens (thus, corresponding to one step of the above walk), depends on the number of red balls, and behaves like the coupon collector problem. Specifically, if there are  $r \leq R$  red balls in the urn, then the probability for an effective step is  $P_{\pm}(r) \geq (1 - P(r))(r/n) \geq r/2n$ , as  $P(r) \leq 1/2$ . This implies that, in expectation, one has to wait at most  $2n/r$  iterations before an effective iteration happens.

► **Lemma 13.** *Let  $\varphi > 0$  be the probability of failure. For any value  $r \leq R$ , the urn spends at most  $O((n/r) \log(\varphi^{-1}))$  regular iterations, throughout the game, having exactly  $r$  balls in it, with probability  $\geq 1 - \varphi$ .*

► **Lemma 14.** *Let  $\varphi > 0$  and  $\vartheta \in (0, 1)$  be parameters, and assume that  $n = \Omega\left(\frac{t^2}{\vartheta} \ln \frac{1}{\varphi}\right)$ . The total number of regular iterations one has to play till the urn contains only blue balls, is  $O(n \log n \log(n\varphi^{-1}))$ , and this bound holds with probability  $\geq 1 - \varphi$ .*

### 3.3 Approximating a centerpoint

#### 3.3.1 The algorithm

Before describing the algorithm, we need the following well known facts [9]:

- (a) **Radon's theorem:** Given a set  $T$  of  $d + 2$  points in  $\mathbb{R}^d$ , one can partition  $T$  into two non-empty sets  $T_1, T_2$ , such that  $\mathcal{CH}(T_1) \cap \mathcal{CH}(T_2) \neq \emptyset$ . A point  $p \in \mathcal{CH}(T_1) \cap \mathcal{CH}(T_2)$  is a **Radon point**.
- (b) Computing a Radon point can be done by solving a system of  $d + 1$  linear equalities in  $d + 2$  variables. This can be completed in  $O(d^3)$  time using Gaussian elimination.
- (c) A Radon point is a  $2/(d + 2)$ -centerpoint of  $T$ .
- (d) Let  $h^+$  be a halfspace containing only one point of  $T$ . Then, a Radon point  $p$  of  $T$  is contained in  $\mathbb{R}^d \setminus h^+$  [1].

#### The algorithm in detail

Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$  for which we would like to approximate its centerpoint. To this end, let  $Q$  be initially  $P$ . In each iteration the algorithm randomly picks  $d + 2$  points (with repetition) from  $Q$ , computes their Radon point, randomly deletes any point of  $Q$ , and inserts the new Radon point into the point set  $Q$ . The claim is that after a sufficient number of iterations, any point of  $Q$  is a  $f(d)$ -centerpoint of  $P$ , where  $f(d) = \Theta(1/d^2)$  (its exact value is specified below in Eq. (3.1)).

► **Remark 15.** The algorithm above is a variant of the algorithm of Clarkson et al. [1]. Their algorithm worked in rounds, in each round generating  $n$  new Radon points, and then replacing the point set with this new set, repeating this sufficient number of times. Our algorithm on the other hand is a “continuous” process.

#### Intuition

A Radon point is a decent center for the points defining it. Visually, the above algorithm causes the points to slowly migrate towards the center region of the original point set.

To see why this is true, pick an arbitrary halfspace  $h^+$  that contains exactly  $f(d)n$  points of  $P$ . In each iteration, only if we picked two (or more) points that are in  $Q \cap h^+$ , the new point might be in  $h^+$ . Observe that we are in the setting of Radon's urn with  $t = d + 2$ . Indeed, color all the points inside  $h^+$  as red, and all the points outside as blue. To apply the Radon's urn analysis above, we require that  $(1 - \vartheta)R = f(d)n$ , which by the choice of  $R = (1 - 2\xi)n/t^2$  in Lemma 10 (and recalling  $\xi = 1/6$ ) implies that

$$(1 - \vartheta) \frac{2n}{3(d + 2)^2} = f(d)n \iff f(d) = \frac{2(1 - \vartheta)}{3(d + 2)^2} \geq \frac{1 - \vartheta}{2(d + 2)^2} \quad (3.1)$$

where  $\vartheta \in (0, 1)$ . We can now apply the Radon's urn analysis to argue that after sufficient number of iterations, all the points of  $Q$  are outside  $h^+$ . Naturally, we need to apply this analysis to all halfspaces.

#### 3.3.2 Analysis

Consider all half-spaces that might be of interest. To this end, consider any hyperplane passing through  $d$  points of  $P$ , and translate it so that it contains on one of its sides exactly  $f(d)n$  points (naturally, there are two such translations). Each such hyperplane thus defines two natural halfspaces. Let  $H$  be the resulting set of halfspaces. Observe that

$|H| \leq 2 \binom{n}{d} \leq 2(ne/d)^d$ . If  $Q$  does not contain any point in any of the halfspaces of  $H$  then all its points are  $f(d)$ -centerpoints. In particular, one can think about this as playing  $|H|$  parallel Radon's urn games. We want the algorithm to succeed with probability  $\geq 1 - \varphi$ . Setting the probability of success for each halfplane of  $H$  to be  $\varphi/|H|$ , and by Lemma 14, we have that all of these halfspaces are empty after playing

$$O(n \log n \log(n|H|\varphi^{-1})) = O(dn \log n \log(n/\varphi))$$

iterations, with probability of success being  $1 - |H|(\varphi/|H|) = 1 - \varphi$  by the union bound. Using Lemma 14 requires that  $n = \Omega(t^2 \vartheta^{-1} \ln(|H|/\varphi)) = \Omega(\vartheta^{-1} d^3 \ln n + \vartheta^{-1} d^2 \ln \varphi^{-1})$  which holds for  $n = \Omega(\vartheta^{-1} d^3 \ln d + \vartheta^{-1} d^2 \ln \varphi^{-1})$ .

Using the fact that computing a Radon point for  $d + 2$  points in  $\mathbb{R}^d$  can be done in  $O(d^3)$  time, we obtain the following result.

► **Lemma 16.** *Let  $\varphi > 0$  and  $\vartheta \in (0, 1)$  be parameters, and let  $P$  be a set of  $n = \Omega(\vartheta^{-1} d^3 \ln d + \vartheta^{-1} d^2 \ln \varphi^{-1})$  points in  $\mathbb{R}^d$ . Let  $\alpha = \frac{1-\vartheta}{2(d+2)^2}$ . Then, one can compute a  $\alpha$ -centerpoint of  $P$  via a randomized algorithm. The running time of the randomized algorithm is  $O(d^3 \cdot dn \log n \log(n/\varphi)) = O(d^4 n \log n \log(n/\varphi))$ , and it succeeds with probability  $\geq 1 - \varphi$ .*

► **Theorem 17.** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , a parameter  $\varphi$ , and a constant  $\vartheta \in (0, 1)$ , one can compute a  $\frac{1-\vartheta}{2(d+2)^2}$ -centerpoint of  $P$ . The running time of the algorithm is  $O(\vartheta^{-3} d^7 \log^3 d \log^3 \varphi^{-1})$ , and it succeeds with probability  $\geq 1 - \varphi$ .*

**Proof.** The idea is to pick a random sample  $S$  from  $P$  that is a  $(\rho, \vartheta/8)$ -relative approximation for halfspaces, where  $\rho = 1/(10d^2)$ . This range space has VC dimension  $d + 1$ , and by Theorem 7, a sample of size

$$\mu = O(\rho^{-1} \vartheta^{-2} (d \log \rho^{-1} + \log \varphi^{-1})) = O(d^2 \vartheta^{-2} (d \log d + \log \varphi^{-1}))$$

is a  $(\rho, \vartheta/8)$ -relative approximation.

Running the algorithm of Lemma 16 on  $S$  with  $\vartheta/8$  yields a  $\tau$ -centerpoint  $\bar{c}$  of  $S$ , where  $\tau = \frac{1-\vartheta/8}{2(d+2)^2} \geq \rho$  for  $d \geq 2$ . By the relative approximation property, this is a  $(1 \pm \vartheta/8)\tau$ -centerpoint of  $P$ . Therefore  $\bar{c}$  is an  $\alpha$ -centerpoint for  $P$ , where

$$\alpha = (1 - \vartheta/8)\tau = \frac{(1 - \vartheta/8)^2}{(d + 2)^2} \geq \frac{1 - \vartheta}{(d + 2)^2}.$$

By Lemma 16, the running time of the resulting algorithm is

$$O(d^4 \mu \log \mu \log(\mu/\varphi)) = O(\vartheta^{-2} \log^2 \vartheta^{-1} d^7 \log^3 d \log^3 \varphi^{-1}) = O(\vartheta^{-3} d^7 \log^3 d \log^3 \varphi^{-1}). \blacktriangleleft$$

Now, one can repeat the above calculations with the parameter  $\xi$ . Intuitively, as  $\xi$  approaches zero, the random walk becomes less unbalanced since it will move left with probability  $1/2 + \xi$  and right with probability  $1/2 - \xi$ . Because of this, there is an increased chance that the random walk will reach  $R$  (and thus fail). In order to preserve that the random walk succeeds with probability at least  $1 - \varphi$ , the sample size  $n$  must depend on the parameter  $\xi$ . In fact, the parameter  $\xi$  allows us to compute centerpoints with quality arbitrarily close to  $1/(d + 2)^2$ .

► **Theorem 18.** *Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , a parameter  $\varphi$ , and a constant  $\gamma \in (0, 1)$ , one can compute a  $\frac{1-\gamma}{(d+2)^2}$ -centerpoint of  $P$ . The running time of the algorithm is  $O(\gamma^{-4} d^7 \log^3 d \log^3(\gamma^{-1} \varphi^{-1}))$ , and it succeeds with probability  $\geq 1 - \varphi$ .*



► **Remark 19.** The above compares favorably to the result of [1, Corollary 3] – they get a running time of  $O(d^9 \log d + d^8 \log^2 \varphi^{-1})$ , which is slower by roughly a factor of  $d^2$ , and computes a  $\frac{1}{4.08(d+2)(d+1)}$ -centerpoint of  $P$  – the quality of the centerpoint is roughly worse by a factor of four.

## 4 Application I: Algorithms with oracle access

In this section we discuss two applications of the improved centerpoint algorithm. Both applications revolve around the idea of *oracle access*. In the first application, we are interested in lower bounding a convex function given an oracle to compute its gradient. In the second, we utilize centerpoints in order to determine whether a given convex body  $C$  is  $\varepsilon$ -heavy using a separation oracle.

### 4.1 Lowerbounding a function with a gradient oracle

► **Definition 20.** Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a convex function. For a number  $c \in \mathbb{R}$ , define the **level set of  $f$**  as  $\mathcal{L}_f(c) = \{p \in \mathbb{R}^d \mid f(p) \leq c\}$ . If  $f$  is a convex function, then  $\mathcal{L}_f(c)$  is a convex set for all  $c \in \mathbb{R}$ .

► **Definition 21.** Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a convex (and possibly non-differentiable) function. For a point  $p \in \mathbb{R}^d$ , a vector  $v \in \mathbb{R}^d$  is a **subgradient** of  $f$  at  $p$  if for all  $q \in \mathbb{R}^d$ ,  $f(q) \geq f(p) + \langle v, q - p \rangle$ . The **subdifferential** of  $f$  at  $p \in \mathbb{R}^d$ , denoted by  $\partial f(p)$ , is the set of all subgradients  $v \in \mathbb{R}^d$  of  $f$  at  $p$ . When the domain of  $f$  is  $\mathbb{R}^d$  and  $f$  is convex, then  $\partial f(p)$  is a non-empty set for all  $p \in \mathbb{R}^d$ .<sup>1</sup>

► **Theorem 22.** Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a convex (possibly non-differentiable) function and  $P$  a set of  $n$  points in  $\mathbb{R}^d$ . Assume that one has access to an oracle which given  $p \in \mathbb{R}^d$  returns an arbitrary element in the subdifferential  $\partial f(p)$ . With  $O(d^2 \log n)$  queries to the oracle, one can compute a point  $q \in \mathbb{R}^d$  (not necessarily in  $P$ ) such that  $f(q) \leq \min_{p \in P} f(p)$ .

**Proof.** Let  $P_1 = P$ , and  $P_i \subseteq P$  denote the set of remaining points at the beginning of the  $i$ th iteration. In iteration  $i$ , for some constant  $c > 0$ , compute a  $(c/d^2)$ -centerpoint  $\bar{c}_i$  of  $P_i$  using Theorem 17 in time  $O(d^7 \log^3 d)$  with success probability  $1/2$ . Define  $C_i = \mathcal{L}_f(f(\bar{c}_i))$ . We now use the oracle to obtain subgradient vector  $v \in \partial f(\bar{c}_i)$ . Using  $v$ , we obtain a  $d$ -dimensional hyperplane  $h_i$  which is tangent to  $C_i$  at  $\bar{c}_i$ . Let  $h_i^+$  be the halfspace formed from  $h_i$  which contains the interior of  $C_i$ . If  $|h_i^- \cap P_i| \geq c|P_i|/d^2$ , then such an iteration is successful and we set  $P_{i+1} = P_i \setminus (h_i^- \cap P_i)$  and continue to iteration  $i + 1$ . Otherwise the iteration has failed and we repeat the  $i$ th iteration. This procedure is repeated until we reach an iteration  $\tau$  in which  $|P_\tau|$  is of constant size. At this stage, the algorithm returns the point achieving the minimum of  $\min_{1 \leq i \leq \tau} f(\bar{c}_i)$  and  $\min_{p \in P_\tau} f(p)$ . Because  $f$  is convex, the algorithm returns a point  $q$  such that  $f(q) \leq f(p)$  for all  $p \in P$ .

As for the number of queries, note that in each iteration the expected number of centerpoint calculations (and thus queries) until a successful iteration is  $O(1)$ . It remains to bound the number of successful iterations. In each successful iteration, a  $c/d^2$ -fraction of points are discarded. Therefore there are at most  $\tau$  iterations, for which  $\tau$  is the smallest number with  $(1 - c/d^2)^\tau n$  smaller than some constant. This implies  $\tau = O(d^2 \log n)$ . ◀

<sup>1</sup> Indeed, consider the epigraph of the function  $f$ ,  $S = \{(x, c) \mid x \in \mathbb{R}^d, c \in \mathbb{R}, f(x) \leq c\} \subseteq \mathbb{R}^{d+1}$ . Observe that  $S$  is a convex set. For a given point  $p \in \mathbb{R}^d$ , by the supporting hyperplane theorem, there is some hyperplane  $h$  tangent to  $S$  at the point  $(p, f(p)) \in \mathbb{R}^{d+1}$ . The normal to this tangent hyperplane  $h$  is one possible subgradient of  $f$  at  $p$ .



## 4.2 Functional nets: A weak net in the oracle model

### 4.2.1 The model, construction, and query process

#### Model

Given a convex body  $C \subseteq \mathbb{R}^d$ , we assume oracle access to it. This is a standard model in optimization. Specifically, given a query point  $q$ , the oracle either returns that  $q \in C$ , or alternatively it returns a (separating) hyperplane  $h$ , such that  $C$  lies completely on one side of  $h$ , and  $q$  lies on the other side.

Our purpose here is to precompute a small subset  $S \subseteq P$ , such that given any convex body  $C$  (with oracle access to it), one can decide if  $C$  is  $\varepsilon$ -light. Specifically, the query algorithm (using only  $S$ , and not the whole point set  $P$ ) generates an (adaptive) sequence of query points  $q_1, q_2, \dots$ , such that if any of these query points are in  $C$ , then the algorithm considers  $C$  to be heavy. Otherwise, if all the query points miss  $C$ , then the algorithm outputs (correctly) that  $C$  is light (i.e.,  $\overline{m}(C) < \varepsilon$ ).

#### Construction

Given  $P$ , the set  $S$  is a random sample from  $P$  of size

$$\mu = O(\varepsilon^{-1} d^3 \log d \log^3 \varepsilon^{-1} + \varepsilon^{-1} \log \varphi^{-1}) = \tilde{O}(d^3/\varepsilon), \quad (4.1)$$

where  $\varphi > 0$  is a prespecified parameter.

#### Query process

Given a convex body  $C$  (with oracle access to it), the algorithm starts with  $S_0 = S$ . In the  $i$ th iteration, the algorithm computes a  $\Omega(1/d^2)$ -centerpoint  $q_i$  of  $S_i$  using the algorithm of Theorem 17, with failure probability at most  $1/4$ . If the oracle returns that  $q_i \in C$ , then the algorithm returns  $q_i$  as a proof of why  $C$  is considered to be heavy. Otherwise, the oracle returns a separating hyperplane  $h_i$ , such that the open halfspace  $h_i^-$  contains  $q_i$ . Let  $S'_i = S_{i-1} \setminus h_i^-$ . If  $|S'_i| \leq (1 - \gamma) |S_{i-1}|$ , where  $\gamma = 1/16d^2$  then we set  $S_i = S'_i$  (such an iteration is called **successful**). Otherwise, we set  $S_i = S_{i-1}$ . The algorithm stops when  $|S_i| \leq \varepsilon |S| / 8$ .

### 4.2.2 Correctness

Let  $I$  be the set of indices of all the successful iterations, and consider the convex set  $C_I = \bigcap_{i \in I} h_i^+$ . The set  $C_I$  is an outer approximation to  $C$ . In particular, for an index  $j$ , let  $C_j = \bigcap_{i \in I, i \leq j} h_i^+$  be this outer approximation in the end of the  $j$ th iteration. We have that  $S_j = S \cap C_j$ .

The proofs of the following results can be found in the full version of the paper [4].

► **Lemma 23.** *There are at most  $\tau = O(d^2 \log \varepsilon^{-1})$  successful iterations. For any  $j$ , the convex polyhedron  $C_j$  is defined by the intersection of at most  $\tau$  closed halfspaces.*

Let  $\mathcal{H}^\tau$  be the set of all of convex polyhedra in  $\mathbb{R}^d$  that are formed by the intersection of  $\tau$  closed halfspaces.

► **Observation 24.** *The VC dimension of  $(\mathbb{R}^d, \mathcal{H}^\tau)$  is*

$$D = O(d\tau \log \tau) = O(d(d^2 \log \varepsilon^{-1}) \log(d^2 \log \varepsilon^{-1})) = O(d^3 (\log d) \log^2 \varepsilon^{-1}).$$

*This follows readily, as the VC dimension of the range space of points in  $\mathbb{R}^d$  and halfspaces is  $d + 1$ , and by the bound of Lemma 8 for the intersection of  $\tau$  such ranges.*

► **Lemma 25.** *The set  $S$  is a relative  $(\varepsilon/8, 1/4)$ -approximation for  $(P, \mathcal{H}^\tau)$ , with probability  $1 - \varphi$ .*

**Proof.** Using Theorem 7 with  $p = \varepsilon/8$ ,  $\hat{\varepsilon} = 1/4$ , and  $\xi = D$ , implies that a random sample of  $P$  of size

$$\begin{aligned} O(\hat{\varepsilon}^{-2} p^{-1} (\xi \log p^{-1} + \log \varphi^{-1})) &= O(\varepsilon^{-1} (D \log \varepsilon^{-1} + \log \varphi^{-1})) \\ &= O(\varepsilon^{-1} (d^3 \log d \log^3 \varepsilon^{-1} + \log \varphi^{-1})) \end{aligned}$$

is the desired relative  $(p, \hat{\varepsilon})$ -approximation with probability  $\geq 1 - \varphi$ . And this is indeed the size of  $S$ , see Eq. (4.1). ◀

► **Lemma 26.** *Given a convex query body  $C$ , the expected number of oracle queries performed by the algorithm is  $O(d^2 \log \varepsilon^{-1})$ , and the expected running time of the algorithm is  $O(d^9 \varepsilon^{-1} \text{polylog})$ , where  $\text{polylog} = O(\log(d\varepsilon^{-1}\varphi^{-1})^{O(1)})$ .*

► **Lemma 27.** *Assuming that  $S$  is the desired relative approximation, then for any query body  $C$ , if the algorithm declares that it is  $\varepsilon$ -light, then  $|C \cap P| < \varepsilon n$ .*

The above implies the following.

► **Theorem 28.** *Let  $P$  be a set of points in  $\mathbb{R}^d$ , and let  $\varepsilon, \varphi > 0$  be parameters. Let  $S$  be a random sample of  $P$  of size*

$$\mu = O(\varepsilon^{-1} d^3 \log d \log^3 \varepsilon^{-1} + \varepsilon^{-1} \log \varphi^{-1}) = \tilde{O}(\varepsilon^{-1} d^3).$$

*Then, for a given query convex body  $C$  endowed with an oracle access, the algorithm described above, which uses only  $S$ , computes a sequence of query points  $q_1, \dots, q_m$ , such that either:*

- (i) *one of the points  $q_i \in C$ , and the algorithm outputs  $q_i$  as a “proof” that  $C$  is  $\varepsilon$ -heavy, or*
- (ii) *the algorithm outputs that  $|C \cap P| < \varepsilon n$ .*

*The query algorithm has the following performance guarantees:*

- (a) *The expected number of oracle queries is  $\mathbf{E}[m] = O(d^2 \log \varepsilon^{-1})$ .*
- (b) *The algorithm itself (ignoring the oracle queries) runs in  $\tilde{O}(d^9 \varepsilon^{-1})$  time*

*The output of the algorithm is correct, for all convex bodies, with probability  $\geq 1 - \varphi$ .*

► **Remark 29.** One may hope to bound the probability of the algorithm reporting a false positive. However this is inherently not possible for any weak  $\varepsilon$ -net construction. Indeed, the algorithm can fail to distinguish between a polygon that contains at least  $\varepsilon n$  of the points of  $P$  and a polygon that contains *none* of the points of  $P$ . Consider  $n$  points  $P$  lying on a circle in  $\mathbb{R}^2$ . Choose  $\varepsilon n$  of these points on the circle, and let  $C$  be the convex hull of these points. Clearly  $C$  contains at least  $\varepsilon n$  points of  $P$ . Now, take each vertex in  $C$  and “slice” it off, forming a new polygon  $C'$  that contains no points from  $P$ . However,  $C'$  is still a large polygon and as such may contain a centerpoint during the execution of the above algorithm. Therefore our algorithm may report that  $C'$  contains a large fraction of the points, even though  $C'$  contains no points of  $P$ , and so it fails to distinguish between  $C$  and  $C'$ .

► **Remark 30.** Clarkson et al. [1] provide also a randomized algorithm that finds a  $(\frac{1}{d+1} - \gamma)$ -centerpoint with probability  $1 - \delta$  in time  $O([d\gamma^{-2} \log(d\gamma^{-1})]^{d+O(1)} \log \delta^{-1})$ . We could use this algorithm instead of Theorem 17 in the query process. Since we are computing a better quality centerpoint, the number of iterations  $\tau$  and sample size  $\mu$  would be smaller by a factor of  $d$ . Specifically,  $\tau = O(d \log \varepsilon^{-1})$  and from Lemma 8, the VC dimension of the range

## 41:12 Journey to the Center of the Point Set

space  $S = (P, \mathcal{H}^\tau)$  becomes  $D = O(d^2 \log d \log^2 \varepsilon^{-1})$ . Following the proof of Lemma 25, we can construct a sample  $S$  which is  $(\varepsilon/8, 1/4)$ -relative approximation for  $S$  with probability  $1 - \varphi$  of size

$$\mu = O(\varepsilon^{-1}(D \log \varepsilon^{-1} + \log \varphi^{-1})) = O(\varepsilon^{-1}(d^2 \log d \log^3 \varepsilon^{-1} + \log \varphi^{-1})). \quad (4.2)$$

### 5 Application II: Constructing center nets

We next introduce a strengthening of the concept of a weak  $\varepsilon$ -net. Namely, we require that there is a point  $p$  in the net which stabs an  $\varepsilon$ -heavy convex body  $C$ , and that  $p$  is also a good centerpoint for  $C \cap P$ .

► **Definition 31.** For a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , and parameters  $\varepsilon, \alpha \in (0, 1)$ , a subset  $\mathcal{W} \subseteq \mathbb{R}^d$  is an  $(\varepsilon, \alpha)$ -**center net** if for any convex shape  $C$ , such that  $|P \cap C| \geq \varepsilon n$ , we have that there is an  $\alpha$ -centerpoint of  $P \cap C$  in  $\mathcal{W}$ .

In this section we prove existence of an  $(\varepsilon, \alpha)$ -center net  $\mathcal{W}$  of size roughly  $O_d(\varepsilon^{-d^2})$ , where

$$\alpha = \frac{c_1}{(d+1) \log \varepsilon^{-1}},$$

and  $c_1 \in (0, 1)$  is some fixed constant to be specified shortly. Note that the quality of the centerpoint is worse by a factor of  $\log \varepsilon^{-1}$  than the best one can hope for.

#### 5.1 The construction

The construction of the center net will be based an algorithm for constructing a weak  $\varepsilon$ -net for  $P$ . In particular, the construction algorithm will use the following two results (the proofs can be found in the full version of the paper [4]).

► **Lemma 32.** Given a set  $P$  of  $n$  points in  $\mathbb{R}^d$ , one can compute a set  $Q$  of  $O(n^{d^2})$  points, such that for any subset  $P' \subseteq P$ , there is a  $1/(d+1)$ -centerpoint of  $P'$  in  $Q$ .

► **Lemma 33.** Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ . Let  $S$  be a random sample from  $P$  of size  $\mu = \tilde{O}(\varepsilon^{-1}d^2)$ , see Eq. (4.2) for the exact bound. Then, one can compute a set of points  $\mathcal{W}$  from  $S$ , of size

$$O(\mu^{d^2}) = O\left(\left(\varepsilon^{-1}(d^2 \log d \log^3 \varepsilon^{-1} + \log \varphi^{-1})\right)^{d^2}\right)$$

which is a weak  $\varepsilon$ -net for  $P$  with probability  $\geq 1 - \varphi$ .

► **Remark 34.** A similar construction of a weak  $\varepsilon$ -net, to the one in Lemma 33, from a small sample was described by Mustafa and Ray [12]. Their sample has exponential dependency on the dimension, so the resulting weak  $\varepsilon$ -net has somewhat worse dependency on the dimension than our construction. In any case, these constructions are inferior as far as the dependency on  $\varepsilon$ , compared to the work of Matoušek and Wagner [10], which shows a weak  $\varepsilon$ -net construction of size  $O_d(\varepsilon^{-d}(\log \varepsilon^{-1})^{O(d^2 \log d)})$ .

The idea will be to repeat the construction of the net of Lemma 33, with somewhat worse constants. Specifically, take a sample  $S$  of size  $\mu = \tilde{O}(\varepsilon^{-1}d^2)$  from  $P$ , see Eq. (4.2) for the exact bound. Next, we construct the set  $\mathcal{W}$  for  $S$ , using the result of Lemma 32. Return  $\mathcal{W}$  as the desired  $(\varepsilon, \alpha)$ -center net.

## 5.2 Correctness

The proof is algorithmic. Fix any convex  $\varepsilon$ -heavy body  $C$ , and let  $S_1 = S$  be the **active set** and let  $P_1 = C \cap P$  be the **residual set** in the beginning of the first iteration.

We now continue in a similar fashion to the algorithm of Theorem 28. In the  $i$ th iteration, the algorithm computes the  $1/(d+1)$ -centerpoint  $q_i$  of  $S_i$  (running times do not matter here, so one can afford computing the best possible centerpoint). If  $q_i$  is a  $2\alpha$ -centerpoint for  $P_i$ , then  $q_i$  is intuitively a good centerpoint for  $P$ , and the algorithm returns  $q_i$  as the desired center point. Observe that by construction,  $q_i \in \mathcal{W}$  as desired.

If not, then there exists a closed halfspace  $h_i^+$  containing  $q_i$  and at most  $2\alpha|P_i|$  points of  $P_i$ . Let

$$P_{i+1} = P_i \setminus h_i^+ \quad \text{and} \quad S_{i+1} = S_i \setminus h_i^+.$$

The algorithm now continues to the next iteration.

### Analysis

The key insight is that the active set  $S_i$  shrinks much faster than the residual set  $P_i$ . However, by construction,  $S_i$  provides a good upper bound to the size of  $P$ . Now once the upper bound provided by  $S_i$  on the size of  $P_i$  is too small, this would imply that the algorithm must have stopped earlier, and found a good centerpoint.

► **Lemma 35.** *Let  $\tau = \lceil 1 + 3(d+1) + (d+1) \log \varepsilon^{-1} \rceil$ , and  $\alpha = 1/(4\tau)$ . Assuming that  $S$  is a relative  $(\varepsilon/8, 1/4)$ -approximation for the range space  $\mathcal{S} = (P, \mathcal{H}^\tau)$ , the above algorithm stops after at most  $\tau$  iterations.*

► **Lemma 36.** *The above algorithm outputs a  $\alpha$ -centerpoint of  $P \cap C$ .*

Arguing as in Remark 30 implies the following.

► **Corollary 37.** *For the above algorithm to succeed with probability  $\geq 1 - \varphi$ , the sample  $S$  needs to be a sample of the size specified by Eq. (4.2).*

## 5.3 The result

► **Theorem 38.** *Let  $P$  be a set of  $n$  points in  $\mathbb{R}^d$ , and  $\varepsilon > 0$  be a parameter. For  $\gamma = \log(1/\varepsilon)$ , there exists a  $(\varepsilon, \Omega(1/(d\gamma)))$ -center net  $\mathcal{W}$  (which is also a weak  $\varepsilon$ -net) of  $P$  (see Definition 31). The size of the net  $\mathcal{W}$  is  $O(\mu^{d^2}) \approx O_d(\varepsilon^{-d^2})$ , where  $\mu = \tilde{O}(\varepsilon^{-1}d^2)$ , see Eq. (4.2) for the exact bound.*

**Proof.** The theorem follows readily from the above, by setting  $\varphi = 1/2$ . ◀

---

### References

- 1 K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterative Radon points. *Internat. J. Comput. Geom. Appl.*, 6:357–377, 1996. URL: <http://cm.bell-labs.com/who/clarkson/center.html>.
- 2 S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Math. Surveys & Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. doi:10.1090/surv/173.
- 3 S. Har-Peled and M. Sharir. Relative  $(p, \varepsilon)$ -Approximations in Geometry. *Discrete Comput. Geom.*, 45(3):462–496, 2011. doi:10.1007/s00454-010-9248-1.
- 4 Sariel Har-Peled and Mitchell Jones. Journey to the Center of the Point Set. *CoRR*, abs/1712.02949, 2019. arXiv:1712.02949.

- 5 D. Haussler and E. Welzl.  $\varepsilon$ -nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987. doi:10.1007/BF02187876.
- 6 C. Keller, S. Smorodinsky, and G. Tardos. Improved bounds on the Hadwiger-Debrunner numbers. *ArXiv e-prints*, December 2015. arXiv:1512.04026.
- 7 C. Keller, S. Smorodinsky, and G. Tardos. On Max-Clique for intersection graphs of sets and the Hadwiger-Debrunner numbers. In Philip N. Klein, editor, *Proc. 28th ACM-SIAM Sympos. Discrete Algs. (SODA)*, pages 2254–2263. SIAM, 2017. doi:10.1137/1.9781611974782.148.
- 8 Y. Li, P. M. Long, and A. Srinivasan. Improved Bounds on the Sample Complexity of Learning. *J. Comput. Syst. Sci.*, 62(3):516–527, 2001.
- 9 J. Matoušek. *Lectures on Discrete Geometry*, volume 212 of *Grad. Text in Math.* Springer, 2002. doi:10.1007/978-1-4613-0039-7/.
- 10 J. Matoušek and U. Wagner. New Constructions of Weak epsilon-Nets. *Discrete Comput. Geom.*, 32(2):195–206, 2004. URL: <http://www.springerlink.com/index/10.1007/s00454-004-1116-4>.
- 11 G. L. Miller and D. R. Sheehy. Approximate centerpoints with proofs. *Comput. Geom.*, 43(8):647–654, 2010. doi:10.1016/j.comgeo.2010.04.006.
- 12 N. H. Mustafa and S. Ray. Weak  $\varepsilon$ -nets have basis of size  $O(\varepsilon^{-1} \log \varepsilon^{-1})$  in any dimension. *Comput. Geom. Theory Appl.*, 40(1):84–91, 2008. doi:10.1016/j.comgeo.2007.02.006.
- 13 N. H. Mustafa and K. Varadarajan. Epsilon-approximations and epsilon-nets. *CoRR*, abs/1702.03676, 2017. arXiv:1702.03676.
- 14 A. Rok and S. Smorodinsky. Weak  $1/r$ -Nets for Moving Points. In *Proc. 32nd Int. Annu. Sympos. Comput. Geom. (SoCG)*, pages 59:1–59:13, 2016. doi:10.4230/LIPIcs.SocG.2016.59.
- 15 N. Rubin. An Improved Bound for Weak Epsilon-Nets in the Plane. In *Proc. 59th Annu. Sympos. on Found. of Comput. Sci., (FOCS)*, pages 224–235, 2018. doi:10.1109/FOCS.2018.00030.
- 16 V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

# Preprocessing Ambiguous Imprecise Points

**Ivor van der Hoog**

Department of Information and Computing Sciences, Utrecht University, The Netherlands  
i.d.vanderhoog@uu.nl

**Irina Kostitsyna**

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands  
i.kostitsyna@tue.nl

**Maarten Löffler**

Department of Information and Computing Sciences, Utrecht University, The Netherlands  
m.loffler@uu.nl

**Bettina Speckmann**

Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands  
b.speckmann@tue.nl

---

## Abstract

Let  $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$  be a set of regions and let  $X = \{x_1, x_2, \dots, x_n\}$  be an (unknown) point set with  $x_i \in R_i$ . Region  $R_i$  represents the uncertainty region of  $x_i$ . We consider the following question: how fast can we establish order if we are allowed to preprocess the regions in  $\mathcal{R}$ ? The *preprocessing model* of uncertainty uses two consecutive phases: a preprocessing phase which has access only to  $\mathcal{R}$  followed by a reconstruction phase during which a desired structure on  $X$  is computed. Recent results in this model parametrize the reconstruction time by the *ply* of  $\mathcal{R}$ , which is the maximum overlap between the regions in  $\mathcal{R}$ . We introduce the *ambiguity*  $\mathcal{A}(\mathcal{R})$  as a more fine-grained measure of the degree of overlap in  $\mathcal{R}$ . We show how to preprocess a set of  $d$ -dimensional disks in  $\mathcal{O}(n \log n)$  time such that we can sort  $X$  (if  $d = 1$ ) and reconstruct a quadtree on  $X$  (if  $d \geq 1$  but constant) in  $\mathcal{O}(\mathcal{A}(\mathcal{R}))$  time. If  $\mathcal{A}(\mathcal{R})$  is sub-linear, then reporting the result dominates the running time of the reconstruction phase. However, we can still return a suitable data structure representing the result in  $\mathcal{O}(\mathcal{A}(\mathcal{R}))$  time.

In one dimension,  $\mathcal{R}$  is a set of intervals and the ambiguity is linked to interval entropy, which in turn relates to the well-studied problem of sorting under partial information. The number of comparisons necessary to find the linear order underlying a poset  $P$  is lower-bounded by the graph entropy of  $P$ . We show that if  $P$  is an interval order, then the ambiguity provides a constant-factor approximation of the graph entropy. This gives a lower bound of  $\Omega(\mathcal{A}(\mathcal{R}))$  in all dimensions for the reconstruction phase (sorting or any proximity structure), independent of any preprocessing; hence our result is tight. Finally, our results imply that one can approximate the entropy of interval graphs in  $\mathcal{O}(n \log n)$  time, improving the  $\mathcal{O}(n^{2.5})$  bound by Cardinal et al.

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

**Keywords and phrases** preprocessing, imprecise points, entropy, sorting, proximity structures

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.42

**Related Version** Find the full version on <https://arxiv.org/abs/1903.08280>.

**Funding** *Ivor van der Hoog*: Supported by the Netherlands Organisation for Scientific Research (NWO); 614.001.504.

*Maarten Löffler*: Partially supported by the Netherlands Organisation for Scientific Research (NWO); 614.001.504.

*Bettina Speckmann*: Partially supported by the Netherlands Organisation for Scientific Research (NWO); 639.023.208.

**Acknowledgements** The authors would like to thank Jean Cardinal for insightful discussions.



© Ivor van der Hoog, Irina Kostitsyna, Maarten Löffler, and Bettina Speckmann;  
licensed under Creative Commons License CC-BY

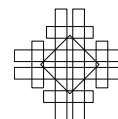
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 42; pp. 42:1–42:16

Leibniz International Proceedings in Informatics



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



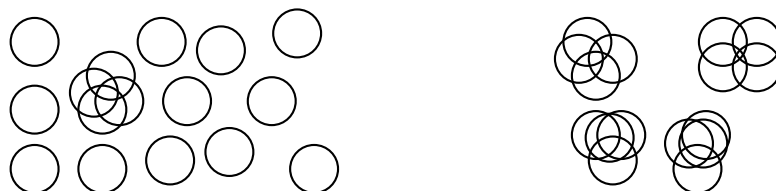
## 1 Introduction

A fundamental assumption in classic algorithms research is that the input data given to an algorithm is exact. Clearly this assumption is generally not justified in practice: real-world data tends to have (measurement or labeling) errors, heterogeneous data sources introduce yet other type of errors, and “big data” is compounding the effects. To increase the relevance of algorithmic techniques for practical applications, various paradigms for dealing with uncertain data have been introduced over the past decades. Many of these approaches have in common that they represent the uncertainty, imprecision, or error of a data point as a *disk* in a suitable distance metric which we call an uncertainty region. We focus on a fundamental problem from the realm of computation with uncertainties and errors: given a set of imprecise points represented by uncertainty regions, how much proximity information do the regions contain about the imprecise points?

**Preprocessing model.** We study this problem within the preprocessing framework initially proposed by Held and Mitchell [11]. In this framework we have a set  $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$  of regions and an point set  $X = \{x_1, x_2, \dots, x_n\}$  with  $x_i \in R_i$ . This model has 2 consecutive phases: a preprocessing phase followed by a reconstruction phase. In the preprocessing phase we have access only to  $\mathcal{R}$  and we typically want to preprocess  $\mathcal{R}$  in  $\mathcal{O}(n \log n)$  time to create some linear-size auxiliary data structure which we will denote by  $\Xi$ . In the reconstruction phase, we have access to  $X$  and we want to construct a desired output on  $X$  using  $\Xi$  faster than would be possible otherwise. Löffler and Snoeyink [18] were the first to use this model as a way to deal with data uncertainty: one may interpret the regions  $\mathcal{R}$  as *imprecise* points, and the points in  $X$  as their true (initially unknown) locations. This interpretation of the preprocessing framework has been successfully applied to various problems in computational geometry [2, 3, 6, 7, 16, 23]. Several results restrict  $\mathcal{R}$  to be a set of disjoint (unit) disks in the plane, while others consider partially overlapping disks. Traditionally, the *ply*  $\Delta(\mathcal{R})$  of  $\mathcal{R}$ , which measures the maximal number of overlapping regions, has been used to measure the degree of overlap, leading, for example, to reconstruction times of  $\mathcal{O}(n \log \Delta(\mathcal{R}))$ .

The ply is arguably a somewhat coarse measure of the degree of overlap of the regions. Consider the following example: suppose that we have a collection of  $\sqrt{n}$  disks in the plane that overlap in one point and that the remainder of  $\mathcal{R}$  is mutually disjoint (see Figure 1 left). Then  $\Delta(\mathcal{R}) = \sqrt{n}$  and the resulting time complexity of the reconstruction phase is  $\mathcal{O}(n \log n)$  even though it might be possible to achieve better bounds ( $\mathcal{R}$  is arguably not in a worst-case configuration for that given ply, see Figure 1 right).

**Ambiguity.** We introduce the *ambiguity*  $A(\mathcal{R})$  as a more fine-grained measure of the degree of overlap in  $\mathcal{R}$ . The ambiguity is based on the number of regions each individual region intersects (see Figure 1). We count this number with respect to particular permutations of



■ **Figure 1** Two sets of 16 disks each in the plane, both with a ply of 4. The ambiguity of the set on the right is four times as large as the ambiguity of the set on the left.



the regions: for each region we count only the overlap with regions that appear earlier in the permutation. A proper technical definition of ambiguity can be found in Section 2. We also show how to compute a 3-approximation of the ambiguity in  $\mathcal{O}(n \log n)$  time.

**Ambiguity and entropy.** In one dimension,  $\mathcal{R}$  is a set of intervals and the ambiguity is linked to interval (and graph) entropy (refer to the full version for a definition), which in turn relates to the well-studied problem of sorting under partial information. Fredman [9] shows that if the only information we are given about a set of values is a partial order  $P$ , and  $e(P)$  is the number of *linear extensions* (total orders compatible with) of  $P$ , then we need at least  $\Omega(\log e(P))$  comparisons to sort the values. Brightwell and Winkler prove that computing the number of linear extensions  $e(P)$  is  $\#P$ -complete [1]. Hence efforts have concentrated on computing approximations, most notably via the concept of graph entropy as introduced by Körner [14]. Specifically, Khan and Kim [13] prove that  $\log e(P) = \Theta(n \cdot H(G))$  where  $H(G)$  denotes the entropy of the *incomparability graph*  $G$  of the poset  $P$ . To the best of our knowledge there is currently no exact algorithm to compute  $H(G)$ . Cardinal et al. [4] describe the fastest known algorithm to approximate  $H(G)$ , which runs in  $\mathcal{O}(n^{2.5})$  time. Refer to the full version for a more in-depth discussion of sorting and its relation to graph entropy.

We consider the special case where the partial order is induced by uncertainty intervals. We define the entropy  $H(\mathcal{R})$  of a set of intervals as the entropy of their intersection graph (which is also an incomparability graph) using the definition of graph entropy given by Körner. In this setting we prove that the ambiguity  $A(\mathcal{R})$  provides a constant-factor approximation of the interval entropy (see Section 2). Since we can compute a constant-factor approximation of the ambiguity in  $\mathcal{O}(n \log n)$  time, we can hence also compute a constant-factor approximation of the entropy of interval graphs in  $\mathcal{O}(n \log n)$  time, thereby improving the result by Cardinal et al. [4] for this special case.

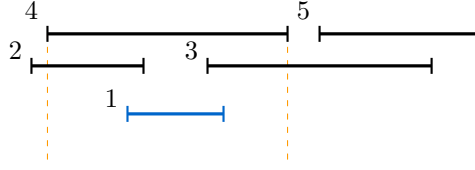
**Ambiguity and reconstruction.** Since  $\Omega(\log e(P))$  is a lower bound for the number of comparisons needed to complete  $P$  into a total order,  $\Omega(A(\mathcal{R}))$  is a lower bound for the reconstruction phase in the preprocessing model when  $\mathcal{R}$  is a set of intervals and the goal is to sort the unknown points in  $X$ . This lower bound extends to higher dimensions and to proximity structures in general, independent of any preprocessing.

The ambiguity  $A(\mathcal{R})$  ranges between 0 and  $\Theta(n \log n)$  for a set of  $n$  regions  $\mathcal{R}$ . If the value of  $A(\mathcal{R})$  lies between  $\Theta(n)$  and  $\Theta(n \log n)$  then we can preprocess  $\mathcal{R}$  in  $\mathcal{O}(n \log n)$  time and sort in  $\mathcal{O}(A(\mathcal{R}))$  time (in one dimension for arbitrary intervals) or build a quadtree in  $\mathcal{O}(A(\mathcal{R}))$  time (in all dimensions for unit disks).

If the ambiguity lies between 0 and  $\Theta(n)$ , then reporting the results explicitly in  $\Omega(n)$  time dominates the reconstruction time. But the ambiguity suggests that the information-theoretic amount of work necessary to compute the results should be lower than  $\Theta(n)$ . To capture this, we hence introduce a new variant of the preprocessing model, which allows us to return a pointer to an implicit representation of the results.

Specifically, in one dimension,  $\mathcal{R}$  is a set of intervals and we aim to return the sorted order of the unknown points in  $X$ . If, for example, all intervals are mutually disjoint, then  $A(\mathcal{R}) = \mathcal{O}(1)$  and we have essentially no time for the reconstruction phase. However, a binary search tree  $T$  on  $\mathcal{R}$ , which we can construct in  $\mathcal{O}(n \log n)$  time in the preprocessing phase, actually captures all necessary information. In the reconstruction phase we can hence return a pointer to  $T$  as an implicit representation of the sorted order. In Section 3 we show how to handle arbitrary sets of intervals in a similar manner. That is, we describe how to construct in  $\mathcal{O}(n \log n)$  time an auxiliary data structure  $\Xi$  on  $\mathcal{R}$  in the preprocessing phase (without access to  $X$ ), such that, in the reconstruction phase (using  $X$ ), we can construct a linear-size AVL-tree  $T$  on  $X$  in  $\mathcal{O}(A(\mathcal{R})) = \mathcal{O}(\log e(\mathcal{R}))$  time, which is tight.





■ **Figure 2** A set of overlapping intervals with a permutation. In this figure  $\Gamma_4^\pi = \{R_1, R_2, R_3, R_4\}$ . In all figures, bottom intervals are indicated in blue (in this case this is only  $R_1$ ).

In all dimensions, we consider  $\mathcal{R}$  to be a set of unit disks and our aim is to return a quadtree  $T$  on the points in  $X$  where each point in  $X$  lies in a unique quadtree cell. Note that in 2 dimensions,  $T$  also allows us to construct e.g. the Delaunay triangulation of  $X$  in linear time [2]. However, we show that constructing such a quadtree explicitly in  $\mathcal{O}(A(\mathcal{R}))$  time is not possible, and the work necessary to distinguish individual points could dominate the running time and overshadow the detail in the analysis brought by the ambiguity measure. We hence follow Buchin et al. [2] and use so-called  $\lambda$ -deflated *quadtrees* which contain up to a constant  $\lambda$  points in each leaf. From  $T$  one can construct a quadtree on  $X$  where each point lies in a unique quadtree cell in linear time. In Section 4 we describe how to reconstruct a linear-size  $\lambda$ -deflated quadtree  $T$  (with a suitable constant  $\lambda$ ) in  $\mathcal{O}(A(\mathcal{R})) = \mathcal{O}(\log e(\mathcal{R}))$  time, which is tight (in fact, in one dimension our result also extends to non-unit intervals).

## 2 Ambiguity

We introduce a new measure on a set of regions  $\mathcal{R}$  to reflect the degree of overlap, which we call the *ambiguity*. The sequence in which we process regions matters (refer to Section 2.1), thus we distinguish between the  $\pi$ -ambiguity defined on a given permutation of the regions in  $\mathcal{R}$ , and the minimum ambiguity defined over all possible permutations. We demonstrate several properties of the ambiguity, and discuss its relation to graph entropy when  $\mathcal{R}$  is a set of intervals in one dimension.

**Processing permutation.** Let  $\mathcal{R}$  be a set of  $n$  regions and let  $\mathcal{R}^\pi = \langle R_1, R_2, \dots, R_n \rangle$  (note that for all  $i$ , the region  $R_i$  could be any region depending on the permutation  $\pi$ ) be the sequence of elements in  $\mathcal{R}$  according to a given permutation  $\pi$ . Then we say that  $\pi$  is a *processing permutation* of  $\mathcal{R}$ . Furthermore, let  $\mathcal{R}_{\leq i}^\pi := \{R_j \mid j \leq i\}$  be the prefix of  $\mathcal{R}^\pi$ , that is, the first  $i$  elements in the sequence  $\mathcal{R}^\pi$ . A permutation  $\pi$  is *containment-compatible* if  $R_i \subset R_j$  implies  $i < j$  for all  $i$  and  $j$  [8]. When  $\pi$  is clear from context, we denote  $\mathcal{R}^\pi$  by  $\mathcal{R}$ .

**Contact set (for a permutation  $\pi$ ).** For a region  $R_i \in \mathcal{R}^\pi$  we define its *contact set*  $\Gamma_i^\pi$  to be the set of regions which precede or are equal to  $R_i$  in the order  $\pi$ , and which intersect  $R_i$ :  $\Gamma_i^\pi := \{R_j \in \mathcal{R}_{\leq i}^\pi \mid R_j \cap R_i \neq \emptyset\}$ . Note that a region is always in its own contact set. A region  $R_i$  whose contact set  $\Gamma_i^\pi$  contains only  $R_i$  itself is called a *bottom region* (refer to Figure 2).

**Ambiguity.** For a set of regions  $\mathcal{R}$  and a fixed permutation  $\pi$  we define the  $\pi$ -ambiguity  $A^\pi(\mathcal{R}) := \sum_i \log |\Gamma_i^\pi|$  (with the logarithm to the base 2). Observe that bottom regions do not contribute to the value of the  $\pi$ -ambiguity. The ambiguity of  $\mathcal{R}$  is now the minimal  $\pi$ -ambiguity over all permutations  $\pi$ ,  $A(\mathcal{R}) := \min_{\pi \in \Pi} A^\pi(\mathcal{R})$ .



$$A^{\pi_1}(\mathcal{R}) = \log 1 + \log 1 + \log 1 + \log 1 + \log 5 \quad A^{\pi_2}(\mathcal{R}) = \log 2 + \log 2 + \log 2 + \log 2 + \log 1$$

■ **Figure 3** An example of the  $\pi$ -ambiguity induced by two permutations  $\pi_1$  (on the left) and  $\pi_2$  (on the right) of the same five intervals. The  $\pi_1$ -ambiguity is  $\log 5$  and the  $\pi_2$ -ambiguity is 4.

### 2.1 Properties of ambiguity

We show the following properties of ambiguity: (1) the  $\pi$ -ambiguity may vary significantly with the choice of the processing permutation  $\pi$ , (2) in one dimension, the  $\pi$ -ambiguity for any containment-compatible permutation  $\pi$  on a set of intervals  $\mathcal{R}$  implies a 3-approximation on the entropy of the interval graph of  $\mathcal{R}$ , and (3) the permutation that realizes the ambiguity is containment-compatible. Therefore in one dimension, the ambiguity of a set of intervals  $\mathcal{R}$  implies a 3-approximation of the entropy of the interval graph of  $\mathcal{R}$ .

We start with the first property: it is easy to see that the processing permutation  $\pi$  has a significant influence on the value of the  $\pi$ -ambiguity (refer to Figure 3). Even though  $\pi$ -ambiguity can vary considerably, we show that if we restrict the permutations to be containment-compatible, their  $\pi$ -ambiguities lie within a constant factor of the ambiguity.

**Interval entropy.** The entropy of a graph  $G$  was first introduced by Körner [14]. Since then several equivalent definitions appeared [22]. We define the *interval entropy*  $H(\mathcal{R})$ , for a set of intervals  $\mathcal{R}$ , as the entropy of the intersection graph of  $\mathcal{R}$ . While investigating the question of sorting an arbitrary poset, Cardinal et al. [4] found an interesting geometrical interpretation of the poset entropy, which applies to our interval entropy: let a poset  $P$  describe a set of (open) intervals  $\mathcal{R}$  combinatorially, that is, for each  $R_i$  we know which intervals intersect  $R_i$ , are contained in  $R_i$ , contain  $R_i$ , and are disjoint from  $R_i$ . Denote by  $E(\mathcal{R})$  the infinite set of sets of intervals on the domain  $(0, 1)$  (that is, each  $\mathcal{I} \in E(\mathcal{R})$  is a set of intervals, where each interval  $I_i \in \mathcal{I}$  has endpoints in  $(0, 1)$ ) which induce the same poset as  $\mathcal{R}$ . Then Cardinal et al. prove the following lemma (see Figure 4 for an illustration):

► **Lemma 1** ([4], Lemma 3.2 paraphrased).

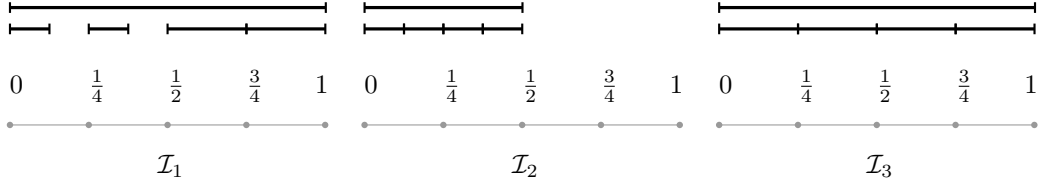
$$H(\mathcal{R}) = \log n - \min_{\mathcal{I} \in E(\mathcal{R})} \left\{ \frac{1}{n} \sum_{I_i \in \mathcal{I}} -\log |I_i| \right\}.$$

We show that the  $\pi$ -ambiguity for any containment-compatible  $\pi$  is a 3-approximation of  $n \cdot H(\mathcal{R})$ . To achieve this we rewrite the lemma from Cardinal et al. in the following way,

$$H(\mathcal{R}) = \log n - \min_{\mathcal{I} \in E(\mathcal{R})} \left\{ \frac{1}{n} \sum_{I_i \in \mathcal{I}} (\log n - \log(n|I_i|)) \right\} = \max_{\mathcal{I} \in E(\mathcal{R})} \left\{ \frac{1}{n} \sum_{I_i \in \mathcal{I}} \log(n|I_i|) \right\}.$$

An embedding  $\mathcal{I}$  gives each interval  $I_i$  a size between 0 and 1. To simplify the algebra later, we re-interpret this size as the *fraction* (weight) of the domain  $(0, 1)$  that  $I_i$  occupies. We associate with each  $\mathcal{I} \in E(\mathcal{R})$  a set of weights  $W$  such that for all  $i$ ,  $w_i = |I_i|$ ; we write  $W \sim E(\mathcal{R})$ . From now on we consider embeddings on the domain  $(0, n)$ : an interval then has a size  $n|I_i| = nw_i$ . The formula for the entropy becomes:

$$H(\mathcal{R}) = \frac{1}{n} \max_{W \sim E(\mathcal{R})} \left\{ \log \left( \prod_{w_i \in W} n w_i \right) \right\}. \tag{1}$$



■ **Figure 4** Let  $\mathcal{R}$  be a set of five intervals, where four intervals are mutually disjoint and contained in one larger interval. We show three embeddings  $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3 \in E(\mathcal{R})$  of these intervals on the domain  $(0, 1)$  with the same combinatorial properties. Embedding  $\mathcal{I}_1$  shows that  $H(\mathcal{R}) \geq \log 5 - \frac{1}{5} \log(1 \cdot \frac{1}{8} \cdot \frac{1}{8} \cdot \frac{1}{4} \cdot \frac{1}{4})$ .  $\mathcal{I}_2$  shows that  $H(\mathcal{R}) \geq \log 5 - \frac{1}{5} \log(\frac{1}{2} \cdot \frac{1}{8} \cdot \frac{1}{8} \cdot \frac{1}{8} \cdot \frac{1}{8})$  and  $\mathcal{I}_3$  is the optimal embedding which shows that  $H(\mathcal{R}) = \log 5 - \frac{1}{5} \log(1 \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4})$ .

**Ambiguity and entropy.** Next, we show that the interval entropy gives an upper bound on the ambiguity. The entropy of  $\mathcal{R}$  is the maximum over all embeddings on  $(0, n)$ , so any embedding of  $\mathcal{R}$  on the domain  $(0, n)$  gives a lower bound on  $H(\mathcal{R})$ . We will create an embedding with a corresponding weight assignment  $W$  such that:

$$A^\pi(\mathcal{R}) = \log \left( \prod_{R_i \in \mathcal{R}} |\Gamma_i^\pi| \right) \leq \log \left( \prod_{w_i \in W} (n w_i)^2 \right) \leq 2nH(\mathcal{R}). \quad (2)$$

We start with the original input embedding of  $\mathcal{R}$  and we sort the coordinates of all the endpoints (both left- and right-). To each endpoint  $p$  we assign a new coordinate  $\frac{k}{2}$  if  $p$  is the  $k$ th endpoint in the sorted order (indexing from 0). Thus, we obtain an embedding of  $\mathcal{R}$  on  $(0, n - \frac{1}{2})$ . For any containment-compatible permutation  $\pi$ , the length of each interval  $R_i$  in this embedding is at least  $\frac{1}{2}|\Gamma_i^\pi|$ , as each interval  $R_i$  contains at least  $|\Gamma_i^\pi| - 1$  endpoints of the intervals from its contact set in its interior. Also note that the distance between every right endpoint and the consecutive endpoint to the right is  $\frac{1}{2}$ . Thus, we can increase the coordinate of every right endpoint by  $\frac{1}{2}$  and obtain an embedding of  $\mathcal{R}$  on  $(0, n)$  with a corresponding weight assignment  $W$ , such that the length of each interval  $R_i$  is at least  $\frac{1}{2}(|\Gamma_i^\pi| + 1)$ . This allows us to prove the following lemma:

▶ **Lemma 2.** *For any containment-compatible permutation  $\pi$  of a set of intervals  $\mathcal{R}$ ,*

$$A^\pi(\mathcal{R}) \leq 2nH(\mathcal{R}).$$

**Proof.** Consider the embedding and corresponding weight assignment  $W$  constructed above. Consider any containment-compatible permutation  $\pi$ . We split the intervals of  $\mathcal{R}$  into four sets depending on the size of their contact set: let  $A := \{R_i \mid |\Gamma_i^\pi| = 1\}$ ,  $B := \{R_i \mid |\Gamma_i^\pi| = 2\}$ ,  $C := \{R_i \mid |\Gamma_i^\pi| = 3\}$  and  $D := \mathcal{R} \setminus \{A, B, C\}$ . Let these sets contain  $a, b, c$  and  $d$  intervals respectively. Then, using Equation (1) for the entropy,

$$\begin{aligned} 2^{nH(\mathcal{R})} &\geq \prod_{R_i \in A} \frac{|\Gamma_i^\pi| + 1}{2} \prod_{R_i \in B} \frac{|\Gamma_i^\pi| + 1}{2} \prod_{R_i \in C} \frac{|\Gamma_i^\pi| + 1}{2} \prod_{R_i \in D} \frac{|\Gamma_i^\pi| + 1}{2} \\ &\geq \left(\frac{2}{2}\right)^a \left(\frac{3}{2}\right)^b \left(\frac{4}{2}\right)^c \left(\frac{4}{2}\right)^d. \end{aligned} \quad (3)$$

On the other hand,

$$\begin{aligned} 2^{nH(\mathcal{R})} &\geq \prod_{R_i \in \mathcal{R}} \frac{|\Gamma_i^\pi| + 1}{2} \geq \prod_{R_i \in A} |\Gamma_i^\pi| \prod_{R_i \in B} \frac{3}{4} |\Gamma_i^\pi| \prod_{R_i \in C} \frac{2}{3} |\Gamma_i^\pi| \prod_{R_i \in D} \frac{1}{2} |\Gamma_i^\pi| \\ &= \left(\frac{3}{4}\right)^b \left(\frac{2}{3}\right)^c \left(\frac{1}{2}\right)^d 2^{A^\pi(\mathcal{R})}, \end{aligned}$$

as

$$\frac{|\Gamma_i^\pi| + 1}{2} \begin{cases} 1 = |\Gamma_i^\pi|, & \text{if } R_i \in A, \\ \frac{3}{2} = \frac{3}{4}|\Gamma_i^\pi|, & \text{if } R_i \in B, \\ 2 = \frac{2}{3}|\Gamma_i^\pi|, & \text{if } R_i \in C, \end{cases} \quad \frac{|\Gamma_i^\pi| + 1}{2} \geq \frac{1}{2}|\Gamma_i^\pi|, \quad \text{if } R_i \in D.$$

Then, using Equation (3) we get

$$2^{nH(\mathcal{R})} \cdot 2^{nH(\mathcal{R})} \geq \left(\frac{3}{2}\right)^b \left(\frac{4}{2}\right)^c \left(\frac{4}{2}\right)^d \cdot \left(\frac{3}{4}\right)^b \left(\frac{2}{3}\right)^c \left(\frac{1}{2}\right)^d 2^{A^\pi(\mathcal{R})} \geq 2^{A^\pi(\mathcal{R})},$$

and therefore

$$2nH(\mathcal{R}) \geq A^\pi(\mathcal{R}). \quad \blacktriangleleft$$

We continue by showing that the ambiguity also gives an upper-bound for the interval entropy. Starting with a helper lemma:

► **Lemma 3.** *Suppose  $\mathcal{R}$  is partitioned into two sets  $X$  and  $Y$  such that for each  $R \in X, R' \in Y$ ,  $R$  and  $R'$  are disjoint. In any weight assignment  $W$  that realizes  $H(\mathcal{R})$ , the intervals in  $X$  together have length  $|X|$  and the intervals in  $Y$  together have length  $|Y|$  on the domain  $(0, n)$ .*

**Proof.** In Equation (1) we rewrote the formula for entropy in terms of weights: for any weight assignment  $W \sim E(\mathcal{R})$ ,  $w_i$  is the proportion that  $R_i$  occupies on the domain, and we embedded  $\mathcal{R}$  on the domain  $(0, n)$ . We can similarly embed  $\mathcal{R}$  on the domain  $(0, \lambda)$  for an arbitrary scalar  $\lambda$ . We define the *relative entropy* of  $\mathcal{R}$  (refer to Figure 5 (top)) as:

$$H(\mathcal{R}, \lambda) := \frac{1}{n} \max_{W \sim E(\mathcal{R})} \left\{ \log \left( \prod_{w_i \in W} \lambda w_i \right) \right\}.$$

Observe that  $H(\mathcal{R}, n) = H(\mathcal{R})$  and that:

$$\forall \lambda, \mu, \quad \mu w_i = \left(\frac{\mu}{\lambda}\right) \lambda w_i \Rightarrow 2^{nH(X, \mu)} = \left(\frac{\mu}{\lambda}\right)^{|X|} 2^{nH(X, \lambda)}. \quad (4)$$

If the intervals in  $X$  can occupy a width of at most  $\lambda$ , then it is always optimal to give the intervals in  $Y$  a total width of  $n - \lambda$  (since the entropy maximizes the product of the lengths of intervals in  $X$  and  $Y$ ). This implies:

$$2^{nH(X \cup Y)} = \max_{\lambda \in [0, n]} \{2^{nH(X, \lambda)} \cdot 2^{nH(Y, n - \lambda)}\}.$$

See Figure 5 (bottom) for an illustration of the argument. If we now substitute Equation (4) into this equation we get that the maximum is realized if  $\lambda = |X|$  which proves the lemma. ◀

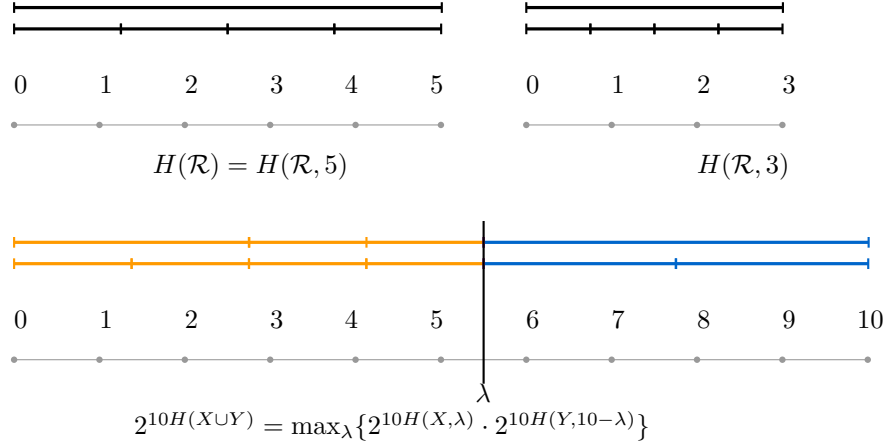
► **Lemma 4.** *Let  $\pi$  be any containment-compatible permutation, then  $nH(\mathcal{R}) \leq 3A^\pi(\mathcal{R})$ .*

**Proof.** We defined  $\mathcal{R}_{\leq i}$  as the prefix of  $\mathcal{R}$ . We prove the lemma with induction on  $i$ .

$$\text{Induction Hypothesis: } \forall j \leq i \quad jH(\mathcal{R}_{\leq j}, j) \leq 3A^\pi(\mathcal{R}_{\leq j}).$$

For  $i = 1$  both the lefthand and the righthand side are 0. So we assume that the lemma holds for all  $j \leq i$  and we prove it for  $j = i + 1$ .  $H(\mathcal{R}_{\leq i+1}, i + 1)$  is the relative entropy of  $\mathcal{R}_{\leq(i+1)}$  on the domain  $(0, i + 1)$ . We know that  $3A^\pi(\mathcal{R}_{\leq(i+1)}) = 3A^\pi(\mathcal{R}_{\leq i}) + 3 \log |\Gamma_{i+1}|$ .

42:8 Preprocessing Ambiguous Imprecise Points



■ **Figure 5** (top left) A set  $\mathcal{R}$  of five intervals and their optimal embedding for the entropy relative to  $\lambda = 5$ . (top right) The optimal embedding of  $\mathcal{R}$  for the entropy relative to  $\lambda = 3$ . Observe that the proportion that each interval obtains of the domain is the same in both embeddings. (bottom) An illustration of the argument for Lemma 3: we see a set  $X$  of 7 intervals and a set  $Y$  of 3 intervals with the intervals in  $X$  disjoint from the intervals in  $Y$ . If we vary  $\lambda$ , we vary the total width on which  $X$  and  $Y$  are embedded. The entropy is given by the maximal embedding and therefore found by optimizing  $\lambda$ .

We make a distinction between two cases:  $|\Gamma_{i+1}| = 1$  or otherwise. If  $|\Gamma_{i+1}| = 1$  then  $R_{i+1}$  is disjoint from  $R_{\leq i}$ . Lemma 3 guarantees, that if we want to embed  $R_{\leq i} \cup \{R_{i+1}\}$  on  $(0, i + 1)$  that  $R_{i+1}$  gets a size of 1. The remaining intervals get embedded with a total width of  $i$  which they already had in the previous iteration. So:

$$(i + 1)H(\mathcal{R}_{\leq(i+1)}, i + 1) = iH(\mathcal{R}_i, i) + \log 1 \leq 3A^\pi(\mathcal{R}_{\leq i}) + 3 \log |\Gamma_{i+1}| = 3A^\pi(\mathcal{R}_{\leq(i+1)}).$$

In the second case  $|\Gamma_{i+1}|$  is at least 2. The other intervals used to be optimally embedded on  $(0, i)$  and are now embedded on  $(0, i + 1)$ . So each of them expands with at most a factor  $\frac{i+1}{i}$  or algebraically:

$$\begin{aligned} nH(\mathcal{R}_{\leq(i+1)}, i + 1) &\leq nH(\mathcal{R}_{\leq i}, i) + \log \left( \left( \frac{i+1}{i} \right)^i \right) + \log((i + 1)w_{i+1}) \leq \\ &nH(\mathcal{R}_{\leq i}, i) + \log e + \log((i + 1)w_{i+1}). \end{aligned}$$

There are  $i - |\Gamma_{i+1}|$  intervals disjoint from  $R_{i+1}$  so Lemma 3 guarantees that  $(i + 1)w_{i+1} \geq |\Gamma_{i+1}| \geq 2$ . It follows that:

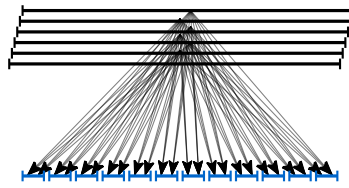
$$nH(\mathcal{R}_{\leq(i+1)}, i + 1) \leq nH(\mathcal{R}_{\leq i}, i) + 3 \log |\Gamma_{i+1}|$$

which implies the Lemma. ◀

Lemmas 2 and 4 imply the following theorem.

► **Theorem 5.** For any set of intervals  $\mathcal{R}$  in one dimension, for any containment-compatible permutation  $\pi$  on  $\mathcal{R}$ ,  $A^\pi(\mathcal{R})$  is a 3-approximation of  $nH(\mathcal{R})$ .

► **Corollary 6.** For any set of intervals  $\mathcal{R}$  in one dimension, the ambiguity  $A(\mathcal{R})$  is a 3-approximation of  $nH(\mathcal{R})$ .



■ **Figure 6** A set of intervals with a containment graph with quadratic complexity.

**Proof.** The permutation which realizes the ambiguity of  $\mathcal{R}$  must always be containment-compatible. This is because swapping a region  $R$  with a region  $R'$  that contains  $R$  in the permutation  $\pi$  always improves the  $\pi$ -ambiguity. ◀

Let  $e(\mathcal{R})$  be the number of linear extensions of the poset induced by  $\mathcal{R}$ . In the proof of Lemma 3.2 [4] Cardinal et al. show that  $\log e(\mathcal{R}) \leq nH(\mathcal{R}) \leq 2 \log e(\mathcal{R})$ . This implies that the interval graph entropy is a lower-bound for constructing any unique linear order underlying a poset. Proximity structures depend on sorting [5]. Thus, we conclude:

▶ **Theorem 7.** *Reconstructing a proximity structure on  $\mathcal{R}$  is lower-bounded by  $\Omega(A(\mathcal{R}))$ .*

### 3 Sorting

Let  $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$  be a set of intervals and let  $X = \{x_1, x_2, \dots, x_n\}$  be a set of points (values) with  $x_i \in R_i$ . We show how to construct an auxiliary structure  $\Xi$  on  $\mathcal{R}$  in the preprocessing phase without using  $X$ , such that, in the reconstruction phase, we can construct a linear-size binary search tree  $T$  on  $X$  in  $\Theta(A(\mathcal{R}))$  time. To achieve this, we first construct a specific containment-compatible permutation  $\pi$  of  $\mathcal{R}$ , and then show how to maintain  $\Xi$  when we process the intervals in this order.

#### 3.1 Level permutation

We need a processing permutation  $\pi$  of  $\mathcal{R}$  with the following conditions:

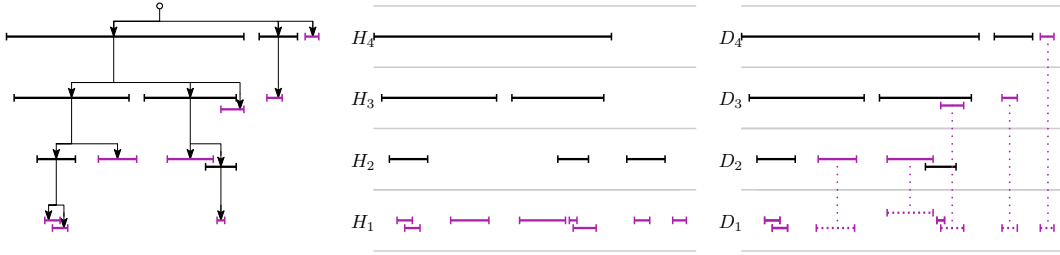
- (i)  $\pi$  is containment-compatible,
- (ii) intervals containing no interval of  $\mathcal{R}$  come first and are ordered from right to left and
- (iii) we can construct  $\pi$  in  $\mathcal{O}(n \log n)$  time.

In Section 2.1 we showed that if condition (i) holds, the  $\pi$ -ambiguity is a lower-bound for sorting  $X$ . In Section 3.2 we show that condition (ii) is useful to reconstruct an AVL-tree on  $X$  in  $\mathcal{O}(A^\pi(\mathcal{R}))$  time. Condition (iii) bounds the time used in the preprocessing phase.

Below, we define two natural partitions of  $\mathcal{R}$  based on the containment graph of  $\mathcal{R}$ : the *height partition* and the *depth partition*. However, a permutation compatible with the height partition satisfies conditions (i) and (ii) but not (iii), and a permutation compatible with the depth partition satisfies conditions (i) and (iii) but not (ii). Therefore, we define a hybrid partition, which we call the *level partition*, which implies a permutation which does satisfy all three conditions, below.

**Containment graph.** For a set of intervals  $\mathcal{R}$ , its containment graph  $G(\mathcal{R})$  represents the containment relations on  $\mathcal{R}$ .  $G(\mathcal{R})$  is a directed acyclic graph where  $R_i$  contains  $R_j$  if and only if there is a directed path from  $R_i$  to  $R_j$  and all intervals  $R \in \mathcal{R}$  that are contained in no other interval of  $\mathcal{R}$  share a common root. The bottom intervals are a subset of the leaves of this graph. Note that  $G(\mathcal{R})$  can have quadratic complexity (Figure 6).

## 42:10 Preprocessing Ambiguous Imprecise Points



■ **Figure 7** (left) A set of intervals  $\mathcal{R}$  and the corresponding containment graph  $G(\mathcal{R})$ , leaves of  $G(\mathcal{R})$  are purple. (middle) The height partition. (right) The depth partition.

**Height and Depth partition.** We define the *height partition* as the partition of  $\mathcal{R}$  into  $m$  levels  $\mathcal{H} = H_1 \dots H_m$ ,  $H_i \subseteq \mathcal{R}$  where all  $R \in H_j$  have *height* (minimal distance from  $R$  to a leaf)  $j + 1$  in  $G(\mathcal{R})$  or equivalently: the intervals in  $H_{j+1}$  contain no intervals in  $\mathcal{R} \setminus H_{\leq j}$  (Figure 7). We analogously define the *depth partition* as the partition of  $\mathcal{R}$  into  $m$  levels  $\mathcal{D} = D_1 \dots D_m$ ,  $D_i \subseteq \mathcal{R}$  where all  $R \in D_j$  have *depth* (maximal distance from the root to  $R$ )  $(m - j)$  in  $G(\mathcal{R})$ . Clearly any permutation compatible with  $\mathcal{H}$  or  $\mathcal{D}$  satisfies condition (i). All leaves of  $G(\mathcal{R})$  have height 1 so per definition are all in  $H_1$  and thus any permutation compatible with  $\mathcal{H}$  that sorts  $H_1$  satisfies condition (ii). Clearly the same is not true for  $\mathcal{D}$ . On the other hand, in Lemma 8 we show how to construct  $\mathcal{D}$  in  $\mathcal{O}(n \log n)$  time. It is unknown whether the height partition can be created in  $\mathcal{O}(n \log n)$  time (refer to the full version).

► **Lemma 8.** *For any set of intervals  $\mathcal{R}$  we can construct  $\mathcal{D}$  in  $\mathcal{O}(n \log n)$  time.*

**Proof.** We iteratively insert intervals from left to right; refer to the full version. ◀

**Level partition.** We now define the *level partition*: a hybrid between  $\mathcal{H}$  and  $\mathcal{D}$ :  $\mathcal{L} = L_1 \dots L_m$ , where all  $R \in L_j$  have depth  $(m - j)$  in  $G(\mathcal{R})$  except for the leaves of  $G(\mathcal{R})$ , which are in  $L_1$  regardless of their depth. We can compute the level partition from  $\mathcal{D}$  in  $\mathcal{O}(n \log n)$  time by identifying all leaves of  $G(\mathcal{R})$  with a range query. The *level permutation* is the permutation where intervals in  $L_i$  precede intervals in  $L_j$  and where within each level the intervals are ordered from right to left. It can be constructed from  $\mathcal{L}$  in  $\mathcal{O}(n \log n)$  time by sorting.

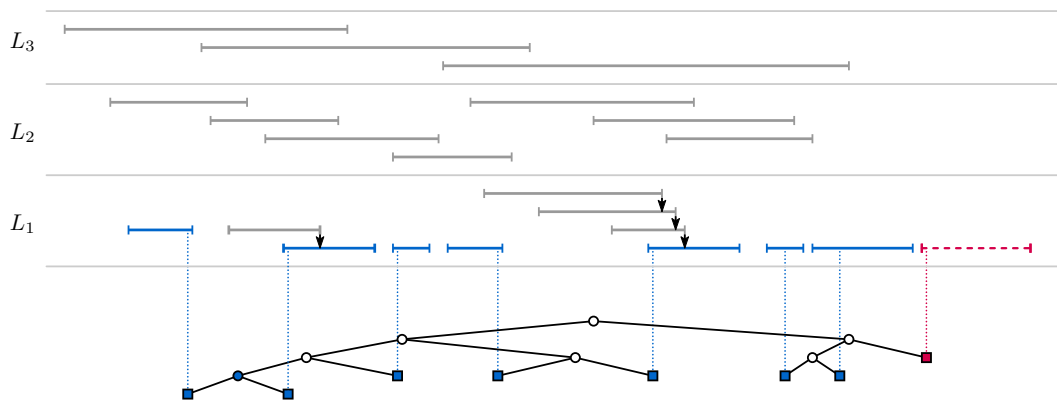
Theorem 9 follows directly from the preceding discussion.

► **Theorem 9.** *The level permutation satisfies conditions (i), (ii) and (iii).*

### 3.2 Algorithm

We continue to describe a preprocessing and reconstruction algorithm to preprocess a set of intervals  $\mathcal{R}$  in  $\mathcal{O}(n \log n)$  time such that we can sort  $X$  in  $\Theta(A(\mathcal{R}))$  time.

**Anchors.** Let  $\pi$  be the level permutation of  $\mathcal{R}$ . In the preprocessing phase we build an AVL-tree  $T$  on the bottom intervals. In the reconstruction phase, we insert each remaining  $x_i \in X$  into  $T$  in the order  $\pi$  in  $\mathcal{O}(A^\pi(\mathcal{R}))$  time. This implies that for bottom intervals we are not allowed to spend even constant time and for each non-bottom interval  $R_i$ , we want to locate  $x_i$  in  $T$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. To achieve this, we supply every non-bottom interval  $R$  with an *anchor* denoted by  $\perp^\pi(R_i)$ . For a non-bottom interval  $R \notin L_1$ , we define its anchor



■ **Figure 8** The auxiliary structure  $\Xi$ . In the level  $L_1$  all non-bottom intervals are shown their anchor. (top) A schematic representation of intervals in the level permutation  $\pi$  (from bottom to top). (bottom) The Fibonacci tree  $T$  containing the subset  $X^b$  corresponding to the bottom intervals. Note that we added one dummy node in red.

as an arbitrary interval contained in  $R$ . All intervals in  $L_1$  are ordered from right to left, so for any non-bottom interval  $R \in L_1$ , its right endpoint is contained in the interval preceding it and we make this interval the anchor of  $R$  (refer to Figure 8).

**Preprocessing phase.** The auxiliary structure  $\Xi$  is an AVL-tree  $T$  on the bottom intervals, augmented with a set of pointers leading from intervals to their anchors. We will implement  $T$  as a leaf-based AVL-tree, i.e., where values are stored in the leaves, and inner nodes are decision nodes. Finally, we will use a doubly linked list to connect the leaves of the tree.

Let  $X^b \subset X$  be the points corresponding to bottom intervals. Bottom intervals are mutually disjoint and we can build an AVL-tree  $T$  on  $X^b$  without knowing their true values. Recall that a Fibonacci tree is a tree binary where for every inner node, its left subtree has a depth 1 greater than its right subtree. A Fibonacci tree is a valid AVL-tree and we construct the AVL-tree over  $X^b$  as a Fibonacci tree where we add at most  $|X^b|$  dummy leaves with value  $\infty$  to ensure that the total number of nodes is a Fibonacci number. Refer to Figure 8 for an example. We remove the bottom intervals from  $\mathcal{R}$  and for each non-bottom interval  $R$  we identify its anchor  $\perp^\pi(R)$  and we supply  $R$  with a pointer to  $\perp^\pi(R)$ . As the final step of the preprocessing phase we connect the leaves of  $T$  in a doubly linked list. To summarize:  $\Xi$  consists of a graph of intervals connected by anchor pointers and an AVL-tree  $T$ . Each bottom interval is in  $T$  and each non-bottom interval has a directed path to a node in  $T$ .

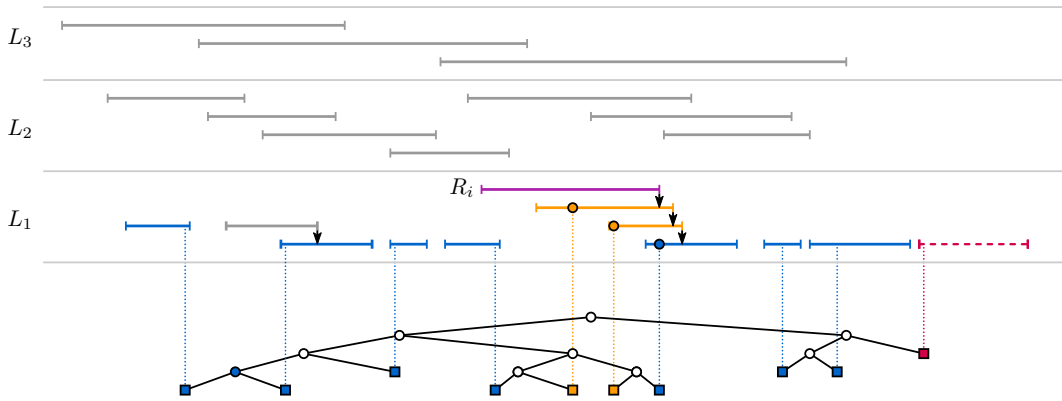
► **Lemma 10.** *We can construct the auxiliary structure  $\Xi$  in  $\mathcal{O}(n \log n)$  time.*

**Proof.** The level partition and permutation can be constructed in  $\mathcal{O}(n \log n)$  time and with it we get access to the intervals in  $L_1$  sorted from right to left. We scan  $L_1$  from right to left and for each interval  $R \in L_1$  we either identify it as a bottom interval or to supply it with its anchor. We identify for each  $R \notin L_1$  its anchor in logarithmic time using a range query. We construct the Fibonacci tree on  $X^b$  with leaf pointers in  $\mathcal{O}(n \log n)$  time [20]. ◀

**Reconstruction phase.** During the reconstruction phase, we need to maintain the balance of  $T$  when we insert new values.  $T$  contains bottom intervals which we are not allowed to change even constant time, so the classical amortized-constant analysis [19] of AVL-trees does not immediately apply. Nonetheless we show in the full version:



## 42:12 Preprocessing Ambiguous Imprecise Points



■ **Figure 9** The tree  $T$  from Figure 8 after two iterations in the reconstruction phase. We inserted the true values of the two orange intervals. Note that an orange interval requested the true value of a bottom interval. At this iteration we want to insert the point  $x_i$  of  $R_i$  into  $T$ .  $R_i$  is a non-bottom interval in  $W_1$  so its anchor must be the interval preceding it.

► **Lemma 11.** *Let  $T$  be an AVL-tree where each inner node has two subtrees with a depth difference of 1. We can dynamically maintain the balance of  $T$  in amortized  $\mathcal{O}(1)$  time.*

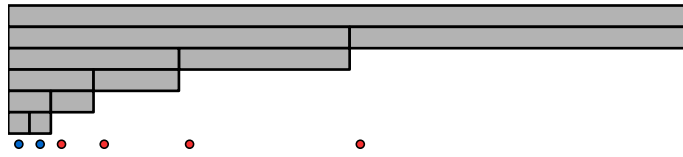
► **Theorem 12.** *Given  $\Xi$ , we can reconstruct an AVL-tree on  $X$  in  $\Theta(A^\pi(\mathcal{R}))$  time.*

**Proof.** Given  $\Xi$  and the level permutation  $\pi$  we want to sort the points in  $X$  (insert them into  $T$ ) in  $\mathcal{O}(A^\pi(\mathcal{R}))$  time. Because  $T$  starts as a Fibonacci tree, Lemma 11 guarantees that we can dynamically maintain the balance of  $T$  with at most  $\mathcal{O}(A^\pi(\mathcal{R}))$  operations. The bottom intervals are already in  $T$ , thus we need to insert only the remaining  $x_i \in X \setminus X^b$ , in the order  $\pi$ , into  $T$  in  $\log |\Gamma_i^\pi|$  time plus some additional time which we charge to the anchor (each anchor will only get charged once).

Whenever we process a non-bottom interval  $R_i$  we know that its anchor is already inserted in  $T$ . By construction, there are at most  $\mathcal{O}(|\Gamma_i^\pi|)$  leaves in  $T$  which have coordinates on the domain of  $R_i$  (because these values can come only from intervals in the contact set of  $R_i$ ). We know that we must insert  $x_i$  next to one of these  $\mathcal{O}(|\Gamma_i^\pi|)$  leaves in  $T$ . This means that if we have a pointer to any leaf on the domain of  $R_i$ , then we locate  $x_i$  in  $T$  with at most  $\mathcal{O}(\log |\Gamma_i^\pi|)$  edge traversals. During these traversals, we collapse each interval we encounter to a point. We obtain such a pointer from  $\perp^\pi(R_i)$ . Assume  $\perp^\pi(R_i) \subset R_i$ . Then the leaf corresponding to  $\perp^\pi(R_i)$  must lie on the domain of  $R_i$ . Otherwise,  $R_i$  and  $\perp^\pi(R_i)$  are both in the level  $L_1$  (illustrated in Figure 9) and  $\perp^\pi(R_i) = R_{i-1}$  and must contain the right endpoint of  $R_i$ . With a similar analysis,  $R_{i-1}$  can locate the right endpoint of  $R_i$  in  $T$  in  $\mathcal{O}(\log |\Gamma_{i-1}^\pi|)$  time. In both cases we found a leaf of  $T$  in  $R_i$  and locate  $x_i$  in  $T$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. Each interval in  $L_1$  has a unique anchor, so each anchor in  $L_1$  is charged this extra work once. ◀

## 4 Quadrees

Let  $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$  be a set of unit intervals in a bounding box (interval)  $\mathcal{B}$  (we discuss how to extend the approach later) and let  $X = \{x_1, x_2, \dots, x_n\}$  be a set of points (values) with  $x_i \in R_i$ . We show how to construct an auxiliary structure  $\Xi$  on  $\mathcal{R}$  in the preprocessing phase without using  $X$ , such that, in the reconstruction phase, we can construct a linear-size quadtree  $T$  on  $X$  in  $\Theta(A(\mathcal{R}))$  time. We recall several standard definitions.



■ **Figure 10** A set of points  $\mathcal{R}$  where the quadtree on  $\mathcal{R}$  has linear depth. If the blue points lie very close, the quadtree on  $\mathcal{R}$  needs unbounded complexity.

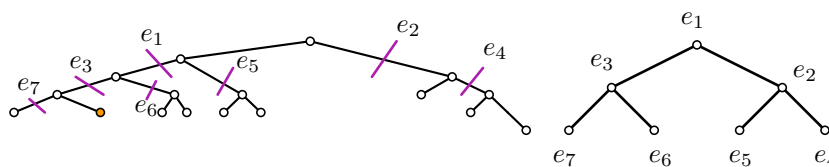
**Point quadtrees.** Suppose that we have a  $d$ -dimensional point set  $X$  in a bounding hypercube  $\mathcal{B}$ . A quadtree on  $(\mathcal{B}, X)$  is defined as follows: *split* operator is an operator that splits any  $d$ -dimensional hypercube into  $2^d$  equal-sized hypercubes called *cells*. We recursively split  $\mathcal{B}$  until each point  $p \in P$  lies within a unique cell [21]. A  $\lambda$ -deflated quadtree is a more relaxed quadtree where  $\mathcal{B}$  is split until each leaf cell contains at most  $\lambda$  points [3].

**Region quadtrees.** Let  $\mathcal{R}$  be a set of  $d$ -dimensional disks in a bounding hypercube  $\mathcal{B}$ . Let  $\mathcal{T}(\mathcal{B})$  be the infinite set of possible quadtree cells on  $\mathcal{B}$ . For each  $R_i \in \mathcal{R}$ , we define its *storing cell* denoted by  $C_i$  as the largest cell in  $\mathcal{T}(\mathcal{B})$  that is contained in  $R_i$  and contains the center of  $R_i$  [17].  $T_i$  is the subtree induced by  $C_i$ . The *neighborhood* of  $R_i$  is the set of possible cells  $C \in \mathcal{T}(\mathcal{B})$  with size  $|C_i|$  that are intersected by  $R_i$ . We consider the quadtree  $T$  on  $\mathcal{R}$  to be the unique compressed quadtree where for each  $R_i \in \mathcal{R}$ , its neighborhood is in  $T$ .

**Edge oracle tree.** Depending on  $\mathcal{B}$  and  $X$ , the quadtree on  $(\mathcal{B}, X)$  does not necessarily have logarithmic depth (Figure 10) thus, point location in  $T$  is non-trivial. Har-Peled [10] introduced a fast point-location structure (later dubbed *edge-oracle tree* [17]) for any quadtree  $T$ . The edge-oracle tree  $E$  is created through *centroid decomposition*. Any tree with bounded degree  $\delta$  has at least one *centroid edge* which separates a tree of  $n$  nodes into two trees with at least  $\frac{n}{\delta}$  and at most  $n - \frac{n}{\delta}$  nodes each. Moreover, one of these 2 trees is a *subtree* of  $T$  (a tree induced by a node as a root). For any subtree  $T'$  of  $T$ , we define its *corresponding node* in  $E$  (edge in  $T$ ) as the lowest node in  $E$  which splits  $T$  into two parts, one of which contains  $T'$  and the other contains the root of  $T$ . This node must exist, is unique and the subtree containing  $T'$  has  $\mathcal{O}(|T'|)$  nodes (refer to Figure 11).

Given a query point  $q$ , we can find the leaf cell  $C_q$  that contains  $q$  in the following way: each decision node  $v$  of  $E$  has 2 children where 1 child node  $w$  corresponds to a subtree  $T_w$  of  $T$ . We test whether  $q$  is contained in  $w$  in  $\mathcal{O}(1)$  time by checking the bounding box of  $T_w$ .

We wish to preprocess  $\mathcal{R}$  such that we can reconstruct a linear-size  $\lambda$ -deflated quadtree  $T$  for  $X$  with pointers between leaves. However,  $T$  does not necessarily have linear size and dynamically maintaining pointers between leaves is non-trivial. To achieve this, one needs to maintain a *compressed* and *smooth* quadtree  $T$  (refer to the full version for details) and Hoog et al. [12] show how to dynamically maintain a smooth compressed quadtree with constant update time. We will build such a quadtree augmented with an edge-oracle tree initialized as a Fibonacci tree. We proceed analogously to the approach in Section 3.



■ **Figure 11** (left) A tree  $T$  with recursive centroid edges. (right) The corresponding edge-oracle tree  $E$ . The orange leaf is a subtree of  $T$  and its corresponding node in  $E$  is  $e_3$ .

### 4.1 1-dimensional quadtrees on unit-size intervals

We show how to construct an auxiliary structure  $\Xi$  on  $\mathcal{R}$  without using  $X$ , such that we can construct a 2-deflated quadtree  $T$  on  $(\mathcal{B}, X)$  in  $\Theta(A(\mathcal{R}))$  time.

**Preprocessing phase.** The auxiliary structure  $\Xi$  will be a smooth compressed quadtree  $T$  on the intervals  $\mathcal{R}$  augmented with an edge-oracle tree  $E$  on  $T$ , anchor pointers, and a containment-compatible processing permutation  $\pi$  of  $\mathcal{R}$ . Given  $T$ , we initialize  $E$  as a Fibonacci tree, possibly adding dummy leaves<sup>1</sup>. We supply each  $R_i$  with a pointer to the node in  $E$  corresponding to  $T_i$  and we call this its *anchor*  $\perp^\pi(R_i)$ .

► **Lemma 13.** *The auxiliary structure  $\Xi$  can be constructed in  $\mathcal{O}(n \log n)$  time.*

**Proof.** Hoog et al. [12] show that for any set of  $d$ -dimensional disks  $\mathcal{R}$ , its smooth compressed quadtree  $T$  on  $\mathcal{R}$  with corresponding edge-oracle tree  $E$  can be constructed in  $\mathcal{O}(n \log n)$  time and that this tree has a worst-case constant update time. We turn  $E$  into a Fibonacci tree by inserting at most  $\mathcal{O}(n)$  dummy leaves in  $\mathcal{O}(n \log n)$  time in total. ◀

**Reconstruction phase.** By construction, each leaf in  $T$  intersects at most 2 bottom intervals of  $\mathcal{R}$  (since these are mutually disjoint). Therefore, we can construct a 2-deflated quadtree on  $X$  by inserting each  $x_i \in X \setminus X^b$  in the order  $\pi$  into  $T$ . We observe the following:

► **Lemma 14.** *When we process an interval  $R_i \in \mathcal{R}$ ,  $R_i$  intersects  $\mathcal{O}(|\Gamma_i^\pi|)$  leaf cells of  $T$ .*

**Proof.** There can be at most 2 bottom intervals (left and right) of  $R_i$  whose neighborhood intersects  $R_i$ . All the other leaves on the domain of  $R_i$  are caused by either already processed points on the domain of  $R_i$  or are dummy nodes. For each dummy node there is a corresponding non-dummy node also on the domain of  $R_i$ . ◀

► **Lemma 15.** *When we process an interval  $R_i$ , we can locate, for any point  $q \in R_i$ , the leaf  $C_q \in T$  which contains  $q$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time.*

**Proof.** If  $C_q \in T_i$  then  $R_i$  has an anchor to  $T_i$  and from this anchor we locate  $C_q$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. Suppose  $C_q$  is to the left of  $T_i$ . We locate the left-most leaf of  $T_i$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time and traverse its neighbor pointer. The neighboring cell must lie in a subtree  $T_q$  neighboring  $T_i$  with  $\mathcal{O}(|\Gamma_i^\pi|)$  nodes and this tree must contain  $C_q$  (Lemma 14). We now have a pointer to a node in  $T_q$  and from this node we locate  $C_q$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. ◀

► **Theorem 16.** *Given  $\Xi$ , we can construct a 2-deflated quadtree on  $X$  in  $\Theta(A^\pi(\mathcal{R}))$  time.*

**Proof.** Given  $\Xi$  and any containment-compatible permutation  $\pi$ , we want to insert  $X$  into  $T$  in  $\Theta(A^\pi(\mathcal{R}))$  time. An insertion in  $T$  creates 2 additional leaves in  $T$  (and therefore also in  $E$ ) and Lemma 11 guarantees that we can dynamically maintain the balance of  $E$  with at most  $\mathcal{O}(A^\pi(\mathcal{R}))$  operations. If we only consider the point set  $X^b \subset X$  corresponding to the bottom intervals then  $T$  is already a 2-deflated quadtree on  $X^b$  independent of where the points of  $X^b$  lie in their uncertainty intervals. Therefore, we only need to insert the remaining  $x_i \in X \setminus X^b$ , in the order  $\pi$ , into  $T$  in  $\log |\Gamma_i^\pi|$  time (potentially collapsing some of the bottom intervals when necessary). Using Lemma 15 we can locate the quadtree leaf  $C_{x_i}$  that contains  $x_i$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. This leaf is intersected by at most 2 bottom intervals, which we collapse into points whose location we locate in constant time using the leaf pointers. Thus each non-bottom interval inserts at most 3 points into  $T$  in  $\mathcal{O}(\log |\Gamma_i^\pi|)$  time. ◀

<sup>1</sup> We may need to allow parents of leaves of  $T$  to have a single dummy leaf.

## 4.2 Generalization

If we stay in one dimension, then the result of Theorem 16 in fact generalizes to the case where  $\mathcal{R}$  is a set of arbitrary intervals since Lemma 14 and 15 do not depend on the intervals being unit size. However, the result also generalizes to the case where  $\mathcal{R}$  is a set of unit-size disks in  $d$  (constant) dimensions: first of all, any permutation of  $\mathcal{R}$  is containment-compatible. If the disks are unit size then each disk intersects at most  $K_d$  bottom disks where  $K_d$  is the kissing number so Lemma 14 generalizes. For any disk  $R_i \in \mathcal{R}$ , recall that  $T_i$  was the subtree of the storing cell of  $R_i$ . Any point  $q \in R_i$  must lie in the *perimeter* of  $T_i$  which consists of at most  $\mathcal{O}(5^d)$  subtrees of size  $\mathcal{O}(|\Gamma_i^\pi|)$  therefore, Lemma 15 also generalizes. The result is even more general: this approach works for any collection  $\mathcal{R}$  of unit-size *fat* convex regions similar to, e.g. [2]. Interestingly, generalizing the result of Theorem 16 both to higher dimensions and to non-unit regions at the same time is not possible: in the full version we show that, independent of preprocessing, reconstructing a  $\lambda$ -deflated quadtree has a lower bound of  $\Omega(\log n)$ , which could be more than  $A(\mathcal{R})$ .

## 5 Conclusion

We introduced the ambiguity  $A(\mathcal{R})$  of a set of regions  $\mathcal{R}$  as a more fine-grained measure of the degree of their overlap. We applied this concept to uncertainty regions representing imprecise points. In the preprocessing model we show that the ambiguity is a natural lower bound for the time complexity of the reconstruction of any proximity structure. We achieved these results via a link to the entropy of partial orders which is of independent interest. If the regions are intervals in 1D we show how to sort in  $\Theta(A(\mathcal{R}))$  time, if the regions are unit balls in any dimension we show how to reconstruct quadtrees  $\Theta(A(\mathcal{R}))$  time.

In the future we plan to investigate if our results can be generalized to other proximity structures such as Delaunay triangulations, minimum spanning trees, and convex hulls. In principle it is possible to convert quadtrees into all of these structures in linear time [15]. However, it is not clear how to do so, when working with an implicit representation of the results in the case that  $A(\mathcal{R})$  is sub-linear.

---

## References

- 1 Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.
- 2 Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Delaunay triangulation of imprecise points simplified and extended. *Algorithmica*, 61:674–693, 2011. doi:10.1007/s00453-010-9430-0.
- 3 Kevin Buchin and Wolfgang Mulzer. Delaunay triangulations in  $O(\text{sort}(n))$  time and more. *Journal of the ACM (JACM)*, 58(2):6, 2011.
- 4 Jean Cardinal, Samuel Fiorini, Gwenaël Joret, Raphaël M Jungers, and J Ian Munro. Sorting under partial information (without the ellipsoid algorithm). *Combinatorica*, 33(6):655–697, 2013.
- 5 Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Introduction*. Springer, 2008.
- 6 Olivier Devillers. Delaunay triangulation of imprecise points, preprocess and actually get a fast query time. *Journal of Computational Geometry*, 2(1):30–45, 2011.
- 7 Esther Ezra and Wolfgang Mulzer. Convex hull of points lying on lines in  $o(n \log n)$  time after preprocessing. *Computational Geometry*, 46(4):417–434, 2013.
- 8 P.C. Fishburn and W.T. Trotter. Geometric containment orders: a survey. *Order*, 15:167–182, 1998.

- 9 Michael L Fredman. How good is the information theory bound in sorting? *Theoretical Computer Science*, 1(4):355–361, 1976.
- 10 Sariel Har-Peled. *Geometric approximation algorithms*. Number 173 in Mathematical Surveys and Monographs. American Mathematical Soc., 2011.
- 11 Martin Held and Joseph SB Mitchell. Triangulating input-constrained planar point sets. *Information Processing Letters*, 109(1):54–56, 2008.
- 12 Ivor Hoog vd, Elena Khramtcova, and Maarten. Löffler. Dynamic Smooth Compressed Quadtrees. In *LIPICs-Leibniz International Proceedings in Informatics*, volume 99. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
- 13 Jeff Kahn and Jeong Han Kim. Entropy and sorting. *Journal of Computer and System Sciences*, 51(3):390–399, 1995.
- 14 János Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *6th Prague conference on information theory*, pages 411–425, 1973.
- 15 Maarten Löffler and Wolfgang Mulzer. Triangulating the square and squaring the triangle: quadtrees and Delaunay triangulations are equivalent. *SIAM Journal on Computing*, 41(4):941–974, 2012.
- 16 Maarten Löffler and Wolfgang Mulzer. Unions of onions: Preprocessing imprecise points for fast onion decomposition. *Journal of Computational Geometry*, 5:1–13, 2014.
- 17 Maarten Löffler, Joseph A Simons, and Darren Strash. Dynamic planar point location with sub-logarithmic local updates. In *Workshop on Algorithms and Data Structures*, pages 499–511. Springer, 2013.
- 18 Maarten Löffler and Jack Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Computational Geometry*, 43(3):234–242, 2010.
- 19 Kurt Mehlhorn and Athanasios Tsakalidis. An amortized analysis of insertions into AVL-trees. *SIAM Journal on Computing*, 15(1):22–33, 1986.
- 20 Jürg Nievergelt and Edward M Reingold. Binary search trees of bounded balance. *SIAM journal on Computing*, 2(1):33–43, 1973.
- 21 Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.
- 22 Gábor Simonyi. Graph entropy: a survey. *Combinatorial Optimization*, 20:399–441, 1995.
- 23 Marc Van Kreveld, Maarten Löffler, and Joseph SB Mitchell. Preprocessing imprecise points and splitting triangulations. *SIAM Journal on Computing*, 39(7):2990–3000, 2010.

# Rods and Rings: Soft Subdivision Planner for $\mathbb{R}^3 \times S^2$

**Ching-Hsiang Hsu**

Department of Computer Science, Courant Institute, New York University, New York, NY, USA  
chhsu@nyu.edu

**Yi-Jen Chiang**

Department of Computer Science and Engineering, Tandon School of Engineering, New York University, Brooklyn, NY, USA  
chiang@nyu.edu

**Chee Yap**

Department of Computer Science, Courant Institute, New York University, New York, NY, USA  
yap@cs.nyu.edu

---

## Abstract

We consider path planning for a rigid spatial robot moving amidst polyhedral obstacles. Our robot is either a rod or a ring. Being axially-symmetric, their configuration space is  $\mathbb{R}^3 \times S^2$  with 5 degrees of freedom (DOF). Correct, complete and practical path planning for such robots is a long standing challenge in robotics. While the rod is one of the most widely studied spatial robots in path planning, the ring seems to be new, and a rare example of a non-simply-connected robot. This work provides rigorous and complete algorithms for these robots with theoretical guarantees. We implemented the algorithms in our open-source Core Library. Experiments show that they are practical, achieving near real-time performance. We compared our planner to state-of-the-art sampling planners in OMPL [31].

Our subdivision path planner is based on the twin foundations of  $\varepsilon$ -exactness and soft predicates. Correct implementation is relatively easy. The technical innovations include subdivision atlases for  $S^2$ , introduction of  $\Sigma_2$  representations for footprints, and extensions of our feature-based technique for “opening up the blackbox of collision detection”.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Computing methodologies  $\rightarrow$  Robotic planning

**Keywords and phrases** Algorithmic Motion Planning, Subdivision Methods, Resolution-Exact Algorithms, Soft Predicates, Spatial Rod Robots, Spatial Ring Robots

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.43

**Related Version** Full version [16] hosted on arXiv as [arXiv:1903.09416](https://arxiv.org/abs/1903.09416) [cs.CG], available at <https://arxiv.org/abs/1903.09416>. Also available at <http://cse.poly.edu/chiang/rod-ring18.pdf>.

**Funding** Supported in part by NSF Grants #CCF-1423228 and #CCF-1563942.

*Ching-Hsiang Hsu*: Supported by NSF Grant #CCF-1423228.

*Chee Yap*: Supported in part by NSF Grants #CCF-1423228 and #CCF-1563942.

## 1 Introduction

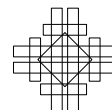
Motion planning [18, 5] is a fundamental topic in robotics because the typical robot is capable of movement. Such algorithms are increasingly relevant with the current surge of interest in inexpensive commercial mobile robots, from domestic robots that vacuum the floor to drones that deliver packages. We focus on what is called **path planning** which, in its elemental form, asks for a collision-free path from a start to a goal position, assuming a known environment. Path planning is based on robot kinematics and collision-detection only, and the variety of such problems are surveyed in [14]. The output of a “path planner” is either a path or a



© Ching-Hsiang Hsu, Yi-Jen Chiang, and Chee Yap;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 43; pp. 43:1–43:17  
Leibniz International Proceedings in Informatics

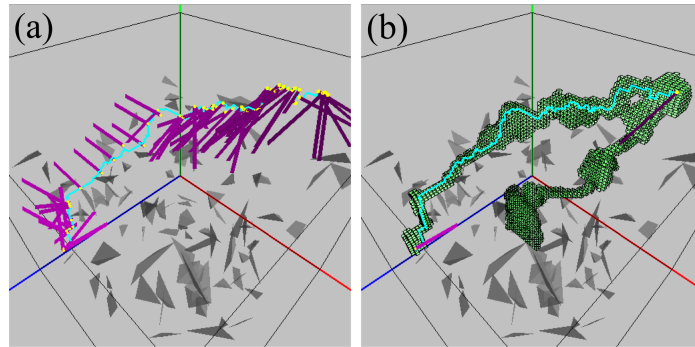


LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



NO-PATH, signifying that no path exists. Remarkably, the single bit of information encoded by NO-PATH is often missing in discussions. The standard definitions of correctness for path planners (**resolution completeness** and **probabilistic completeness**) omit this bit [32]. The last 30 years have seen a flowering of practical path planning algorithms. The dominant algorithmic paradigm of these planners has been variants of the **Sampling Approach** such as PRM, EST, RRT, SRT, etc (see [5, p. 201]). Because this bit of information is not built into the specification of such algorithms, it has led to non-termination issues and a large literature addressing the “narrow passage problem” (e.g., [22, 8]). Our present paper is based on the **Subdivision Approach**. This approach has a venerable history in robotics – see [3, 40] for early planners based on subdivision.

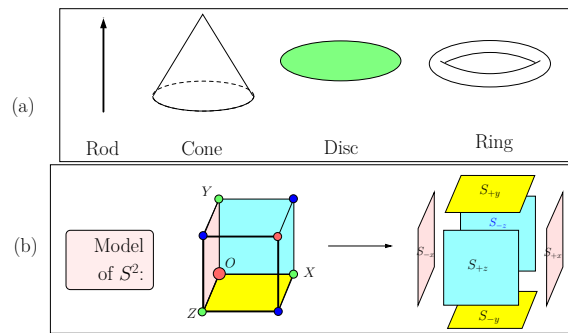
Exact path planning has many issues including a serious gap between theory and implementability. In [32, 36], we introduced a theoretical framework based on subdivision to close this gap. This paper demonstrates for the first time that our framework is able to achieve rigorous state-of-the-art planners in 3D. Figure 1 shows our rod robot in an environment with 100 random tetrahedra. Figure 6 shows our ring robot in an environment with pillars and L-shaped posts. See a video demo from [http://cs.nyu.edu/exact/gallery/rod-ring/rod\\_ring.html](http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html).



■ **Figure 1** Rod robot amidst 100 random tetrahedra: (a) trace of a found path; (b) subdivision of translational boxes on the path.

In this paper, we consider a rigid spatial robot  $R_0$  that has an axis of symmetry. See Figure 2(a) for several possibilities for  $R_0$ : rod (“ladder”), cone (“space shuttle”), disc (“frisbee”) and ring (“space station”). Our techniques easily allow these robots to be “thickened” by Minkowski sum with balls (see [33]). The **configuration space** may be taken to be  $C_{space} = \mathbb{R}^3 \times S^2$  where  $S^2$  is the unit 2-sphere. We identify  $R_0$  with a closed subset of  $\mathbb{R}^3$ , called its “canonical footprint”. E.g., if  $R_0$  is a rod (resp., ring), then the canonical footprint is a line segment (resp., circle) in  $\mathbb{R}^3$ . Each configuration  $\gamma \in C_{space}$  corresponds to a rotated translated copy of the canonical footprint, which we denote by  $Fp(\gamma)$ . Path planning involves another input, the **obstacle set**  $\Omega \subseteq \mathbb{R}^3$  that the robot must avoid. We assume that  $\Omega$  is a closed polyhedral set. Say  $\gamma$  is **free** if  $Fp(\gamma) \cap \Omega$  is empty. The **free space** comprising all the free configurations is an open set by our assumptions, and is denoted  $C_{free} = C_{free}(\Omega)$ . A parametrized continuous curve  $\mu : [0, 1] \rightarrow C_{space}$  is called a **path** if the range of  $\mu$  is in  $C_{free}$ . Path planning amounts to finding such paths. Following [40], we need to classify boxes  $B \subseteq C_{space}$  into one of three types: **FREE**, **STUCK** or **MIXED**. Let  $C(B)$  denote the classification of  $B$ :  $C(B) = \text{FREE}$  if  $B \subseteq C_{free}$ , and  $C(B) = \text{STUCK}$  if  $B$  is in the interior of  $C_{space} \setminus C_{free}$ . Otherwise,  $C(B) = \text{MIXED}$ . One of our goals is to introduce classifications  $\tilde{C}(B)$  that are “soft versions” of  $C(B)$  (see the full paper [16] Appendix A).





■ **Figure 2** (a) 3D rigid robots with  $C_{space} = \mathbb{R}^3 \times S^2$ . (b) Subdivision atlas for  $S^2$  via  $\widehat{S}^2$ .

We present four desiderata in path planning:

- (G0) the planner must be mathematically rigorous and complete;
- (G1) it must have correct implementations which are also:
- (G2) relatively easy to achieve and
- (G3) practically efficient.

In (G0), we use the standard Computer Science notion of an algorithm being **complete** if (a) it is partially complete<sup>1</sup> and (b) it halts. The notions of **resolution completeness** and **probabilistic completeness** in robotics have requirement (a) but not (b). In probabilistic-complete algorithms, halting with NO-PATH is achieved heuristically by putting limits on time and/or number of samples. But such limits are not intrinsic to the input instance. In resolution-complete algorithms, NO-PATH halting is based on width  $w$  of subdivision box being small enough (say  $w < \epsilon$ ). One issue is that the width of a box is a direct measure of clearance (but there is a nontrivial correlation); secondly, box predicates are numerical and “accurate enough” ( $\sigma$ -effective in our theory). These issues are exacerbated when algorithms do not use box predicates, but perform sampling at grid points of the subdivision. In contrast, our NO-PATH guarantees an intrinsic property: there is no path of clearance  $K\epsilon$  (see below).

But desideratum (G0) is only the base line. A (G0)-planner may not be worth much in a practical area like robotics unless it also has implementations with properties (G1-G3). E.g., the usual exact algorithms satisfy (G0) but their typical implementations fail (G1). With proper methods [30], it is possible to satisfy (G1); Halperin et al [13] give such solutions in 2D using CGAL. Both (G0) and (G1) can be formalized (see next), but (G2) and (G3) are informal. The robotics community has developed various criteria to evaluate (G2) and (G3). The accepted practice is having an implementation (proving (G2)) that achieves “real time” performance on a suite of non-trivial instances (proving (G3)).

The main contribution of this paper is the design of planners for spatial robots with 5 DOFs that have the “good” properties (G0-G3). This seems to be the first for such robots. To achieve our results, we introduce theoretical innovations and algorithmic techniques that may prove to be more widely applicable.

In path planning and in Computational Geometry, there is a widely accepted interpretation of desideratum (G0): it is usually simply called “exact algorithms”. But to stress our interest in alternative notions of exactness, we refer to the standard notion as **exact (unqualified)**. Planners that are exact (unqualified) are first shown in [26]; this can be viewed as a fundamental result on decidability of connectivity in semi-algebraic sets [1]. The curse of

<sup>1</sup> Partial completeness means the algorithm produces a correct output *provided* it halts.



exact (unqualified) algorithms is that the algorithm must detect any degeneracies in the input and handle them explicitly. But exact (unqualified) algorithms are rare, mainly because degeneracies are numerous and hard to analyze: the usual expedient is to assume “nice” (non-degenerate) inputs. So the typical exact (unqualified) algorithms in the literature are **conditional** algorithms, i.e., its correctness is conditioned on niceness assumptions. Such gaps in exact (unqualified) algorithms are not an issue as long as they are not implemented. For non-linear problems beyond 2D, complete degeneracy analysis is largely non-existent. This is vividly seen in the fact that, despite long-time interest, there is still no exact (unqualified) algorithm for the Euclidean Voronoi diagram of a polyhedral set (see [15, 12, 11, 34]). For similar reasons, unconditional exact (unqualified) path planners in 3D are unknown.

We now address (G1-G3). The typical implementation is based on machine arithmetic (the IEEE standard), which may satisfy (G2) but almost certainly not (G1). We regard this as a (G1-G2) trade-off. In fact, our implementations here as well as in our previous papers [32, 20, 33] are such machine implementations. This follows the practice in the robotics community, in order to have a fair comparison against other implementations. Below, we shall expand on our claims about (G1-G3) including how to achieve theoretically correct implementation (G1). What makes this possible is our replacement of “exact (unqualified)” planners by “exact (up to resolution)” planners, defined below:

**Resolution-Exact Path Planning** for robot  $R_0$ :  
**Input:**  $(\alpha, \beta, \Omega; B_0, \varepsilon)$   
 where  $\alpha, \beta \in C_{space}(R_0)$  is the start and goal,  $\Omega \subseteq \mathbb{R}^3$   
 the obstacle set,  $B_0 \subseteq C_{space}(R_0)$  is a box, and  $\varepsilon > 0$ .  
**Output:** Halt with either an  $\Omega$ -free path from  $\alpha$  to  $\beta$  in  $B_0$ ,  
 or **NO-PATH** satisfying the conditions (P) and (N) below.

The **resolution-exact planner** (or,  $\varepsilon$ -exact planner) has an **accuracy constant**  $K > 1$  (independent of input) such that its output satisfies two conditions:

- (P) If there is a path (from  $\alpha$  to  $\beta$  in  $B_0$ ) of clearance  $K\varepsilon$ , the output *must* be<sup>2</sup> a path.
- (N) If there is no path in  $B_0$  of clearance  $\varepsilon/K$ , the output *must* be **NO-PATH**.

Here, clearance of a path is the minimum separation of the obstacle set  $\Omega$  from the robot’s footprint on the path. Note that the preconditions for (P) and (N) are not exhaustive: in case the input fails both preconditions, our planner may either output a path or **NO-PATH**. This indeterminacy is essential to escape from exact computation (and arguably justified for robotics [36]). The constant  $K > 1$  is treated in more detail in [32, 37]. But resolution-exactness is just a definition. How do we design such algorithms? We propose to use subdivision, and couple with **soft predicates** to exploit resolution-exactness. We replace the classification  $C(B)$  by a soft version  $\tilde{C}(B)$  [32]. This leads to a general resolution-exact planner which we call **Soft Subdivision Search** (SSS) [36, 37] that shares many of the favorable properties of sampling planners (and unlike exact planners). We demonstrated in [32, 20, 33] that for planar robots with up to 4 DOFs, our planners can consistently outperform state-of-the-art sampling planners.

## 1.1 What is New: Contributions of This Paper

In this work, we design  $\varepsilon$ -exact planners for rods and rings, with accompanying implementation that addresses the desiderata (G0-G3). This fulfills a long-time challenge in robotics. We are able to do this because of the twin foundations of resolution-exactness and soft-predicates.

<sup>2</sup> For simplicity, we do not require the output path to have any particular clearance, but we could require clearance  $\geq \varepsilon/K$  as in [32].

Although we had already used this foundation to implement a variety of planar robots [32, 20, 33, 39] that can match or surpass state-of-the-art sampling methods, it was by no means assured that we can extend this success to 3D robots. Indeed, the present work required a series of technical innovations: **(I)** One major technical difference from our previous work on planar robots is that we had to give up the notion of “forbidden orientations” (which seems “forbidding” for 3D robots). We introduced an alternative approach based on the “safe-and-effective” approximation of footprint of boxes. We then show how to achieve such approximations for the rod and ring robots separately. **(II)** The approximated footprints of boxes are represented by what we call  $\Sigma_2$ -sets (Sec. 4.1); this representation supports desideratum (G2) for easy implementation. One side benefit of  $\Sigma_2$ -sets is that they are very flexible; thus, we can now easily extend our planners to “thick” versions of the rod or ring. In contrast, the forbidden orientation approach requires non-trivial analysis to justify the “thick” version [33]. The trade-off in using  $\Sigma_2$ -sets is a modest increase in the accuracy constant  $K$ . **(III)** We also need good representations of the 5-DOF configuration space. Here we introduce the square model of  $S^2$  to avoid the singularities in the usual spherical polar coordinates [19], **and also** to support subdivision in non-Euclidean spaces. **(IV)** Not only is the geometry in 3D more involved, but the increased degree of freedom requires new techniques to further improve efficiency. Here, the search heuristic based on Voronoi diagrams becomes critical to achieve real-time performance (desideratum (G3)).

## Overview of the Paper

Section 2 is a brief literature review. Section 3 explains an essential preliminary to doing subdivision in  $S^2$ . Sections 4–6 describe our techniques for computing approximate footprints of rods and rings. We discuss efficiency and experimental results in Section 7. We conclude in Section 8. All proofs and some background are given in the full paper [16] (in appendices).

## 2 Literature Review

Halperin et al [14] gave a general survey of path planning. An early survey is [35] where two universal approaches to exact path planning were described: cell-decomposition [25] and retraction [24, 23, 4]. Since exact path planning is a semi-algebraic problem [26], it is reducible to general (double-exponential) cylindrical algebraic decomposition techniques [1]. But exploiting path planning as a connectivity problem yields singly-exponential time (e.g, [10]). The case of a planar rod (called “ladder”) was first studied in [25] using cell-decomposition. More efficient (quadratic time) methods based on the retraction method were introduced in [28, 29]. On-line versions for a planar rod are also available [7, 6].

Spatial rods were first treated in [27]. The combinatorial complexity of its free space is  $\Omega(n^4)$  in the worst case and this can be closely matched by an  $O(n^{4+\epsilon})$  time algorithm [17]. The most detailed published planner for a 3D rod is Lee and Choset [19]. They use a retraction approach. The paper exposes many useful and interesting details of their computational primitives (see its appendices). In particular, they follow a Voronoi edge by a numerical path tracking. But like most numerical code, there is no a priori guarantee of correctness. Though the goal is an exact path planner, degeneracies are not fully discussed. Their two accompanying videos have no timing or experimental data.

One of the few papers to address the non-existence of paths is Zhang et al [38]. Their implementation work is perhaps the closest to our current work, using subdivision. They noted that “no good implementations are known for general robots with higher than three DOFs”. They achieved planners with 3 and 4 DOFs (one of which is a spatial robot). Although their planners can detect NO-PATH, they do not guarantee detection (this is impossible without exact computation).

### 3 Subdivision Charts and Atlas for $S^2$

**Terminology.** We fix some terminology for the rest of the paper. The fundamental **footprint map**  $Fp$  from configuration space  $C_{space} = C_{space}(R_0)$  to subsets of  $\mathbb{R}^3$  was introduced above. If  $B \subseteq C_{space}$  is any set of configurations, we define  $Fp(B)$  as the union of  $Fp(\gamma)$  as  $\gamma$  ranges over  $B$ . Typically,  $B$  is a “box” of  $C_{space}$  (see below for its meaning in non-Euclidean space  $S^2$ ). We may assume  $\Omega \subseteq \mathbb{R}^3$  is regular (i.e., equal to the closure of its interior). Although  $\Omega$  need not be bounded (e.g., it may be the complement of a box), we assume its boundary  $\partial(\Omega)$  is a bounded set. Then  $\partial(\Omega)$  is partitioned into a set of **(boundary) features: corners** (points), **edges** (relatively open line segments), or **walls** (relatively open triangles). Let  $\Phi(\Omega)$  denote the set of features of  $\Omega$ . The (minimal) set of corners and edges is uniquely defined by  $\Omega$ , but walls depend on a triangulation of  $\partial\Omega$ . If  $A, B \subseteq \mathbb{R}^3$ , define their **separation**  $\text{Sep}(A, B) := \inf \{\|a - b\| : a \in A, b \in B\}$  where  $\|a\|$  is the Euclidean norm. The **clearance** of  $\gamma$  is  $\text{Sep}(Fp(\gamma), \Omega)$ . Say  $\gamma$  is  **$\Omega$ -free** (or simply **free**) if it has positive clearance. Let  $C_{free} = C_{free}(\Omega)$  be the set of  $\Omega$ -free configurations. The **clearance** of a path  $\mu : [0, 1] \rightarrow C_{space}$  is the minimum clearance attained by  $\mu(t)$  as  $t$  ranges over  $[0, 1]$ .

**Subdivision in Non-Euclidean Spaces.** Our  $C_{space}$  has an Euclidean part ( $\mathbb{R}^3$ ) and a non-Euclidean part ( $S^2$ ). We know how to do subdivision in  $\mathbb{R}^3$  but it is less clear for  $S^2$ . Non-Euclidean spaces can be represented either (1) as a submanifold of  $\mathbb{R}^m$  for some  $m$  (e.g.,  $SO(3) \subseteq \mathbb{R}^9$  viewed as orthogonal matrices) or (2) as a subset of  $\mathbb{R}^m$  subject to identification (in the sense of quotient topology [21]). A common representation of  $S^2$  (e.g., [19]) uses a pair of angles (i.e., spherical polar coordinates)  $(\theta, \phi) \in [0, 2\pi] \times [-\pi/2, \pi/2]$  with the identification  $(\theta, \phi) \equiv (\theta', \phi')$  iff  $\{\theta, \theta'\} = \{0, 2\pi\}$  or  $\phi = \phi' = \pi/2$  (North Pole) or  $\phi = \phi' = -\pi/2$  (South Pole). Thus an entire circle of values  $\theta$  is identified with each pole, causing severe distortions near the poles which are singularities. So the numerical primitives in [19, Appendix F] have severe numerical instabilities.

To obtain a representation of  $S^2$  without singularities, we use the map [37]

$$q \in \mathbb{R}^3 \mapsto \widehat{q} := q / \|q\|_\infty$$

whose range is the boundary of a 3D cube  $\widehat{S}^2 := \partial([-1, 1]^3)$ . This map is a bijection when its domain is restricted to  $S^2$ , with inverse map  $q \in \widehat{S}^2 \mapsto \bar{q} := q / \|q\|_2 \in S^2$ . Thus  $\bar{q}$  is the identity for  $q \in S^2$ . We call  $\widehat{S}^2$  the **square model** of  $S^2$ . We view  $S^2$  and  $\widehat{S}^2$  as metric spaces:  $S^2$  has a natural metric whose geodesics are arcs of great circles. The geodesics on  $S^2$  are mapped to the corresponding polygonal geodesic paths on  $\widehat{S}^2$  by  $q \mapsto \widehat{q}$ . Define the constant

$$C_0 := \sup_{p \neq q \in S^2} \left\{ \max \left\{ \frac{d_2(p, q)}{\widehat{d}_2(\widehat{p}, \widehat{q})}, \frac{\widehat{d}_2(\widehat{p}, \widehat{q})}{d_2(p, q)} \right\} \right\}$$

where  $d_2$  and  $\widehat{d}_2$  are the metrics on  $S^2$  and  $\widehat{S}^2$  respectively. Clearly  $C_0 \geq 1$ . Intuitively,  $C_0$  is the largest distortion factor produced by the map  $q \mapsto \widehat{q}$  (by definition the inverse map has the same factor).

► **Lemma 1.**  $C_0 = \sqrt{3}$ .

The proof in the full paper [16] Appendix B.1 also shows that the worst distortion is near the corners of  $\widehat{S}^2$ . The constant  $C_0$  is one of the 4 constants that go into the ultimate accuracy constant  $K$  in the definition of  $\varepsilon$ -exactness (see [37] for details).

It is obvious how to do subdivision in  $\widehat{S^2}$ . This is illustrated in Figure 2(b). After the first subdivision of  $\widehat{S^2}$  into 6 faces, subsequent subdivision is just the usual quadtree subdivision of each face. We interpret the subdivision of  $\widehat{S^2}$  as a corresponding subdivision of  $S^2$ . In [37], we give the general framework using the notion of **subdivision charts and atlases** (borrowing terms from manifold theory).

#### 4 Approximate Footprints for Boxes in $\mathbb{R}^3 \times S^2$

We focus on soft predicates because, in principle, once we have designed and implemented such a predicate, we already have a rigorous and complete planner within the **Soft Subdivision Search** (SSS) framework [32, 37]. (The SSS framework is summarized in the full paper [16] Appendix A.) As noted in the introduction, our soft predicate  $\widetilde{C}$  classifies any input box  $B \subseteq C_{space}$  into one of 3 possible values. A key idea of our 2-link robot work [20, 33] is the notion of “forbidden orientations” (of a box  $B$ , in the presence of  $\Omega$ ). The same concept may be attempted for  $\mathbb{R}^3 \times S^2$ , except that the details seem to be formidable to analyze and to implement. Instead, this paper introduces a direct approximation of the **footprint of a box**,  $Fp(B) := \bigcup \{Fp(\gamma) : \gamma \in B\}$ . We now introduce  $\widetilde{Fp}(B) \subseteq \mathbb{R}^3$  as the **approximate footprint**, and discuss its properties. This section is abstract, in order to expose the mathematical structure of what is needed to achieve resolution-exactness for our planners. The reader might peek at the next two sections to see the instantiations of these concepts for the rod/ring robot.

To understand what is needed of this approximation, recall that our approach to soft predicates is based on the “method of features” [32]. The idea is to maintain a set  $\widetilde{\phi}(B)$  of approximate features for each box  $B$ . We softly classify  $B$  as  $\widetilde{C}(B) = \text{MIXED}$  as long as  $\widetilde{\phi}(B)$  is non-empty; otherwise, we can decide whether  $\widetilde{C}(B) = C(B)$  is **FREE** or **STUCK**. This decision is relatively easy in 2D, but is more involved in 3D and detailed in the full paper [16] Appendix B.2. For correctness of this procedure, we require

$$\widetilde{\phi}(B/\sigma) \subseteq \phi(B) \subseteq \widetilde{\phi}(B). \tag{1}$$

Here  $\sigma > 1$  is some global constant and “ $B/\sigma$ ” denotes the box  $B$  shrunk by factor  $\sigma$ . Basically, (1) guarantees that our soft predicate  $\widetilde{C}(B)$  is conservative and  $\sigma$ -effective (i.e., if  $B$  is free then  $\widetilde{C}(B/\sigma) = \text{FREE}$ ). For computational efficiency, we want the approximate feature sets to have **inheritance property**, i.e.,

$$\widetilde{\phi}(B) \subseteq \widetilde{\phi}(\text{parent}(B)). \tag{2}$$

We now show what this computational scheme demands of our approximate footprint. Define the **exact feature set** of box  $B$  as usual:  $\phi(B) := \{f \in \Phi(\Omega) : f \cap Fp(B) \neq \emptyset\}$  and (tentatively) the **approximate feature set** of box  $B$  as

$$\widetilde{\phi}(B) := \left\{ f \in \Phi(\Omega) : f \cap \widetilde{Fp}(B) \neq \emptyset \right\}. \tag{3}$$

The important point is that  $\widetilde{Fp}(B)$  is defined prior to  $\widetilde{\phi}(B)$ . We need the fundamental inclusions

$$\widetilde{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \widetilde{Fp}(B). \tag{4}$$

Note that this immediately implies (1). Unfortunately, (3) and (4) together do not guarantee inheritance, i.e., (2). Instead, we define  $\widetilde{\phi}'(B)$  recursively as follows:

$$\widetilde{\phi}'(B) := \begin{cases} \left\{ f \in \Phi(\Omega) : f \cap \widetilde{Fp}(B) \neq \emptyset \right\} & \text{if } B \text{ is the root,} \\ \left\{ f \in \widetilde{\phi}'(\text{parent}(B)) : f \cap \widetilde{Fp}(B) \neq \emptyset \right\} & \text{else.} \end{cases} \tag{5}$$

Notice that this only defines  $\tilde{\phi}'(B)$  when  $B$  is an aligned box (i.e., obtained by recursive subdivision of the root box). But  $B/\sigma$  is never aligned when  $B$  is aligned, and thus  $\tilde{\phi}'(B/\sigma)$  is not captured by (5). Therefore we introduce a parallel definition:

$$\tilde{\phi}'(B/\sigma) := \begin{cases} \left\{ f \in \Phi(\Omega) : f \cap \widetilde{Fp}(B/\sigma) \neq \emptyset \right\} & \text{if } B \text{ is the root,} \\ \left\{ f \in \tilde{\phi}'(\text{parent}(B)/\sigma) : f \cap \widetilde{Fp}(B/\sigma) \neq \emptyset \right\} & \text{else.} \end{cases} \quad (6)$$

Now,  $\tilde{\phi}'(B)$  satisfies (2). But does it satisfy (1), which is necessary for correctness? This is answered affirmatively by the following lemma (proved in the full paper [16] Appendix B.3):

► **Lemma 2.** *If the approximate footprint  $\widetilde{Fp}(B)$  satisfies Eq. (4), then  $\tilde{\phi}'(B)$  satisfies Eq. (1), i.e.,*

$$\tilde{\phi}'(B/\sigma) \subseteq \phi(B) \subseteq \tilde{\phi}'(B).$$

Since  $\tilde{\phi}'(B)$  has all the properties we need, we have no further use for the definition of  $\tilde{\phi}(B)$  given in (3). Henceforth, we simply write “ $\tilde{\phi}(B)$ ” to refer to the set  $\tilde{\phi}'(B)$  defined in (5) and (6).

**Geometric Notations.** We will be using planar concepts like circles, squares, etc. for sets that lie in some plane of  $\mathbb{R}^3$ . We shall call them **embedded** circles, squares, etc. By definition, if  $X$  is an embedded object then it defines a unique plane  $Plane(X)$  (unless  $X$  lies in a line). Let  $Ball(r, c) \subseteq \mathbb{R}^3$  denote a ball of radius  $r$  centered at  $c$ . If  $c$  is the origin, we simply write  $Ball(r)$ . Suppose  $X \subseteq \mathbb{R}^3$  is any non-empty set. Let  $Ball(X)$  denote the **circumscribing ball** of  $X$ , defined as the smallest ball containing  $X$ . Next, if  $c \notin X$  then  $Cone(c, X)$  denotes the union of all the rays from  $c$  through points in  $X$ , called the **cone** of  $X$  with **apex**  $c$ . We consider two cases of  $X$  in this cone definition: if  $X$  is a ball, then  $Cone(c, X)$  is called a **round cone**. If the radius of ball  $X$  is  $r$  and the distance from the center of  $Ball(X)$  to  $c$  is  $h \geq r$ , then call  $\arcsin(r/h)$  the **half-angle** of the cone; note that the angle at the apex is twice this half-angle. If  $X$  is an embedded square, we call  $Cone(c, X)$  a **square cone**, and the ray from  $c$  through the center of the square is called the **axis** of the square cone. If  $P$  is any plane that intersects the axis of a square cone  $Cone(c, X)$ , then  $P \cap Cone(c, X)$  is a square iff  $P$  is parallel to square  $X$ . A **ring** (resp., **cylinder**) is the Minkowski sum of an embedded circle (resp., a line) with a ball. Finally consider a box  $B = B^t \times B^r \subseteq \mathbb{R}^3 \times \widehat{S}^2$  where  $B^t$  and  $B^r$  are the **translational** and **rotational** components of  $B$ , and  $B^r$  is either  $\widehat{S}^2$  or a subsquare of a face of  $\widehat{S}^2$ . We let  $m_B$  and  $r_B$  denote the center and radius (distance from the center to any corner) of  $B^t$ . The **cone of**  $B$ , denoted  $Cone(B)$ , is the round cone  $Cone(m_B, Ball(m_B + B^r))$ . If the center of square  $m_B + B^r$  is  $c$  and width of  $B^r$  is  $w$ , then  $Cone(B)$  is just  $Cone(m_B, Ball(c, w/\sqrt{2}))$ .

## 4.1 On $\Sigma_2$ -Sets

Besides the above inclusion properties of  $\widetilde{Fp}(B)$ , we also need to decide if  $\widetilde{Fp}(B)$  intersects a given feature  $f$ . We say  $\widetilde{Fp}(B)$  is “nice” if there are intersection algorithms that are easy to implement (desideratum G2) and practically efficient (desideratum G3). We now formalize and generalize some “niceness” properties of  $\widetilde{Fp}(B)$  that were implicit in our previous work ([32, 20, 33], especially [39]).

An **elementary set** (in  $\mathbb{R}^3$ ) is defined to be one of the following sets or their complements: half space, ball, ring, cone or cylinder. Let  $\mathcal{E}$  (or  $\mathcal{E}_3$ ) denote the set of elementary sets in  $\mathbb{R}^3$ . In  $\mathbb{R}^2$ , we have a similar notion of elementary sets  $\mathcal{E}_2$  comprising half-planes, discs or their complements. All these elementary sets are defined by a single polynomial inequality – so

technically, they are all “algebraic half-spaces”. The sets in  $\mathcal{E}$  are evidently “nice” (niceness of a ring has some subtleties – see Sec. 6). We next extend our collection of nice sets: define a  $\Pi_1$ -set to be a finite intersection of elementary sets. We regard a  $\Pi_1$ -set  $S = \bigcap_{i=1}^n S_i$  to be “nice” because we can easily check if a feature  $f$  intersects  $S$  by a simple while-loop (see below). Notice that  $\Pi_1$  contains all convex polytopes in  $\mathbb{R}^3$ . Our definitions of  $\widetilde{Fp}(B)$  in [32, 20, 33] are all  $\Pi_1$ -sets. But in [39], we make a further extension: define a  $\Sigma_2$ -set to be a finite union of the  $\Pi_1$ -sets, i.e., each  $\Sigma_2$ -set  $S$  has the form

$$S = \bigcup_{i=1}^n \bigcap_{j=1}^{m_i} S_{ij} \tag{7}$$

where  $S_{ij}$ ’s are elementary sets. We still say such an  $S$  is “nice” since checking if a feature  $f$  intersects  $S$  can be written in a doubly-nested loop (see below). Although this intersection is more expensive to check than with a  $\Pi_1$ -set, it may result in fewer subdivisions and better efficiency in the overall algorithm. Thus, there is an accuracy-efficiency trade-off. *Good approximations of footprints are harder to do accurately in 3D, and the extra power of  $\Sigma_2$  seems critical.*

We can put all these in the framework of a well-known<sup>3</sup> construction of an infinite hierarchy of sets, starting from some initial collection of sets. If  $\Delta$  is any collection of sets, let  $\Pi(\Delta)$  denote the collection of finite intersections of sets in  $\Delta$ ; similarly,  $\Sigma(\Delta)$  denotes the collection of finite unions of sets in  $\Delta$ . Then, starting with any collection  $\Delta_1$  of sets, define the infinite hierarchy of sets:

$$\Sigma_i, \Pi_i, \Delta_i \quad (i \geq 1) \tag{8}$$

where  $\Sigma_i := \Sigma(\Delta_i)$ ,  $\Pi_i := \Pi(\Delta_i)$ , and  $\Delta_{i+1} := \Sigma_i \cup \Pi_i$ . An element of  $\Sigma_i$  or  $\Pi_i$  is simply called a  $\Sigma_i$ -set or a  $\Pi_i$ -set.

We call (7) a  $\Sigma_2$ -decomposition of  $S$ , where  $\Delta_1 := \mathcal{E}$ . Note that this decomposition may not be unique, but in the cases arising from our simple robots, there is often an obvious optimal description. Moreover,  $n$  and  $m_i$ ’s are small constants. We can construct new sets by manipulating such a decomposition, e.g., replacing each  $S_{ij}$  by its  $\tau$ -expansion, i.e.,  $S_{ij} \oplus \text{Ball}(\tau)$  (where  $\oplus$  denotes the Minkowski sum), which remains elementary. Under certain conditions, the corresponding set is a reasonable approximation to  $S \oplus \text{Ball}(\tau)$ . If so, we can generalize the corresponding soft predicate to robots with thickness  $\tau$ .

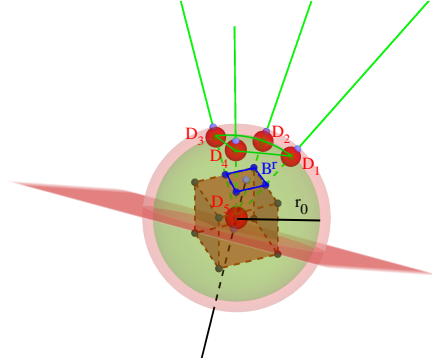
Once we have a  $\Sigma_2$ -decomposition of  $\widetilde{Fp}(B)$ , we can implement the intersection test with relative ease (G2) and quite efficiently (G3). For instance we can test intersection of the set  $S$  in (7) with a feature  $f$  by writing a doubly nested loop. At the beginning of the inner loop, we can initialize a set  $f_0$  to  $f$ . Then the inner loop amounts to the update “ $f_0 \leftarrow f_0 \cap S_{ij}$ ” for  $j = 1, \dots, m_i$ . If ever  $f_0$  becomes empty, we know that the set  $S_i = \bigcap_{j=1}^{m_i} S_{ij}$  has empty intersection with  $f$ . The possibility of such representations is by no means automatic but in the next two sections we verify that they can be achieved for our rod and ring robots. These sections make our planners fully “explicit” for an implementation.

## 5 Soft Predicates for a Rod Robot

In this section,  $R_0$  is a rod with length  $r_0$ ; we choose one endpoint of the rod as the rotation center. Let  $B = B^t \times B^r \subseteq \mathbb{R}^3 \times \widetilde{S}^2$  be a box. Our main goal is to define approximate footprint  $\widetilde{Fp}(B)$ , and to prove the basic inclusions in Eqs. (4) and (1). This turns out to be a  $\Pi_1$ -set (we also indicate a more accurate  $\Sigma_2$ -set.)

<sup>3</sup> From mathematical analysis, constructive set theory and complexity theory.





■ **Figure 3** Rod Robot:  $Fp(B) = Fp_0(B) \oplus (B^t - m_B)$  where  $Fp_0(B)$  is indicated by the four green rays.

It is useful to define the **inner footprint** of  $B$ ,  $Fp_0(B)$ , as  $Fp(m_B \times B^r)$ .

This set is the intersection of a ball and a square cone:

$$Fp_0(B) = \text{Ball}(r_0, m_B) \cap \text{Cone}(m_B, B^r + m_B). \quad (9)$$

The edges of this square cone is shown as green lines in Figure 3; furthermore, the brown box is  $\widehat{S^2} + m_B$  (translation of  $\widehat{S^2}$  so that it is centered at  $m_B$ ). Note that the box footprint  $Fp(B)$  is the Minkowski sum of  $Fp_0(B)$  with  $B^t - m_B$  (the translation of  $B^t$  to make it centered at the origin). It is immediate that

$$Fp_0(B) \subseteq \text{Cone}(B).$$

Thus we may write  $\text{Cone}(m_B, B^r + m_B)$  as the intersection of four half spaces  $H_i$  ( $i = 1, \dots, 4$ ). Let  $\text{Cone}^{(+r_B)}(m_B, B^r + m_B)$  denote the intersection of the expanded half-spaces,  $H_i \oplus \text{Ball}(r_B)$  ( $i = 1, \dots, 4$ ). In general,  $\text{Cone}^{(+r_B)}(m_B, B^r + m_B)$  is not a cone (it may not have a unique ‘‘apex’’). Similarly we ‘‘expand’’ the inner footprint of (9) into

$$‘‘\widetilde{Fp}(B)’’:= \text{Ball}(r_0 + r_B, m_B) \cap \text{Cone}^{(+r_B)}(m_B, B^r + m_B). \quad (10)$$

We use quotes for ‘‘ $\widetilde{Fp}(B)$ ’’ in (10) because we view it as a candidate for an approximate footprint of  $B$ . Certainly, it has the desired property of containing the exact footprint  $Fp(B)$ . Unfortunately, this is not good enough. To see this, let  $\theta$  be the half-angle of the round cone  $\text{Cone}(B) = \text{Cone}(m_B, \text{Ball}(B^r + m_B))$ . Then Hausdorff distance of ‘‘ $\widetilde{Fp}(B)$ ’’ from  $Fp(B)$  can be arbitrarily big as  $\theta$  becomes arbitrarily small. Indeed  $\theta$  can be arbitrarily small because it can be proportional to the input resolution  $\varepsilon$ . We conclude that such a planner is not resolution-exact. To fix this problem, we finally define

$$\widetilde{Fp}(B) := ‘‘\widetilde{Fp}(B)’’ \cap H_0 \quad (11)$$

where  $H_0$  is another half space. A natural choice for  $H_0$  is the half-space ‘‘above’’ the pink-color plane of Figure 3, defined as the plane normal to the axis of cone  $\text{Cone}(B)$  and at distance  $r_B$  ‘‘below’’  $m_B$ . We can also use the ‘‘horizontal’’ plane that is parallel to  $B^r$  and containing the ‘‘lower’’ face of  $B^t$ . We adopt this latter  $H_0$  to have a simpler geometric structure.

This completes the description of  $\widetilde{Fp}(B)$ . It should be clear that checking if  $\widetilde{Fp}(B)$  intersects any feature  $f$  is relatively easy (since it is even a  $\Pi_1$ -set). In the full paper [16] Appendix C we prove the following theorem:

► **Theorem 3.** *The approximate footprint  $\widetilde{Fp}(B)$  as defined for a rod robot satisfies Eq. (4), i.e., there exists some fixed constant  $\sigma > 1$  such that  $\widetilde{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \widetilde{Fp}(B)$ .*

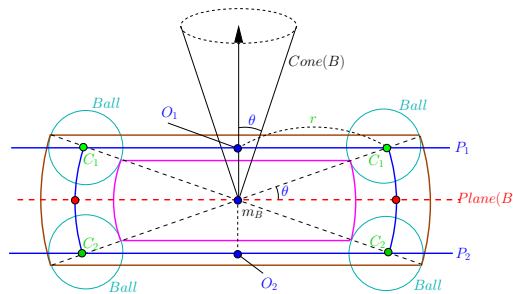
**6 Soft Predicates for a Ring Robot**

Let  $R_0$  be a ring robot. Its footprint is an embedded circle of radius  $r_0$ . First we show how to compute  $\text{Sep}(C, f)$ , the separation of an embedded circle  $C$  from a feature  $f$ . This was treated in detail by Eberly [9]. This is easy when  $f$  is a point or a plane. When  $f$  is a line, Eberly gave two formulations: they reduce to solving a system of 2 quadratic equations in 2 variables, and hence to solving a quartic equation; see the full paper [16] Appendix D.1 for more details. The predicate “Does  $f$  intersect  $C \oplus \text{Ball}(r')$ , a ring of thickness  $r'$ ?” is needed later; it reduces to “Is  $\text{Sep}(C, f) \leq r'$ ?”.

Our next task is to describe an approximate footprint, First recall the round cone of box  $B$  defined in the previous section:  $\text{Cone}(B) = \text{Cone}(m_B, \text{Ball}(m_B + B^r))$ . Let  $\theta = \theta(B)$  be the half-angle of this cone, and  $c$  the center of  $B^r$ . Here, we think of  $c$  as a point of  $\widehat{S^2}$ , and define  $\gamma(B) := m_B \times c$  viewed as an element of  $\mathbb{R}^3 \times \widehat{S^2}$ . Call  $\gamma(B)$  the **central configuration** of box  $B$ . Let  $\text{Ray}(B)$  be the ray from  $m_B$  through  $m_B + c$ . If  $\text{Plane}(B)$  is the plane through  $m_B$  and normal to  $\text{Ray}(B)$ , then the footprint  $\text{Fp}(\gamma(B))$  is an embedded circle lying in  $\text{Plane}(B)$ . We define the **inner footprint** of  $B$  as  $\text{Fp}_0(B) := \text{Fp}(m_B \times B^r)$ . The map  $q \mapsto \bar{q}$  is the inverse of  $q \mapsto \hat{q}$ , taking  $c \in \widehat{S^2}$  to  $\bar{c} \in S^2$ . It is hard to work with  $\text{Fp}_0(B)$ . Instead consider the set  $D(B)$  of all points in  $S^2$  whose distance<sup>4</sup> from  $\bar{c}$  is at most  $\theta(B)$ . So  $D(B)$  is the intersection of  $S^2$  with a round cone with ray from the origin to  $c$ . Then we have  $\text{Fp}_0(B) \subseteq \text{Fp}_1(B)$  where

$$\text{Fp}_1(B) := \text{Fp}(m_B \times D(B)). \tag{12}$$

Our main computational interest is the approximate footprint of  $B$  defined as



**Figure 4** Ring Robot: central cross-section of  $\text{Fp}_1(B)$  appears as two blue arcs.  $\widetilde{\text{Fp}}(B)$  equals the union of two “thick rings” and a “truncated annulus”. The axis of  $\text{Cone}(B)$  is shown as a vertical ray. Each *Ball* has radius  $r_B$ .

$$\widetilde{\text{Fp}}(B) := \text{Fp}_1(B) \oplus \text{Ball}(r_B). \tag{13}$$

Note that  $\text{Fp}_1(B)$  has a simple geometric description. We illustrate this in Figure 4 using a central cross-section with a plane through  $m_B$  containing the axis of  $\text{Cone}(B)$  (the axis of  $\text{Cone}(B)$  is drawn vertically). The footprint of  $\gamma(B)$  is a circle that appears as two red dots in the horizontal line (i.e.,  $\text{Plane}(B)$ ). Let  $S^2(m_B, r_0)$  denote the 2-sphere centered at  $m_B$  with radius  $r_0$ . Then  $\text{Fp}_1(B)$  is the intersection of  $S^2(m_B, r_0)$  with a slab (i.e., intersection of two half-spaces whose bounding planes  $P_1$  and  $P_2$  are parallel to  $\text{Plane}(B)$ ). These planes

<sup>4</sup> Recall that  $S^2$  is a metric space whose geodesics are arcs of great circles.



appear as two horizontal blue lines in Figure 4. In the cross section,  $Fp_1(B)$  are seen as two blue circular arcs. For  $i = 1, 2$ , let  $C_i = P_i \cap S^2(m_B, r_0)$ ; it is an embedded circle that appears as a pair of green points in Figure 4. Each  $C_i$  is centered at  $O_i$ , with radius  $r = r_0 \cos \theta$ ; see Figure 4.

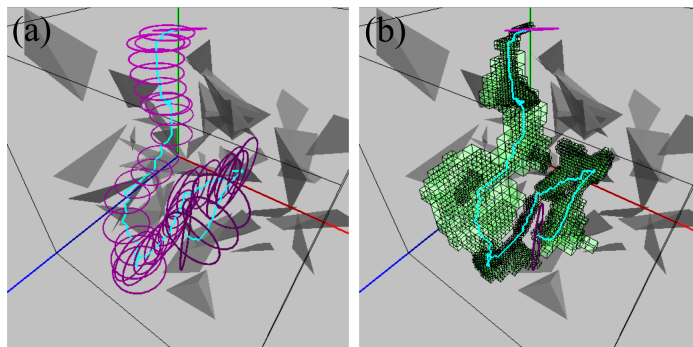
We can now describe a  $\Sigma_2$ -decomposition of  $\widetilde{Fp}(B)$ : it is the union of two “thick rings”,  $C_1 \oplus \text{Ball}(r_B)$  and  $C_2 \oplus \text{Ball}(r_B)$  (both of thickness  $r_B$ ), and a shape  $\text{Ann}(B)$  which we call a **truncated annulus**. First of all, the region bounded between the spheres  $S^2(m_B, r_0 + r_B)$  (the brown arcs in the figure) and  $S^2(m_B, r_0 - r_B)$  (the magenta arcs) is called a (solid) annulus. Let  $C_i^*$  denote the embedded disc whose relative boundary is  $C_i$ . Then we have two round cones,  $\text{Cone}(m_B, C_1^*)$  and  $\text{Cone}(m_B, C_2^*)$ . Together, they form a *double cone* that is actually a simpler object for computation! Finally, define  $\text{Ann}(B)$  to be the intersection of the annulus with the complements of the double cone.

For each thick ring  $C_i \oplus \text{Ball}(r_B)$ , deciding “Does a feature  $f$  intersect  $C_i \oplus \text{Ball}(r_B)$ ?” is equivalent to “Is  $\text{Sep}(C_i, f) \leq r_B$ ?” (see beginning of this section). In the full paper [16], Appendix D.1 discusses this computation and Appendix D.2 proves the following theorem:

► **Theorem 4.** *The approximate footprint  $\widetilde{Fp}(B)$  as defined for a ring robot satisfies Eq. (4), i.e., there exists some fixed constant  $\sigma > 1$  such that  $\widetilde{Fp}(B/\sigma) \subseteq Fp(B) \subseteq \widetilde{Fp}(B)$ .*

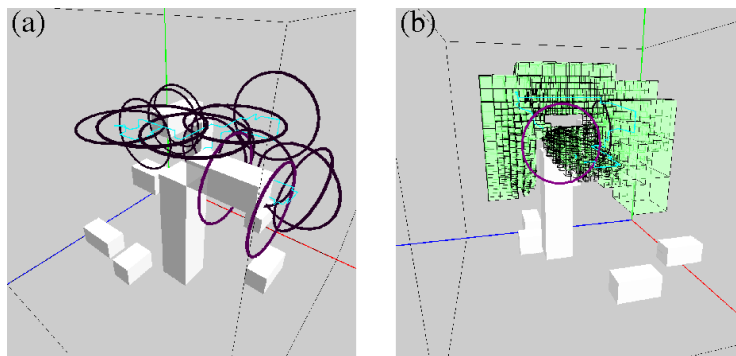
## 7 Practical Efficiency of Correct Implementations

We have developed  $\varepsilon$ -exact planners for rod and ring robots. We have explicitly exposed all the details necessary for a correct implementation, i.e., criterion (G1). The careful design of the approximate footprints of boxes as  $\Sigma_2$ -sets ensures (G2), i.e., it would be relatively easy to implement. We now address (G3) or practical efficiency. For robots with 5 or more DOFs, it becomes extremely critical that good search strategies are deployed. In this paper, we have found that some form of Voronoi heuristic is extremely effective: the idea is to find paths along Voronoi curves (in the sense of [24, 28]), and exploit subdivision Voronoi techniques based (again) on the method of features [34, 2]. There are subtleties necessitating the use of pseudo-Voronoi curves [19, 28, 29]. Since we do not rely on Voronoi heuristics for correctness, simple expedients are available. To recognize Voronoi curves, we maintain (in addition to the collision-detection feature set  $\widetilde{\phi}(B)$ ), the **Voronoi feature set**  $\widetilde{\phi}_V(B)$ . These two sets have some connection but there are no obvious inclusion relationships.

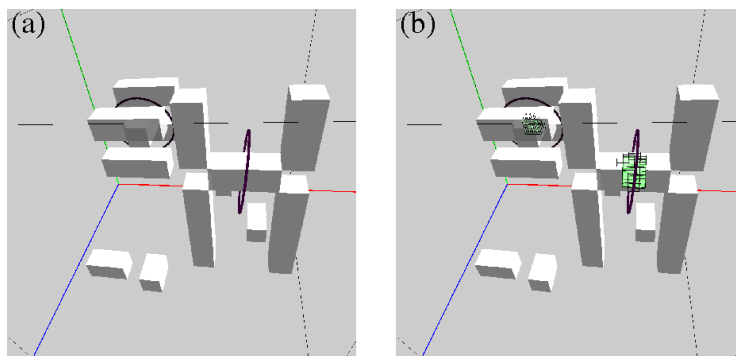


■ **Figure 5** Ring robot amidst 40 random tetrahedra: (a) trace of a found path; (b) subdivision of translational boxes on the path.

Our current implementation achieves near real-time performance (see video [http://cs.nyu.edu/exact/gallery/rod-ring/rod\\_ring.html](http://cs.nyu.edu/exact/gallery/rod-ring/rod_ring.html)). Table 1 summarizes ex-



■ **Figure 6** Ring robot amidst pillars and L-shaped posts (Posts): (a) trace of a found path; (b) subdivision of translational boxes on the path.



■ **Figure 7** Ring robot amidst another set of pillars and posts (Posts2): (a) start and goal configurations (no path found); (b) subdivision of translational boxes during the search.

■ **Table 1** Rod and Ring Experiments.

|        |         | Rod    |            | Robot                           |                              |      |             |              |  |
|--------|---------|--------|------------|---------------------------------|------------------------------|------|-------------|--------------|--|
| Exp. # | Envir.  | Length | $\epsilon$ | Start Conf.                     | Goal Conf.                   | Path | Time (s)    | #Boxes (K)   |  |
| 1/2    | Rand100 | 120    | 16/8       | (240, 120, 360, -0.5, -0.5, -1) | (220, 50, 80, 0.1, 0.8, 1)   | Y/Y  | 1.05/2.82   | 8.1/22.1     |  |
| 3/4    | Rand100 | 120    | 16/8       | (400, 60, 380, -1, 0, 0)        | (200, 200, 240, 0, 1, 0)     | Y/Y  | 1.43/3.92   | 19.3/62.2    |  |
| 5/6    | Rand40  | 160    | 16/8       | (80, 32, 480, 0, 0, -1)         | (240, 440, 200, 1, 0, 0)     | Y/Y  | 16.12/90.65 | 244.7/1138.2 |  |
| 7/8    | Rand40  | 160    | 16/8       | (400, 480, 80, 0, -1, 0)        | (30, 80, 480, 0.5, 0.1, -1)  | Y/Y  | 14.54/9.4   | 217.5/113.0  |  |
| 9/10   | Posts   | 60     | 16/8       | (160, 480, 190, 0, 0.1, -1)     | (390, 60, 420, 1, 0, 0)      | Y/Y  | 0.07/0.13   | 2.1/3.7      |  |
| 11/12  | Posts   | 60     | 16/8       | (320, 120, 320, 0, 1, 0)        | (200, 360, 60, 0, -1, 0)     | N/N  | 1.77/242.7  | 25.6/3790.3  |  |
|        |         | Ring   |            | Robot                           |                              |      |             |              |  |
| Exp. # | Envir.  | Radius | $\epsilon$ | Start Conf.                     | Goal Conf.                   | Path | Time (s)    | #Boxes (K)   |  |
| 1/2    | Rand100 | 40     | 16/8       | (240, 120, 360, -0.5, -0.5, -1) | (220, 50, 80, 0.1, 0.8, 1)   | Y/Y  | 0.66/0.57   | 2.72/2.63    |  |
| 3/4    | Rand100 | 40     | 16/8       | (400, 60, 380, -1, 0, 0)        | (160, 240, 240, 0, 1, 0)     | Y/Y  | 0.25/0.24   | 0.92/0.92    |  |
| 5/6    | Rand40  | 60     | 16/8       | (80, 120, 480, 0, 0, -1)        | (240, 440, 200, 1, 0, 0)     | Y/Y  | 9.38/30.61  | 38.37/71.05  |  |
| 7/8    | Rand40  | 60     | 16/8       | (400, 480, 80, 0, -1, 0)        | (100, 80, 480, 0.5, 0.1, -1) | Y/Y  | 2.07/3.76   | 10.61/139.90 |  |
| 9/10   | Posts   | 60     | 16/8       | (200, 320, 190, 0, 0.1, -1)     | (390, 320, 320, 1, 0, 0)     | Y/Y  | 35.68/89.7  | 114.3/139.3  |  |
| 11/12  | Posts2  | 60     | 16/8       | (410, 90, 190, 0, 0.1, -1)      | (315, 220, 325, 0, 1, 0)     | N/N  | 7.03/271.3  | 8.1/539.1    |  |

periments on our rod and ring robots. The environments Rand100, Rand40 (100 and 40 random tetrahedra), Posts and Posts2 are shown in Figs. 1, 5, 6 and 7. The dimensions of the environments are  $512^3$ . Our implementation uses C++ and OpenGL on the Qt platform. Our code, data and experiments are distributed<sup>5</sup> with our open source `Core Library`. We ran our experiments on a MacBook Pro under Mac OS X 10.10.5 with a 2.5 GHz Intel Core i7 processor, 16GB DDR3-1600 MHz RAM and 500GB Flash Storage. Details about these experiments are found in a folder in `Core Library` for this paper; a `Makefile` there can automatically run all the experiments. Thus these results are reproducible from the data there.

Table 2 (correlated with Table 1 by the Exp #'s) compares our methods with various sampling-based planners in OMPL [31], where we accepted the default parameters and each instance was run 10 times, with the “average time (in s)/standard deviation/success rate” reported. This comparison has various caveats: we simulated the rod and ring robots by polyhedral approximations. We usually outperform RRT in cases of PATH. In case of NO-PATH, we terminated in real time while all sampling methods timed out (300s).

■ **Table 2** Comparison with Sampling Methods in OMPL (the best run-time is shown in bold).

| Rod Robot  |         |                 |                        |                 |                       |               |               |               |                       |
|------------|---------|-----------------|------------------------|-----------------|-----------------------|---------------|---------------|---------------|-----------------------|
| Exp. #     | Ours    | PRM             | Lazy PRM               | RRT             | Lazy RRT              | RRT Connect   | PDST          | BFMT          | Lazy Bi-KPIECE        |
| 1          | 1.05/Y  | 0.036/0.027/1   | <b>0.017</b> /0.024/1  | 1.18/0.74/1     | 0.019/0.023/1         | 0.22/0.043/1  | 0.058/0.055/1 | 1.11/0.18/1   | 0.58/0.36/1           |
| 3          | 1.43/Y  | 0.05/0.047/1    | 0.028/0.019/1          | 1.73/0.82/1     | <b>0.024</b> /0.024/1 | 0.23/0.023/1  | 0.1/0.056/1   | 1.51/0.2/1    | 0.59/0.28/1           |
| 5          | 16.12/Y | 0.044/0.036/1   | 0.051/0.025/1          | 22.1/43/1       | <b>0.036</b> /0.032/1 | 0.99/0.36/1   | 0.21/0.11/1   | 1.74/0.33/1   | 0.44/0.18/1           |
| 7          | 14.54/Y | 0.077/0.04/1    | <b>0.03</b> /0.02/1    | 10.31/6.08/1    | 0.033/0.023/1         | 1.26/0.5/1    | 0.26/0.2/1    | 1.74/0.32/1   | 0.5/0.21/1            |
| 9          | 0.07/Y  | 0.0058/0.002/1  | 0.0038/0.0044/1        | 1.17/0.78/1     | <b>0.003</b> /0.002/1 | 0.084/0.017/1 | 0.025/0.025/1 | 0.3/0.053/1   | 0.065/0.032/1         |
| 11         | 1.77/N  | 300/0/0         | 300/0/0                | 300/0/0         | 300/0/0               | 300/0/0       | 300/0/0       | 300/0/0       | 300/0/0               |
| Ring Robot |         |                 |                        |                 |                       |               |               |               |                       |
| Exp. #     | Ours    | PRM             | Lazy PRM               | RRT             | Lazy RRT              | RRT Connect   | PDST          | BFMT          | Lazy Bi-KPIECE        |
| 1          | 0.66/Y  | 0.0057/0.0026/1 | <b>0.0056</b> /0.005/1 | 1.15/0.93/1     | 0.0061/0.009/1        | 0.037/0.005/1 | 0.015/0.01/1  | 0.15/0.01/1   | 0.077/0.035/1         |
| 3          | 0.25/Y  | 0.0085/0.0067/1 | <b>0.003</b> /0.002/1  | 24.16/54/1      | 0.012/0.0085/1        | 0.052/0.011/1 | 0.008/0.008/1 | 0.145/0.023/1 | 0.068/0.032/1         |
| 5          | 9.38/Y  | 0.019/0.014/1   | <b>0.01</b> /0.004/1   | 300/0/0         | 150.07/10.05/0.5      | 0.53/0.03/1   | 0.057/0.023/1 | 0.22/0.04/1   | 0.093/0.022/1         |
| 7          | 2.07/Y  | 0.024/0.0066/1  | <b>0.013</b> /0.006/1  | 3/1.49/1        | 0.068/0.007/1         | 1.46/0.14/1   | 0.059/0.044/1 | 0.27/0.048/1  | 0.12/0.027/1          |
| 9          | 35.68/Y | 0.63/0.4/1      | 136.6/138.6/0.7        | 111.4/119.4/0.8 | 300/0/0               | 1.65/0.49/1   | 3.36/4.13/1   | 0.23/0.04/1   | <b>0.097</b> /0.033/1 |
| 11         | 7.03/N  | 300/0/0         | 300/0/0                | 300/0/0         | 300/0/0               | 300/0/0       | 300/0/0       | 300/0/0       | 300/0/0               |

## 8 Conclusions

Path planning in 3D has many challenges. Our 5-DOF spatial robots have pushed the current limits of subdivision methods. To our knowledge there is no similar algorithm with comparable rigor or guarantees. Conventional wisdom says that sampling methods can achieve higher DOFs than subdivision. By an estimate of Choset et al [5, p. 202], sampling methods are limited to 5 – 12 DOFs. We believe our approach can reach 6-DOF spatial robots. Since resolution-exactness delivers stronger guarantees than probabilistic-completeness, we expect a performance hit compared to sampling methods. But for simple planar robots (up to 4 DOFs) [32, 20, 33, 39] we observed no such trade-offs because we outperform state-of-the-art sampling methods (such as OMPL [31]) often by two orders of magnitude. But in the 5-DOF robots of this paper, we see that our performance is competitive with sampling methods. It is not clear to us that subdivision is inherently inferior to sampling (we can also do random subdivision). It is true that each additional degree of freedom is conquered only with effort and suitable techniques. This remark seems to cut across both subdivision and sampling approaches; but it hits subdivision harder because of our stronger guarantees.

<sup>5</sup> <http://cs.nyu.edu/exact/core/download/core/>

## References

- 1 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2nd edition, 2006.
- 2 Huxley Bennett, Evanthia Papadopoulou, and Chee Yap. Planar minimization diagrams via subdivision with applications to anisotropic Voronoi diagrams. *Eurographics Symposium on Geometric Processing*, 35(5), 2016. SGP 2016, Berlin, Germany. June 20-24, 2016.
- 3 Rodney A. Brooks and Tomas Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *Proc. 8th Intl. Joint Conf. on Artificial intelligence - Volume 2*, pages 799–806, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.
- 4 John Canny. Computing roadmaps of general semi-algebraic sets. *The Computer Journal*, 36(5):504–514, 1993.
- 5 H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.
- 6 Howie Choset, Brian Mirtich, and Joel Burdick. Sensor based planning for a planar rod robot: Incremental construction of the planar Rod-HGVS. In *IEEE Intl. Conf. on Robotics and Automation (ICRA '97)*, pages 3427–3434, 1997.
- 7 James Cox and Chee K. Yap. On-line motion planning: case of a planar rod. *Annals of Mathematics and Artificial Intelligence*, 3:1–20, 1991. Special journal issue. Also: NYU-Courant Institute, Robotics Lab., No.187, 1988.
- 8 Jory Denny, Kensen Shi, and Nancy M. Amato. Lazy Toggle PRM: a Single Query approach to motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2407–2414, 2013. Karlsruhe, Germany. May 2013.
- 9 David Eberly. Distance to circles in 3D, May 31 2015. Downloaded from <https://www.geometrictools.com/Documentation/Documentation.html>.
- 10 Mohab Safey el Din and Eric Schost. A baby steps/giant steps probabilistic algorithm for computing roadmaps in smooth bounded real hypersurface. *Discrete and Comp. Geom.*, 45(1):181–220, 2011.
- 11 Hazel Everett, Christian Gillot, Daniel Lazard, Sylvain Lazard, and Marc Pouget. The Voronoi diagram of three arbitrary lines in  $\mathbb{R}^3$ . In *25th European Workshop on Computational Geometry (EuroCG'09)*, 2009. March 2009, Bruxelles, Belgium.
- 12 Hazel Everett, Daniel Lazard, Sylvain Lazard, and Mohab Safey el Din. The Voronoi diagram of three lines. *Discrete and Comp. Geom.*, 42(1):94–130, 2009. See also 23rd SoCG, 2007. pp.255–264.
- 13 Dan Halperin, Efi Fogel, and Ron Wein. *CGAL Arrangements and Their Applications*. Springer-Verlag, Berlin and Heidelberg, 2012.
- 14 Dan Halperin, Oren Salzman, and Micha Sharir. Algorithmic motion planning. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017. Expanded from second edition.
- 15 Michael Hemmer, Ophir Setter, and Dan Halperin. Constructing the exact Voronoi diagram of arbitrary lines in three-dimensional space. In *Algorithms – ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 398–409. Springer Berlin / Heidelberg, 2010.
- 16 Ching-Hsiang Hsu, Yi-Jen Chiang, and Chee Yap. Rods and rings: Soft subdivision planner for  $\mathbb{R}^3 \times \mathbb{S}^2$ , 2019. Hosted on arXiv as [arXiv:1903.09416 \[cs.CG\]](https://arxiv.org/abs/1903.09416). Also available at <http://cse.poly.edu/chiang/rod-ring18.pdf>.
- 17 V. Koltun. Planes are not flat: rigid motion planning in three dimensions. In *Proc. 16th ACM-SIAM Sympos. Discrete Algorithms*, pages 505–514, 2005.
- 18 Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.
- 19 Ji Yeong Lee and Howie Choset. Sensor-based planning for a rod-shaped robot in 3 dimensions: Piecewise retracts of  $\mathbb{R}^3 \times \mathbb{S}^2$ . *Int'l. J. Robotics Research*, 24(5):343–383, 2005.

- 20 Zhongdi Luo, Yi-Jen Chiang, Jyh-Ming Lien, and Chee Yap. Resolution exact algorithms for link robots. In *Proc. 11th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '14)*, volume 107 of *Springer Tracts in Advanced Robotics (STAR)*, pages 353–370, 2015. 3-5 Aug 2014, Boğazici University, Istanbul, Turkey.
- 21 James R. Munkres. *Topology*. Prentice-Hall, Inc, second edition, 2000.
- 22 Michal Nowakiewicz. MST-Based method for 6DOF rigid body motion planning in narrow passages. In *Proc. IEEE/RSJ International Conf. on Intelligent Robots and Systems*, pages 5380–5385, 2010. Oct 18–22, 2010. Taipei, Taiwan.
- 23 Colm Ó'Dúnlaing, Micha Sharir, and Chee K. Yap. Retraction: a new approach to motion-planning. *ACM Symp. Theory of Comput.*, 15:207–220, 1983.
- 24 Colm Ó'Dúnlaing and Chee K. Yap. A “Retraction” method for planning the motion of a disc. *J. Algorithms*, 6:104–111, 1985. Also, Chapter 6 in *Planning, Geometry, and Complexity*, eds. Schwartz, Sharir and Hopcroft, Ablex Pub. Corp., Norwood, NJ. 1987.
- 25 J. T. Schwartz and M. Sharir. On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.
- 26 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.
- 27 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: V. the case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Comm. Pure and Applied Math.*, 37(6):815–848, 1984. doi:10.1002/cpa.3160370605.
- 28 M. Sharir, C. O'D'únlaing, and C. Yap. Generalized Voronoi diagrams for moving a ladder I: topological analysis. *Communications in Pure and Applied Math.*, XXXIX:423–483, 1986. Also: NYU-Courant Institute, Robotics Lab., No. 32, Oct 1984.
- 29 M. Sharir, C. O'D'únlaing, and C. Yap. Generalized Voronoi diagrams for moving a ladder II: efficient computation of the diagram. *Algorithmica*, 2:27–59, 1987. Also: NYU-Courant Institute, Robotics Lab., No. 33, Oct 1984.
- 30 Vikram Sharma and Chee K. Yap. Robust geometric computation. In Jacob E. Goodman, Joseph O'Rourke, and Csaba Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 45, pages 1189–1224. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017. Revised and expanded from 2004 version.
- 31 I.A. Şucan, M. Moll, and L.E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012. doi:10.1109/MRA.2012.2205651.
- 32 Cong Wang, Yi-Jen Chiang, and Chee Yap. On soft predicates in subdivision motion planning. *Comput. Geometry: Theory and Appl. (Special Issue for SoCG'13)*, 48(8):589–605, September 2015.
- 33 Chee Yap, Zhongdi Luo, and Ching-Hsiang Hsu. Resolution-exact planner for thick non-crossing 2-link robots. In *Proc. 12th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '16)*, 2016. 13-16 Dec 2016, San Francisco. The appendix in the full paper (and arXiv from <http://cs.nyu.edu/exact/> (and arXiv:1704.05123 [cs.CG]) contains proofs and additional experimental data.
- 34 Chee Yap, Vikram Sharma, and Jyh-Ming Lien. Towards exact numerical Voronoi diagrams. In *9th Int'l Symp. of Voronoi Diagrams in Science and Engineering (ISVD)*., pages 2–16. IEEE, 2012. Invited Talk. June 27-29, 2012, Rutgers University, NJ. doi:10.1109/ISVD.2012.31.
- 35 Chee K. Yap. Algorithmic motion planning. In J.T. Schwartz and C.K. Yap, editors, *Advances in Robotics, Vol. 1: Algorithmic and geometric issues*, volume 1, pages 95–143. Lawrence Erlbaum Associates, 1987.
- 36 Chee K. Yap. Soft subdivision search in motion planning. In A. Aladren et al., editor, *Proceedings, 1st Workshop on Robotics Challenge and Vision (RCV 2013)*, 2013. A Computing Community Consortium (CCC) Best Paper Award, Robotics Science and Systems Conference (RSS 2013), Berlin. In arXiv:1402.3213.


- 37 Chee K. Yap. Soft subdivision search and motion planning, II: Axiomatics. In *Frontiers in Algorithmics*, volume 9130 of *Lecture Notes in Comp.Sci.*, pages 7–22. Springer, 2015. Plenary Talk at 9th FAW. Guilin, China. Aug 3-5, 2015.
- 38 Liangjun Zhang, Young J. Kim, and Dinesh Manocha. Efficient cell labeling and path non-existence computation using C-obstacle query. *Int'l. J. Robotics Research*, 27(11–12):1246–1257, 2008.
- 39 Bo Zhou, Yi-Jen Chiang, and Chee Yap. Soft subdivision motion planning for complex planar robots. In *Proc. 26th European Symp. Algo.(ESA)*, pages 73:1–73:14, 2018. Helsinki, Finland, Aug 20-24, 2018.
- 40 D.J. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.



# 3-Manifold Triangulations with Small Treewidth

Kristóf Huszár 

Institute of Science and Technology Austria (IST Austria),  
Am Campus 1, 3400 Klosterneuburg, Austria  
kristof.huszar@ist.ac.at

Jonathan Spreer 

Institut für Mathematik, Freie Universität Berlin,  
Arnimallee 2, 14195 Berlin, Germany  
jonathan.spreer@fu-berlin.de

---

## Abstract

---

Motivated by fixed-parameter tractable (FPT) problems in computational topology, we consider the treewidth  $\text{tw}(M)$  of a compact, connected 3-manifold  $M$ , defined to be the minimum treewidth of the face pairing graph of any triangulation  $T$  of  $M$ . In this setting the relationship between the topology of a 3-manifold and its treewidth is of particular interest.

First, as a corollary of work of Jaco and Rubinstein, we prove that for any closed, orientable 3-manifold  $M$  the treewidth  $\text{tw}(M)$  is at most  $4g(M)-2$ , where  $g(M)$  denotes Heegaard genus of  $M$ . In combination with our earlier work with Wagner, this yields that for non-Haken manifolds the Heegaard genus and the treewidth are within a constant factor.

Second, we characterize all 3-manifolds of treewidth one: These are precisely the lens spaces and a single other Seifert fibered space. Furthermore, we show that all remaining orientable Seifert fibered spaces over the 2-sphere or a non-orientable surface have treewidth two. In particular, for every spherical 3-manifold we exhibit a triangulation of treewidth at most two.

Our results further validate the parameter of treewidth (and other related parameters such as cutwidth or congestion) to be useful for topological computing, and also shed more light on the scope of existing FPT-algorithms in the field.

**2012 ACM Subject Classification** Mathematics of computing → Geometric topology; Theory of computation → Fixed parameter tractability

**Keywords and phrases** computational 3-manifold topology, fixed-parameter tractability, layered triangulations, structural graph theory, treewidth, cutwidth, Heegaard genus, lens spaces, Seifert fibered spaces

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.44

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1812.05528>.

**Funding** *Kristóf Huszár*: Partially supported by the Einstein Foundation Berlin (grant EVF-2015-230) for his visits at Freie Universität Berlin.

*Jonathan Spreer*: Supported by grant EVF-2015-230 of the Einstein Foundation Berlin as well as by the DFG Collaborative Research Center SFB/TRR 109 “Discretization in Geometry and Dynamics”.

**Acknowledgements** We thank the developers of the free software *Regina* [11, 12] for creating a fantastic tool, and the anonymous reviewers for useful comments and suggestions regarding the exposition. KH thanks the people at the Discrete Geometry Group, Freie Universität Berlin, for their hospitality.

## 1 Introduction

Any given topological 3-manifold  $\mathcal{M}$  admits infinitely many combinatorially distinct triangulations  $\mathcal{T}$ , and the feasibility of a particular algorithmic task about  $\mathcal{M}$  might greatly depend on the choice of the input triangulation  $\mathcal{T}$ . Hence, it is an important question in



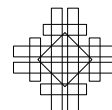
© Kristóf Huszár and Jonathan Spreer;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 44; pp. 44:1–44:20

Leibniz International Proceedings in Informatics



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





computational topology, how “well-behaved” a triangulation can be, taking into account “topological properties” of the underlying 3-manifold.

More concretely, there exist several algorithms in computational 3-manifold topology which solve inherently difficult (e.g., **NP**-hard) problems in linear time in the input size, once the input triangulation has a dual graph of bounded treewidth [13, 14, 15, 16, 32]. Such fixed-parameter tractable (FPT) algorithms are not only of theoretical importance but also provide practical tools: some of them are implemented in software packages such as *Regina* [11, 12] and, in selected cases, outperform previous state-of-the-art methods.

The presence of algorithms FPT in the treewidth of the dual graph of a triangulation immediately poses the following question. Given a 3-manifold  $\mathcal{M}$ , how small can the treewidth of the dual graph of a triangulation of  $\mathcal{M}$  be? This question has recently been investigated in a number of contexts, settling, for instance, that for some 3-manifolds there is no hope of finding triangulations with dual graphs of small treewidth [25] (see [17] for related work concerning the respective question about knots and their diagrams). Hyperbolic 3-manifolds nevertheless always admit triangulations of treewidth upper-bounded by their volume [31].

In this article we also focus on constructing small treewidth triangulations informed by the topological structure of a 3-manifold. To this end, we consider the notion of treewidth (cutwidth) of a 3-manifold as being the smallest treewidth (cutwidth) of a dual graph ranging over all triangulations thereof. The necessary background is introduced in Section 2.

In Section 3, building on [27], we show that the Heegaard genus dominates the cutwidth (and thus the treewidth as well) by virtue of the following statement.

► **Theorem 1.** *Let  $\mathcal{M}$  be a closed, orientable 3-manifold, and let  $\text{cw}(\mathcal{M})$  and  $\mathfrak{g}(\mathcal{M})$  respectively denote the cutwidth and the Heegaard genus of  $\mathcal{M}$ . We have  $\text{cw}(\mathcal{M}) \leq 4\mathfrak{g}(\mathcal{M}) - 2$ .*

Theorem 1, in combination with recent work by the authors and Wagner [25], implies that for the class of so-called non-Haken 3-manifolds, the Heegaard genus is in fact within a constant factor of both the cutwidth and the treewidth of a 3-manifold, providing an interesting connection between a classical topological invariant and topological properties directly related to the triangulations of a manifold. In Section 4, we further strengthen this link by looking at very small values of Heegaard genus and treewidth:

► **Theorem 2.** *The class of 3-manifolds of treewidth at most one coincides with that of Heegaard genus at most one together with the Seifert fibered space  $\text{SFS}[\mathbb{S}^2 : (2, 1), (2, 1), (2, -1)]$  of Heegaard genus two.*

In contrast, in Section 5 we show – by exhibiting treewidth two triangulations for all orientable Seifert fibered spaces over  $\mathbb{S}^2$  (Theorem 15) or a non-orientable surface (Theorem 16) – that linking Heegaard genus to treewidth fails to hold in general in a very strong sense: There are infinite families of 3-manifolds of unbounded Heegaard genus which are all of treewidth two (Corollary 21). Extending these observations we deduce that the treewidth of all 3-manifolds with spherical or  $\mathbb{S}^2 \times \mathbb{R}$  geometry equals two (Corollary 18).

Finally, combining these results, we determine the treewidth of 4889 out of the 4979 manifolds in the ( $\leq 10$ )-tetrahedra census (Table 1). Specifically, only 90 of them have treewidth possibly higher than two. These computations also confirm that not all minimal triangulations are of minimum treewidth (Corollary 20).

Altogether, our results and experiments further suggest that the treewidth of a 3-manifold is an interesting notion at the interface of topology and combinatorics which is well-suited to indicate the power of FPT algorithms in computational 3-manifold topology.

► **Remark 3.** The various triangulations described in Section 5 are available in form of a short *Regina* script [11, 12] in the full version of this article [24].

## 2 Preliminaries

### 2.1 Graphs

A *graph* (more precisely, a *multigraph*)  $G = (V, E)$  is an ordered pair consisting of a finite set  $V = V(G)$  of *nodes* and of a multiset  $E = E(G)$  of unordered pairs of nodes, called *arcs*.<sup>1</sup> A *loop* is an arc  $e \in E$  which is a multiset itself, e.g.,  $e = \{v, v\}$  for some  $v \in V$ . The *degree*  $\deg(v)$  of a node  $v \in V$  equals the number of arcs containing it, counted with multiplicity. If all of its nodes have the same degree  $k \in \mathbb{N}$ , a graph is called *k-regular*. A *tree* is a connected graph with  $n$  nodes and  $n - 1$  arcs. The term *leaf* denotes a node of degree one.

For general background on graph theory we refer to [19].

**Treewidth.** Originating from graph minor theory [29] and central to parametrized complexity [20, Part III], treewidth [5, 7, 41] measures the similarity of a given graph to a tree. More precisely, a *tree decomposition* of  $G = (V, E)$  is a pair  $(\{B_i : i \in I\}, T = (I, F))$  with *bags*  $B_i \subseteq V$ ,  $i \in I$ , and a tree  $T = (I, F)$ , such that a)  $\bigcup_{i \in I} B_i = V$ , b) for every arc  $\{u, v\} \in E$ , there exists  $i \in I$  with  $\{u, v\} \subseteq B_i$ , and c) for every  $v \in V$ ,  $T_v = \{i \in I : v \in B_i\}$  spans a connected subtree of  $T$ . The *width* of a tree decomposition equals  $\max_{i \in I} |B_i| - 1$ , and the *treewidth*  $\text{tw}(G)$  is the smallest width of any tree decomposition of  $G$ .

On one hand, treewidth is useful in the analysis of algorithms [8]. On the other hand, congestion (also known as carving-width) and cutwidth have recently turned out to be helpful mediators to connect treewidth with classical topological invariants [17, 25, 31]. In this work, alongside with treewidth, we also work with cutwidth.

**Cutwidth.** Consider an ordering  $(v_1, \dots, v_n)$  of  $V$ . The set  $C_\ell = \{\{v_i, v_j\} \in E : i \leq \ell < j\}$ , where  $1 \leq \ell < n$ , is called a *cutset*. The *width* of the ordering is the size of the largest cutset. The *cutwidth* [18], denoted by  $\text{cw}(G)$ , is the minimum width over all orderings of  $V$ .

### 2.2 Triangulations and Heegaard splittings of 3-manifolds

The main objects of study in this article are *3-manifolds*, i.e., topological spaces in which every point has a neighborhood homeomorphic to  $\mathbb{R}^3$  or to the closed upper half-space  $\{(x, y, z) \in \mathbb{R}^3 : z \geq 0\}$ . For a 3-manifold  $\mathcal{M}$ , its *boundary*  $\partial\mathcal{M}$  consists of all points of  $\mathcal{M}$  not having a neighborhood homeomorphic to  $\mathbb{R}^3$ . A 3-manifold is *closed* if it is compact and has an empty boundary. Two 3-manifolds are considered equivalent if they are homeomorphic. We refer to [44] for an introduction to 3-manifolds (cf. [22], [23], [26] and [48]), and to [42, Lecture 1] for an overview of the key concepts defined in this subsection.

All 3-manifolds considered in this paper are compact and orientable.

**Triangulations.** In the field of computational topology, a 3-manifold is often presented as a *triangulation* [4, 34], i.e., a finite collection of abstract tetrahedra “glued together” by identifying pairs of their triangular faces called *triangles*. Due to these *face gluings*, several tetrahedral edges (or vertices) are also identified and we refer to the result as a single *edge (or vertex) of the triangulation*. The face gluings, however, cannot be arbitrary. For a triangulation  $\mathcal{T}$  to describe a closed 3-manifold, it is necessary and sufficient that no

<sup>1</sup> Throughout the article, the terms *edge* and *vertex* denote an edge or vertex of a triangulated surface or 3-manifold, while the words *arc* and *node* refer to an edge or vertex in a graph.

tetrahedral edge is identified with itself in reverse, and the boundary of a small neighborhood around each vertex is  $\mathbb{S}^2$ , a 2-sphere. If, in addition, the boundaries of small neighborhoods of some of the vertices are disks, then  $\mathcal{T}$  describes a 3-manifold with boundary.

To study a triangulation  $\mathcal{T}$ , it is often useful to consider its *dual graph*  $\Gamma(\mathcal{T})$ , whose nodes and arcs correspond to the tetrahedra of  $\mathcal{T}$  and to the face gluings between them, respectively. By construction,  $\Gamma(\mathcal{T})$  is a multigraph with maximum degree  $\leq 4$ .

**Heegaard splittings.** *Handlebodies*, which can be thought of as thickened graphs, provide another way to describe 3-manifolds. A *Heegaard splitting* [43] is a decomposition  $\mathcal{M} = \mathcal{H} \cup_f \mathcal{H}'$ , where we start with the disjoint union of two homeomorphic handlebodies,  $\mathcal{H}$  and  $\mathcal{H}'$  and then identify their boundary surfaces via a homeomorphism  $f: \partial\mathcal{H} \rightarrow \partial\mathcal{H}'$  referred to as the *attaching map*. Every closed, compact, and orientable 3-manifold  $\mathcal{M}$  can be obtained this way. Moreover, we may assume, without loss of generality, that  $f$  is orientation-preserving. The smallest genus of a boundary surface ranging over all Heegaard splittings of  $\mathcal{M}$ , denoted by  $g(\mathcal{M})$ , is called the *Heegaard genus* of  $\mathcal{M}$ . Heegaard splittings with isotopic attaching maps yield homeomorphic 3-manifolds, hence are considered equivalent.

### 2.3 Orientable Seifert fibered spaces

Seifert fibered spaces, see [46], comprise an important class of 3-manifolds. Here we describe the orientable ones following [42] (cf. [22, Sec. 2.1], [26, Ch. VI], [35], [36], or [44, Sec. 3.7]).

Let us consider the surface  $\mathcal{F}_{g,r} = \mathcal{F}_g \setminus (\text{int } D_1 \cup \dots \cup \text{int } D_r)$  obtained from the closed orientable genus  $g$  surface by removing the interiors of  $r$  pairwise disjoint disks. Taking the product with the circle  $\mathbb{S}^1$  yields an orientable 3-manifold  $\mathcal{F}_{g,r} \times \mathbb{S}^1$  whose boundary consists of  $r$  tori; namely,  $\partial(\mathcal{F}_{g,r} \times \mathbb{S}^1) = (\partial D_1) \times \mathbb{S}^1 \cup \dots \cup (\partial D_r) \times \mathbb{S}^1$ . For each  $(\partial D_i) \times \mathbb{S}^1$ ,  $1 \leq i \leq r$ , we glue in a solid torus so that its meridian wraps  $a_i$  times around the meridian  $(\partial D_i) \times \{y_i\}$  and  $b_i$  times around the longitude  $\{x_i\} \times \mathbb{S}^1$  of  $(\partial D_i) \times \mathbb{S}^1$ . Here  $a_i$  and  $b_i$  are assumed to be coprime integers with  $a_i \geq 2$ , and the point  $(x_i, y_i) \in (\partial D_i) \times \mathbb{S}^1$  is chosen arbitrarily. This way we obtain a closed orientable 3-manifold  $\mathcal{M} = \text{SFS}[\mathcal{F}_g : (a_1, b_1), \dots, (a_r, b_r)]$  which is called the *Seifert fibered space over  $\mathcal{F}_g$  with  $r$  exceptional (or singular) fibers*. In relation to  $\mathcal{M}$ , the surface  $\mathcal{F}_g$  is referred to as the *base space* (or *orbit surface*).

► **Example 4.** *Lens spaces*, the 3-manifolds of Heegaard genus one, coincide with Seifert fibered spaces over  $\mathbb{S}^2$  having at most one (or two, cf. [42, p. 27]) exceptional fiber(s).<sup>2</sup>

**Non-orientable base spaces.** With a slight modification of the above construction, one can obtain additional orientable Seifert fibered spaces having non-orientable base spaces. Beginning with  $\mathcal{N}_g$ , the non-orientable genus  $g$  surface, we pass to  $\mathcal{N}_{g,r}$  by adding  $r$  punctures (i.e., by removing  $r$  pairwise disjoint open disks). At this point, however, instead of taking the product  $\mathcal{N}_{g,r} \times \mathbb{S}^1$  (which yields a non-orientable 3-manifold) we consider the “orientable  $\mathbb{S}^1$ -bundle” over  $\mathcal{N}_{g,r}$ , which has again  $r$  torus boundary components. As before, we conclude by gluing in  $r$  solid tori, specified by pairs of coprime integers  $(a_i, b_i)$  with  $a_i \geq 2$ , where  $1 \leq i \leq r$ . The notation for the resulting 3-manifold remains the same.

See [30, Section 2] for a concrete and detailed description of Seifert fibered spaces both over orientable and non-orientable surfaces (cf. the classes  $\{Oo, g\}$  and  $\{On, g\}$  therein).

<sup>2</sup> In particular, we regard  $\mathbb{S}^2 \times \mathbb{S}^1$  (the SFS over  $\mathbb{S}^2$  without exceptional fibers) to be a lens space as well.

**Geometric structures on 3-manifolds.** The significance of Seifert fibered spaces is exemplified by their role in the geometrization of 3-manifolds – a celebrated program initiated by Thurston [47], influenced by Hamilton [21], and completed by Perelman [37, 39, 38], cf. [3, 28, 33, 40] – as they account for six out of the eight possible “model geometries” [45] the building blocks may admit in the “canonical decomposition” of a closed 3-manifold.

### 3 The treewidth of a 3-manifold

In this section we prove Theorem 1. For this, we first recall how to turn graph-theoretical parameters, such as treewidth or cutwidth, into topological invariants of 3-manifolds. This is followed by a very brief and selective introduction to the theory *layered triangulations* as defined by Jaco and Rubinstein [27]. We then present the proof of Theorem 1 which, on the topological level, is a direct consequence of this theory, and conclude with a remark on some practical aspects derived from the constructive nature of the proof.

#### 3.1 Topological invariants from graph parameters

Recall the notions of treewidth and cutwidth from Section 2.1.

► **Definition 5.** Let  $\mathcal{M}$  be a 3-manifold and let  $\mathcal{T}$  be a triangulation of  $\mathcal{M}$ . By the treewidth of  $\mathcal{T}$  we mean  $\text{tw}(\Gamma(\mathcal{T}))$ , i.e., the treewidth of its dual graph, and the treewidth  $\text{tw}(\mathcal{M})$  of  $\mathcal{M}$  is defined to be the smallest treewidth of any triangulation of  $\mathcal{M}$ . In other words,

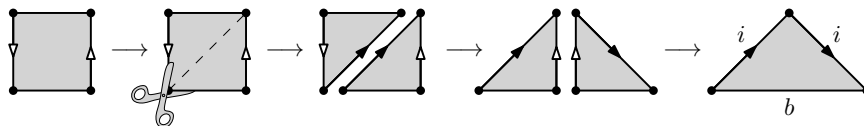
$$\text{tw}(\mathcal{M}) = \min\{\text{tw}(\Gamma(\mathcal{T})) : \mathcal{T} \text{ is a triangulation of } \mathcal{M}\}. \quad (1)$$

The definition of *cutwidth*  $\text{cw}(\mathcal{M})$  is analogous. Using [6, Theorems 47 and 49] it follows that  $\text{tw}(\mathcal{M}) \leq \text{cw}(\mathcal{M})$ . Complementing Definition 5, we note that there are simple arguments proving that any 3-manifold admits triangulations of arbitrarily high treewidth (cf. [24]).

#### 3.2 Layered triangulations

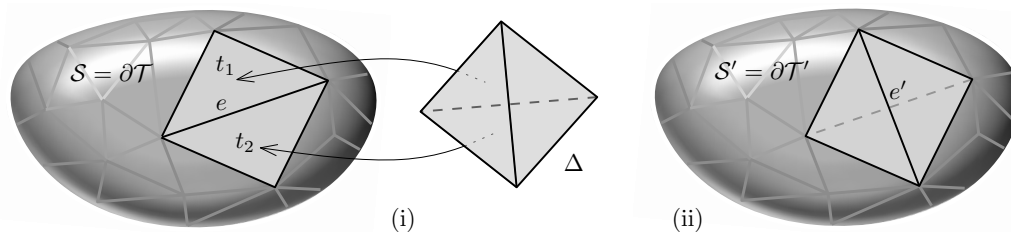
The theory of layered triangulations of 3-manifolds, due to Jaco and Rubinstein [27], captures the inherently topological notion of a Heegaard splitting, see Section 2.2, in a combinatorial way. Here we outline the terminology important for our purposes. Despite all the technicalities, the nomenclature is very expressive and encapsulates much of the intuition.

**Spines and layerings.** Let  $\mathcal{N}_{g,r}$  denote the non-orientable surface of genus  $g$  with  $r$  punctures (i.e., boundary components). A  $g$ -*spine* is a 1-vertex triangulation of  $\mathcal{N}_{g,1}$ . It has one vertex,  $3g - 1$  edges (out of which  $3g - 2$  are interior and one is on the boundary), and  $2g - 1$  triangles. In particular, the Euler characteristic of any  $g$ -spine equals  $1 - g$ .



■ **Figure 1** Transforming the (well-known depiction of the) Möbius band – the non-orientable surface of genus one with one puncture – into a 1-spine with interior edge  $i$ .

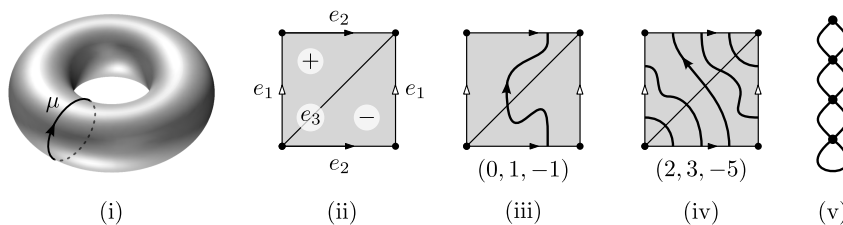
Now consider a triangulation  $\mathcal{S}$  of a surface – usually seen as a  $g$ -spine or as the boundary of a triangulated 3-manifold – and let  $e$  be an interior edge of  $\mathcal{S}$  with  $t_1$  and  $t_2$  being the two triangles of  $\mathcal{S}$  containing  $e$ . Gluing a tetrahedron  $\Delta$  along  $t_1$  and  $t_2$  without a twist is called a *layering* onto the edge  $e$  of the surface  $\mathcal{S}$ , cf. Figure 2(i). Importantly, we allow  $t_1$  and  $t_2$  to coincide, e.g., when layering on the interior edge of a 1-spine (Figure 1, right).



■ **Figure 2** (i) Layering onto the edge  $e$  of  $\mathcal{S} = \partial\mathcal{T}$ , (ii) which has the effect of “flipping”  $e$ .

**Layered handlebodies.** It is a pleasant fact that by layering a tetrahedron onto each of the  $3g - 2$  interior edges of a  $g$ -spine we obtain a triangulation of the genus  $g$  handlebody  $\mathcal{H}_g$ , called a *minimal layered triangulation* thereof (see Figure 4). More generally, we call any triangulation obtained by additional layerings a *layered triangulation* of  $\mathcal{H}_g$  [26, Section 9].

The case  $g = 1$  is of particular importance. Starting with a 1-spine (Figure 1) and layering on its interior edge  $i$  produces a 1-tetrahedron triangulation  $\mathcal{T}$  of the solid torus  $\mathcal{H}_1$  ([24]). Its boundary  $\mathcal{S} = \partial\mathcal{T}$  is the unique 2-triangle triangulation of the torus with one vertex and three edges, and layering onto any edge of  $\mathcal{S}$  yields another triangulation of  $\mathcal{H}_1$ . We may iterate this procedure to obtain further triangulations, any of which we call a *layered solid torus* (cf. [10, Section 1.2] for a detailed exposition). By construction, its dual graph consists of a single loop at the end of a path of double arcs; see, e.g, Figure 3(v).



■ **Figure 3**

While combinatorially the same, boundary triangulations of layered solid tori generally are not isotopic; they can be described as follows. Consider a “reference torus”  $\mathbb{T}$  with a fixed meridian  $\mu$ , Figure 3(i). Given a layered solid torus, its boundary induces a triangulation of  $\mathbb{T}$ . Label the two triangles with  $+$  and  $-$ , and the three edges with  $e_1, e_2$ , and  $e_3$ , Figure 3(ii); and orientation of  $\mu$ . Let  $p, q$  and  $r$  denote the geometric intersection number of  $\mu$  with  $e_1, e_2$  and  $e_3$ , respectively. We say that the corresponding layered solid torus is of type  $(p, q, r)$ , or LST $(p, q, r)$  for short. See, e.g., Figure 3(iii)–(iv).

It can be shown that  $p, q, r$  are always coprime with  $p + q + r = 0$ . Conversely, for any such triplet, one can construct a layered solid torus of type  $(p, q, r)$ , cf. [10, Algorithm 1.2.17].

**Layered 3-manifolds.** Let  $\mathcal{M}$  be a closed, orientable 3-manifold given via a Heegaard splitting  $\mathcal{M} = \mathcal{H} \cup_f \mathcal{H}'$ . If  $\mathcal{H}$  and  $\mathcal{H}'$  can be endowed with layered triangulations  $\mathcal{T}$  and  $\mathcal{T}'$ , respectively, such that the attaching map  $f$  is simplicial, then the union  $\mathcal{T} \cup_f \mathcal{T}'$  triangulates  $\mathcal{M}$  and is called a *layered triangulation* of  $\mathcal{M}$ . The next theorem is fundamental.

► **Theorem 6** (Jaco–Rubinstein, Theorem 10.1 of [27]). *Every closed, orientable 3-manifold admits a layered triangulation (which is a one-vertex triangulation by construction).*

### 3.3 Treewidth versus Heegaard genus

In [25, Theorem 4] it was shown that for a closed, orientable, irreducible, non-Haken (cf. [25, Section 2.2]) 3-manifold  $\mathcal{M}$ , the Heegaard genus  $\mathfrak{g}(\mathcal{M})$  and the treewidth  $\text{tw}(\mathcal{M})$  satisfy

$$\mathfrak{g}(\mathcal{M}) < 24(\text{tw}(\mathcal{M}) + 1). \quad (2)$$

In this section we further explore the connection between these two parameters, guided by two questions: **1.** Does a reverse inequality hold? **2.** Can one refine the assumptions?

For the first one, we give an affirmative answer (Theorem 1). The result is almost immediate if one inspects a layered triangulation of a closed, orientable 3-manifold. Due to work of Jaco and Rubinstein, this approach is always possible (cf. Theorem 6).

The second question is more open-ended. As a first step, we observe the following.

► **Proposition 7.** *There exists an infinite family of 3-manifolds of bounded cutwidth – hence of bounded treewidth – with unbounded Heegaard genus.*

This follows from the fact that, while Heegaard genus is additive under taking connected sums, cutwidth essentially is not affected by this operation, see the full version [24].

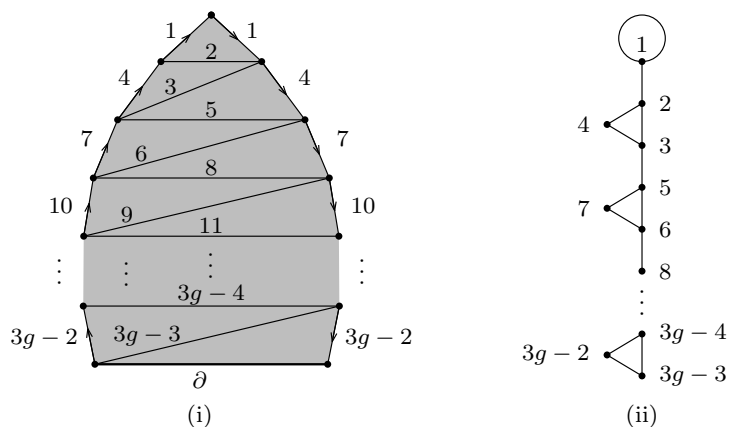
► **Remark 8.** Proposition 7 shows that among reducible 3-manifolds one can easily find infinite families which violate (2). Nevertheless, irreducibility alone is insufficient for (2) to hold. In particular, in Section 5 we prove that orientable Seifert fibered spaces over  $\mathbb{S}^2$  have treewidth at most two (Theorem 15). However, all but two of them are irreducible [44, Theorem 3.7.17] and they can have arbitrarily large Heegaard genus [9, Theorem 1.1].

Recent work of de Mesmay, Purcell, Schleimer, and Sedgwick [17] suggests that one might be able to obtain an inequality similar to (2) for (closed) Haken manifolds as well, by imposing appropriate conditions on the (incompressible) surfaces they contain.

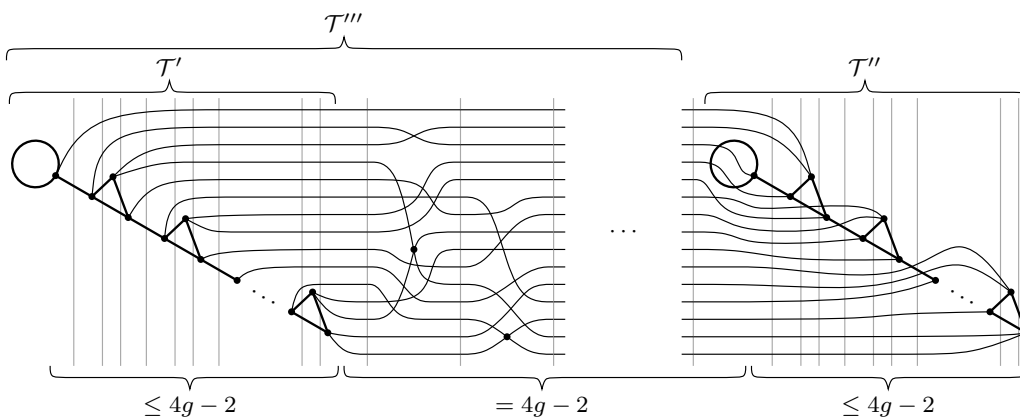
Nevertheless, as mentioned before, a reverse inequality always holds. In order to establish that, we utilize layered triangulations. See Section 3.2 for a brief introduction.

**Proof of Theorem 1.** Let  $g = \mathfrak{g}(\mathcal{M})$ . Consider the  $g$ -spine  $S$  in Figure 4(i) together with the indicated order in which we layer onto the  $3g - 2$  interior edges of  $S$  to build two copies  $\mathcal{T}'$  and  $\mathcal{T}''$  of a minimal layered triangulation of the genus  $g$  handlebody. See Figure 4(ii) for the dual graph of  $\mathcal{T}'$  (and of  $\mathcal{T}''$ ). Note that  $\partial\mathcal{T}'$  and  $\partial\mathcal{T}''$  consist of  $4g - 2$  triangles each.

By Theorem 6, we may extend  $\mathcal{T}'$  to a layered triangulation  $\mathcal{T}'''$  which can be glued to  $\mathcal{T}''$  along a simplicial map  $f : \partial\mathcal{T}''' \rightarrow \partial\mathcal{T}''$  to yield a triangulation  $\mathcal{T} = \mathcal{T}''' \cup_f \mathcal{T}''$  of  $\mathcal{M}$ . This construction imposes a natural ordering on the tetrahedra of  $\mathcal{T}$ : **1.** Start by ordering the tetrahedra of  $\mathcal{T}'$  according to the labels of the edges of  $S$  onto which they are initially layered. **2.** Continue with all tetrahedra between  $\mathcal{T}'$  and  $\mathcal{T}''$  in the order they are attached to  $\mathcal{T}'$  in order to build up  $\mathcal{T}'''$ . **3.** Finish with the tetrahedra of  $\mathcal{T}''$  again in the order of the labels of the edges of  $S$  onto which they are layered. This way we obtain a linear layout of the nodes of  $\Gamma(\mathcal{T})$  which realizes width  $4g - 2$  (Figure 5). Therefore  $\text{cw}(\mathcal{M}) \leq 4g - 2$ . ◀



■ **Figure 4** A  $g$ -spine  $S$  together with the order in which we layer onto its interior edges (i), and the dual graph of resulting minimal layered triangulation of the genus  $g$  handlebody (ii).



■ **Figure 5** A linear layout showing that  $\text{cw}(\mathcal{M})$  is bounded above by  $4\mathfrak{g}(\mathcal{M}) - 2$ .

Combining Theorem 1 with  $\text{tw}(\mathcal{M}) \leq \text{cw}(\mathcal{M})$ , we directly deduce the following.

► **Corollary 9.** *For any closed, orientable, irreducible, non-Haken 3-manifold  $\mathcal{M}$  the Heegaard genus  $\mathfrak{g}(\mathcal{M})$  and the treewidth  $\text{tw}(\mathcal{M})$  satisfy*

$$\frac{1}{4} \text{tw}(\mathcal{M}) + 2 \leq \mathfrak{g}(\mathcal{M}) < 24(\text{tw}(\mathcal{M}) + 1). \tag{3}$$

In [1, Question 5.3] the authors ask whether computing the Heegaard genus of a 3-manifold is still hard when restricting to the set of non-Haken 3-manifolds. Corollary 9 implies that the answer to this question also has implications on the hardness of computing or approximating the treewidth of non-Haken manifolds.

### 3.4 An algorithmic aspect of layered triangulations

Layered triangulations are intimately related to the rich theory of surface homeomorphisms, and in particular the notion of the mapping class group. Making use of this connection, as well as some very useful results due to Bell [2], we present a general algorithmic method to turn a 3-manifold  $\mathcal{M}$ , given by a small genus Heegaard splitting in some reasonable way, into a triangulation of  $\mathcal{M}$  while staying in full control over the size of this triangulation.

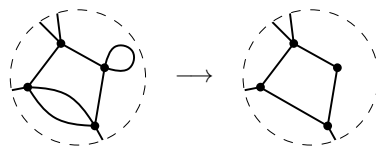


Namely, if  $\mathcal{M}$  is given by a genus  $g$  Heegaard splitting with the attaching map presented as a word  $w$  over a set of Dehn twists  $X$  generating the genus  $g$  mapping class group, then there exists a constant  $K(g)$  such that we can construct a layered triangulation of  $\mathcal{M}$  of size  $O(K(g)|w|)$ , cutwidth  $\leq 4g - 2$ , in time  $O(K(g)(|X| + |w|))$ .

See the full version of this article [24] for the definition of all notions underlying this observation, a precise formulation of the above statement, and a proof.

**4 3-Manifolds of treewidth at most one**

This section is dedicated to the proof of Theorem 2. As the treewidth is not sensitive to multiple arcs or loops, it is helpful to also consider *simplifications* of multigraphs, in which we forget about loops and reduce each multiple arc to a single one (Figure 6).



**Figure 6** The local effect of simplification in a multigraph.

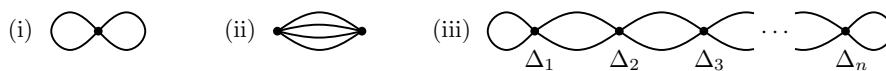
One direction in Theorem 2 immediately follows from work of Jaco and Rubinstein.

► **Theorem 10** (cf. Theorem 6.1 of [27]). *Every lens space admits a layered triangulation  $\mathcal{T}$  with the simplification of  $\Gamma(\mathcal{T})$  being a path. In particular, all 3-manifolds of Heegaard genus at most one have treewidth at most one.*

For the proof of the other direction, the starting point is the following observation.

► **Lemma 11.** *If the simplification of a 4-regular multigraph  $G$  is a tree, then it is a path.*

**Proof.** Let  $S(G)$  denote the simplification of  $G$ . Call an arc of  $S(G)$  *even* (resp. *odd*) if its corresponding multiple arc in  $G$  consist of an even (resp. odd) number of arcs. Let  $Odd(G)$  be the subgraph of  $S(G)$  consisting of all odd arcs. It follows from a straightforward parity argument that all nodes in  $Odd(G)$  have an even degree. In particular, if the set  $E(Odd(G))$  of arcs is nonempty, then it necessarily contains a cycle. However, this cannot happen as  $S(G)$  is a tree by assumption. Consequently, all arcs of  $S(G)$  must be even. This implies that every node of  $S(G)$  has degree at most 2 (otherwise there would be a node in  $G$  with degree  $> 4$ ), which in turn implies that  $S(G)$  is a path. ◀



**Figure 7** The only possible dual graphs (corresp. to closed 3-manifolds) of treewidth at most one.

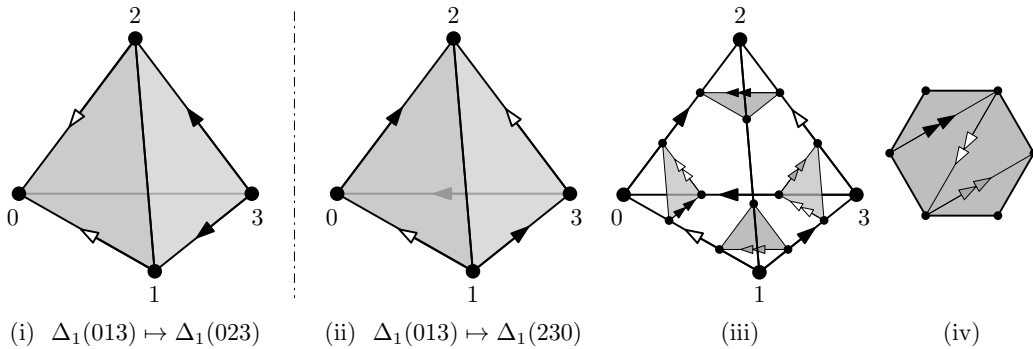
Consequently, if  $tw(\Gamma(\mathcal{T})) \leq 1$  for a triangulation  $\mathcal{T}$  of a closed 3-manifold, then  $\Gamma(\mathcal{T})$  is a “thick” path. If  $tw(\Gamma(\mathcal{T})) = 0$ , then  $\Gamma(\mathcal{T})$  is a single node with two loops (Figure 7(i)). By looking at the *Closed Census* [12], the only orientable 3-manifolds admitting a dual graph of this kind are  $\mathbb{S}^3$  and two lens spaces. If  $\Gamma(\mathcal{T})$  has a quadruple arc, then it must be a path of length two (Figure 7(ii)), and the only 3-manifold not a lens space appearing here is  $SFS[\mathbb{S}^2 : (2, 1), (2, 1), (2, -1)]$  which has Heegaard genus two, cf. [42, p. 27]. Otherwise, order the tetrahedra  $\Delta_1, \dots, \Delta_n$  of  $\mathcal{T}$  as shown in Figure 7(iii), and define  $\mathcal{T}_i \subset \mathcal{T}$  to be the  $i^{th}$  subcomplex of  $\mathcal{T}$  consisting of  $\Delta_1, \dots, \Delta_i$ .



$\mathcal{T}_1$  is obtained by identifying two triangles of  $\Delta_1$ . Without loss of generality, we may assume that these are the triangles  $\Delta_1(013)$  and  $\Delta_1(023)$ . A priori, there are six possible face gluings between them (corresponding to the six bijections  $\{0, 1, 2\} \rightarrow \{0, 2, 3\}$ ).

The gluing  $\Delta_1(013) \mapsto \Delta_1(023)$  yields a 3-vertex triangulation of the 3-ball, called a *snapped 3-ball*, and is an admissible choice for  $\mathcal{T}_1$ , Figure 8(i).  $\Delta_1(013) \mapsto \Delta_1(032)$  and  $\Delta_1(013) \mapsto \Delta_1(203)$  both create Möbius bands as vertex links of the vertices (0) and (2), respectively, and thus these 1-tetrahedron complexes cannot be subcomplexes of a 3-manifold triangulation.  $\Delta_1(013) \mapsto \Delta_1(230)$  and  $\Delta_1(013) \mapsto \Delta_1(302)$  both produce valid but isomorphic choices for  $\mathcal{T}_1$ : the minimal layered solid torus of type LST(1, 2, -3), Figure 8(ii). Lastly,  $\Delta_1(013) \mapsto \Delta_1(320)$  identifies the edge (03) with itself in reverse, it is hence invalid.

We discuss the two valid cases separately, starting with the latter one.



■ **Figure 8** The snapped 3-ball (i). A layered solid torus (ii), with four normal triangles comprising the single vertex link (iii), which is a triangulated hexagonal disk (iv).

► **Lemma 12.** *Let  $\mathcal{T}$  be a triangulation of a closed, orientable 3-manifold of treewidth one, with  $\mathcal{T}_1$  being a solid torus. Then  $\mathcal{T}$  triangulates a 3-manifold of Heegaard genus one.*

**Proof.** The proof consists of the following parts. **1.** We systematize all subcomplexes  $\mathcal{T}_2 \subset \mathcal{T}$  which arise from gluing a tetrahedron  $\Delta_2$  to  $\mathcal{T}_1$  along two triangular faces, and discard all complexes which cannot be part of a 3-manifold triangulation. **2.** We discuss the possible combinatorial types of subcomplexes  $\mathcal{T}_i$ ,  $i > 2$ , and triangulations of 3-manifolds arising from the remaining cases. **3.** We show for all resulting triangulations, that the fundamental group of the underlying manifold is, and is thus of Heegaard genus at most one.

To enumerate all possibilities for  $\mathcal{T}_2$ , assume, without loss of generality, that  $\mathcal{T}_1$  is obtained by  $\Delta_1(013) \mapsto \Delta_1(230)$ . The boundary  $\partial\mathcal{T}_1$  is built from two triangles  $(012)_\partial$  and  $(123)_\partial$ , sharing an edge (12), via the identifications  $(01) = (23)$  and  $(02) = (13)$ , see Figure 8(ii). The vertex link of  $\mathcal{T}_1$  is a triangulated hexagon as shown in Figure 8(iii)–(iv).

The second subcomplex  $\mathcal{T}_2$  is obtained from  $\mathcal{T}_1$  by gluing  $\Delta_2$  to the boundary of  $\mathcal{T}_2$  along two of its triangles. By symmetry, we are free to choose the first gluing. Hence, without loss of generality, let  $\mathcal{T}'_2$  be the complex obtained from  $\mathcal{T}_1$  by gluing  $\Delta_2$  to  $\mathcal{T}_1$  with gluing  $\Delta_2(012) \mapsto (012)_\partial$ . The result is a 2-vertex triangulated solid torus with four boundary triangles  $\Delta_2(013)$ ,  $\Delta_2(023)$ ,  $\Delta_2(123)$  and  $(123)_\partial$ , see Figure 9(ii). Since adjacent edges in the boundary of the unique vertex link of  $\mathcal{T}_1$  are always normal arcs in distinct triangles of  $\partial\mathcal{T}_1$ , the vertex links of  $\mathcal{T}'_2$  must be a triangulated 9-gon and a single triangle, see Figure 9(iii).

Note that both vertex links of  $\mathcal{T}'_2$  are symmetric with respect to the normal arcs coming from boundary triangles  $\Delta_2(013)$ ,  $\Delta_2(023)$  and  $\Delta_2(123)$ . By this symmetry, we are free to choose whether to glue  $\Delta_2(013)$ ,  $\Delta_2(023)$  or  $\Delta_2(123)$  to  $(123)_\partial$ , in order to obtain  $\mathcal{T}_2$ .

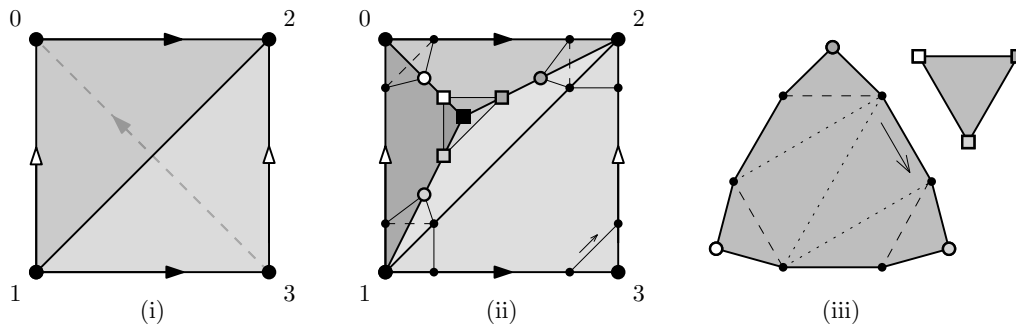


Figure 9 The solid torus  $\mathcal{T}_1$  (i), the complex  $\mathcal{T}_2'$  (ii), and the vertex links of  $\mathcal{T}_2'$  (iii).

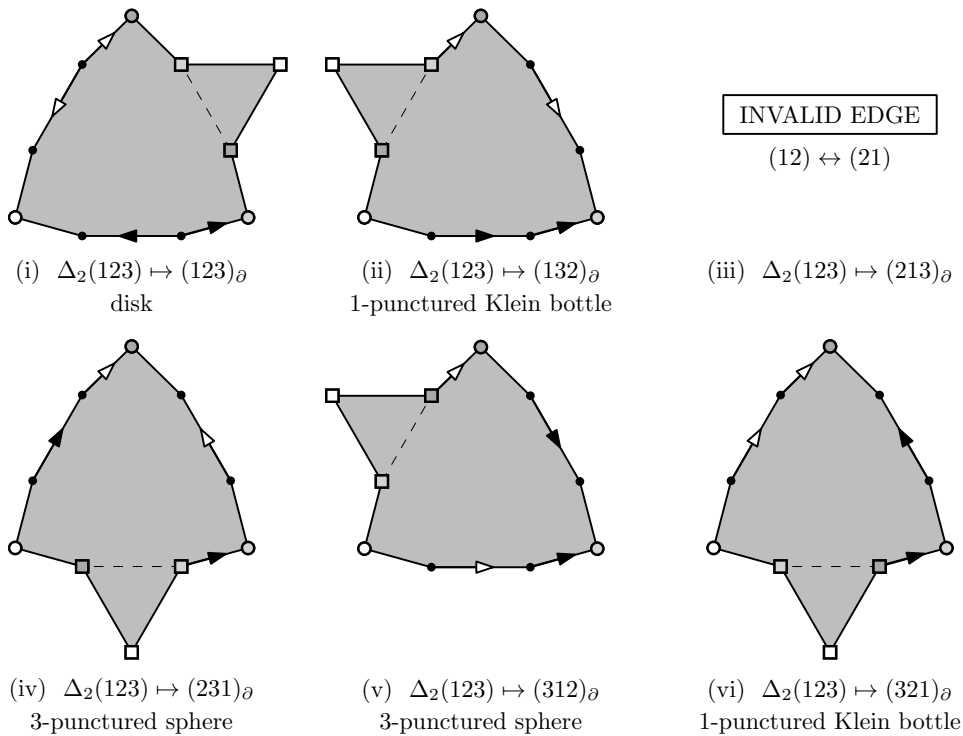


Figure 10

Therefore we have the following six possibilities to consider (Figure 10).

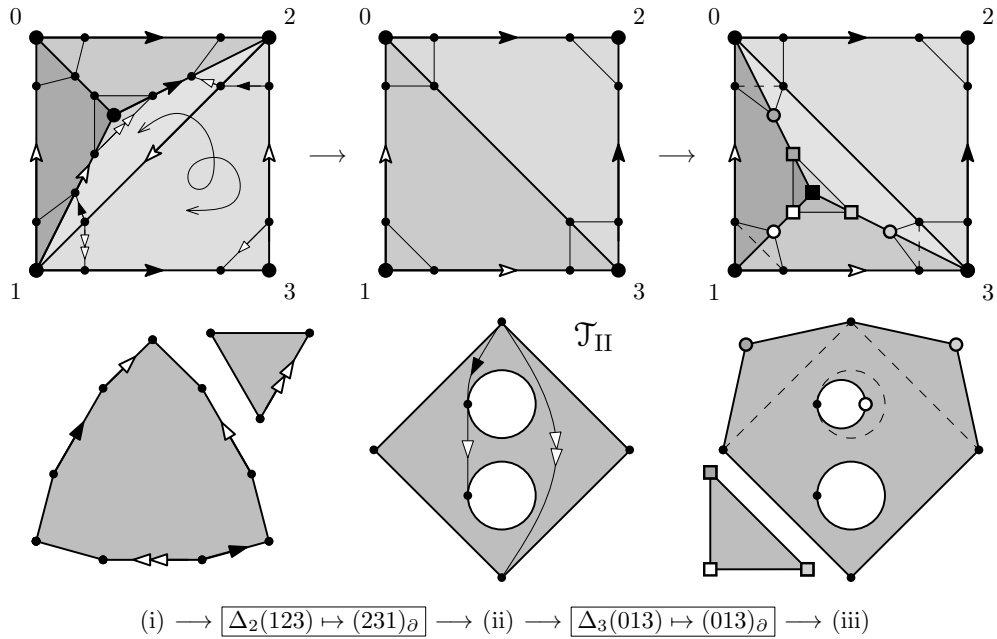
$\Delta_2(123) \mapsto (123)_\partial$  is a layering onto (12). It yields another layered solid torus with vertex link a triangulated hexagon with edges adjacent in the boundary of the link being normal arcs in distinct faces in  $\partial\mathcal{T}_2$ . Hence, in this case we have the same options for  $\mathcal{T}_3$  as the ones in this list. Any complex obtained by iterating this case is of this type.

Here, as well as for the remainder of this proof, whenever we obtain a subcomplex with all cases for the next subcomplex equal to a case already considered (i.e., isomorphic boundary complexes compatible with isomorphic boundaries of vertex links), we talk about these cases to be of the same *type*. We denote the one obtained via  $\Delta_2(123) \mapsto (123)_\partial$  by  $\mathcal{T}_1$ .

$\Delta_2(123) \mapsto (132)_\partial$  is invalid, as it creates a punctured Klein bottle as vertex link.

$\Delta_2(123) \mapsto (213)_\partial$  is invalid, as it identifies (12) on the boundary with itself in reverse.  
 $\Delta_2(123) \mapsto (231)_\partial$  results in a 1-vertex complex with triangles  $(013)_\partial$  and  $(023)_\partial$  comprising its boundary, which is isomorphic to the boundary of the snapped 3-ball with all of its three vertices identified. The vertex link is a 3-punctured sphere with two boundary components being normal loop arcs and one consisting of the remaining four normal arcs. This complex is discussed in detail below, and we denote its type by  $\mathcal{T}_{II}$ .  
 $\Delta_2(123) \mapsto (312)_\partial$  gives a 1-vertex complex of type  $\mathcal{T}_{II}$  as in the previous case.  
 $\Delta_2(123) \mapsto (321)_\partial$  is invalid: it produces a punctured Klein bottle in the vertex link.

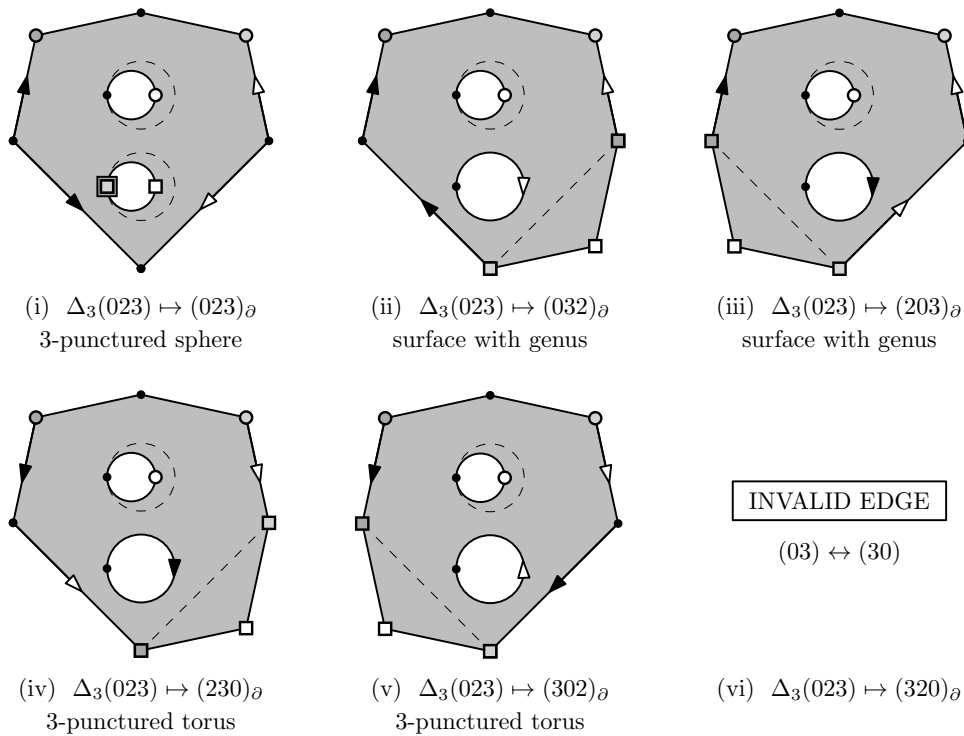
Now we discuss complexes of type  $\mathcal{T}_{II}$ . To this end, let  $\mathcal{T}_2$  be the complex in Figure 11(ii) defining this type. By gluing  $\Delta_3$  to  $\mathcal{T}_2$  along a boundary triangle, say  $\Delta_3(013) \mapsto (013)_\partial$ , we obtain a complex  $\mathcal{T}'_3$  (see Figure 11(iii)). Note that no boundary edge of the 3-punctured sphere vertex link  $\mathcal{L}$  can be identified with an edge in another boundary component of  $\mathcal{L}$ , for that would create genus in  $\mathcal{L}$  (an obstruction to being a subcomplex of a 3-manifold triangulation in which all vertex links must be  $S^2$ ). As shown in Figure 12, there is a unique gluing to avoid this, namely  $\Delta_3(023) \mapsto (023)_\partial$ , which yields a 1-vertex complex  $\mathcal{T}_3$  with vertex link still being a 3-punctured sphere, but now with three boundary components consisting of two edges each, as indicated in Figure 12(i). Let  $\mathcal{T}_{III}$  denote its type. Repeating the same argument for  $\mathcal{T}_3$  implies that a valid  $\mathcal{T}_4$  must be again of type  $\mathcal{T}_{II}$ .



■ Figure 11

Altogether, the type of each intermediate complex  $\mathcal{T}_i$  ( $i < n$ ) is either  $\mathcal{T}_I$  (layered solid torus), or one of the two types  $\mathcal{T}_{II}$  and  $\mathcal{T}_{III}$  of 1-vertex complexes with a 3-punctured sphere as vertex link and boundary isomorphic to that of the snapped 3-ball with all vertices identified. If  $\mathcal{T}_{n-1}$  is of type  $\mathcal{T}_I$ , then it can always be completed to a triangulation of a closed 3-manifold by either adding a minimal layered solid torus or a snapped 3-ball. If  $\mathcal{T}_{n-1}$  is of type  $\mathcal{T}_{II}$  or  $\mathcal{T}_{III}$ , then it may be completed – if possible – by adding a snapped 3-ball.

To conclude that any resulting  $\mathcal{T}$  triangulates a 3-manifold of Heegaard genus at most one, first we observe that the fundamental group of  $\pi_1(\mathcal{T})$  is generated by one element.



■ Figure 12

Indeed,  $\pi_1(\mathcal{T}_1)$  is isomorphic to  $\mathbb{Z}$  and is generated by a boundary edge. Furthermore, since  $\mathcal{T}_1$  only has one vertex, all edges in  $\mathcal{T}_1$  must be loop edges, and no edge which is trivial in  $\pi_1(\mathcal{T}_1)$  can become non-trivial in the process of building up the triangulation of the closed 3-manifold. When we extend  $\mathcal{T}_1$  by attaching further tetrahedra along two triangles each, then either all edges of the newly added tetrahedron are identified with edges of the previous complex, or – in case of a layering – the unique new boundary edge can be expressed in terms of the existing generator. In both cases, the fundamental group of the new complex admits a presentation with one generator. Moreover, no new generator can arise from inserting a minimal layered solid torus or snapped 3-ball in the last step.

So either  $\pi_1(\mathcal{T})$  is infinite cyclic, i.e., isomorphic to  $\mathbb{Z}$ , in which case  $\mathcal{T}$  must be a triangulation of  $\mathbb{S}^2 \times \mathbb{S}^1$  [22]; or  $\pi_1(\mathcal{T})$  is finite, but then it is spherical by the Geometrization Theorem [40, p. 104], and thus must be a lens space [47, Theorem 4.4.14.(a)]. ◀

► **Lemma 13.** *Let  $\mathcal{T}$  be an  $n$ -tetrahedron triangulation of a closed, orientable 3-manifold of treewidth one, with both  $\mathcal{T}_1$  and  $\mathcal{T} \setminus \mathcal{T}_{n-1}$  being a snapped 3-ball. Then  $\mathcal{T}$  triangulates a 3-manifold of Heegaard genus one.*

The proof follows the same structure as the one of Lemma 12 (cf. [24]).

## 5 Some 3-manifolds of treewidth two

In what follows, we use the classification of 3-manifolds of treewidth one (Theorem 2) to show that a large class of orientable Seifert fibered spaces and some *graph manifolds* have treewidth two. This is done by exhibiting appropriate triangulations, which have all the hallmarks of a space station. First, we give an overview of the building blocks.

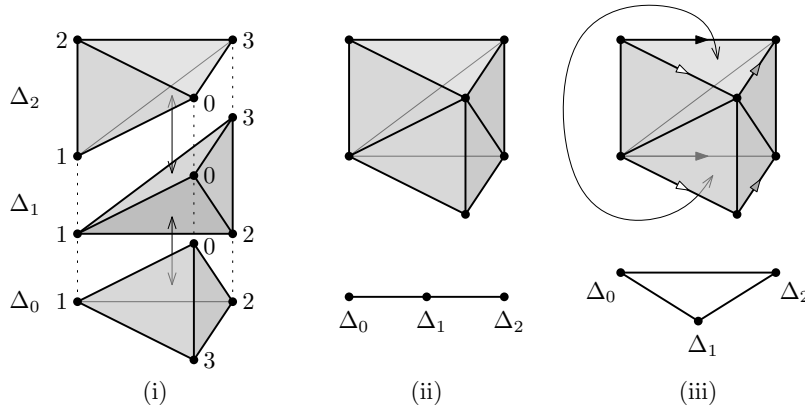
**Robotic arms.** These are just the layered triangulations of the solid torus with 2-triangle boundaries introduced in Section 3.2 and encountered in the proof of Theorem 2. Their dual graphs are thick paths, see Figure 3(v). A layered solid torus is of type  $LST(p, q, r)$  if its meridional disk intersects its boundary edges  $p, q$  and  $r$  times. For any coprime  $p, q, r$  with  $p + q + r = 0$ , a triangulation of type  $LST(p, q, r)$  can be realized by [10, Algorithm 1.2.17].

► **Example 14.** A special class of robotic arms are the ones of type  $LST(0, 1, 1)$ , as they can be used to trivially fill-in superfluous torus boundary components without inserting an unwanted exceptional fiber into a Seifert fibered space (cf. descriptions of  $\mathbb{A}_2$  and  $\mathbb{A}_1$  below). One of the standard triangulations of robotic arms of type  $LST(0, 1, 1)$  has three tetrahedra  $\Delta_i, 0 \leq i \leq 2$ , and is given by the gluing relations (4).

$$\begin{aligned} \Delta_0(023) &\mapsto \Delta_1(013), & \Delta_0(123) &\mapsto \Delta_1(120), & \Delta_1(023) &\mapsto \Delta_2(201), \\ \Delta_1(123) &\mapsto \Delta_2(301), & \Delta_2(023) &\mapsto \Delta_2(312). \end{aligned} \tag{4}$$

**Core unit with three docking sites.** Start with a triangle  $t$ , take the product  $t \times [0, 1]$ , triangulate it using three tetrahedra, Figure 13(i)–(ii), and glue  $t \times \{0\}$  to  $t \times \{1\}$  without a twist, Figure 13(iii). The dual graph of the resulting complex  $\mathbb{A}_3$  – topologically a solid torus – is  $K_3$ , hence of treewidth two. Its boundary – a 6-triangle triangulation of the torus – can be split into three 2-triangle annuli, corresponding to the edges of  $t$ , each of which we call a *docking site*. Edges running along a fiber and thus of type “vertex of base triangle cross circle” are termed *vertical edges*. Edges orthogonal to the fibers, i.e., the edges of  $t \times \{0\} = t \times \{1\}$ , are termed *horizontal edges*. The remaining edges are referred to as *diagonal edges*. More concisely, the triangulation of  $\mathbb{A}_3$  has gluing relations

$$\Delta_0(012) \mapsto \Delta_1(012), \quad \Delta_1(013) \mapsto \Delta_2(013), \quad \Delta_2(023) \mapsto \Delta_0(312). \tag{5}$$



■ **Figure 13** Construction of the core unit  $\mathbb{A}_3$  with three docking sites.

**Core assembly with  $r$  docking sites.** For  $r = 2$  (resp.  $r = 1$ ), take a core unit  $\mathbb{A}_3$  and glue a robotic arm of type  $LST(0, 1, 1)$  onto one (resp. two) of its docking sites such that the *unique boundary edge* of the robotic arm (i.e., the boundary edge which is only contained in one tetrahedron of the layered solid torus) is glued to a horizontal boundary edge of  $\mathbb{A}_3$  (see Example 14 for a detailed description of a particular triangulation of a layered solid torus of type  $LST(0, 1, 1)$  with unique boundary edge being  $\Delta_0(01)$ ). The resulting complex

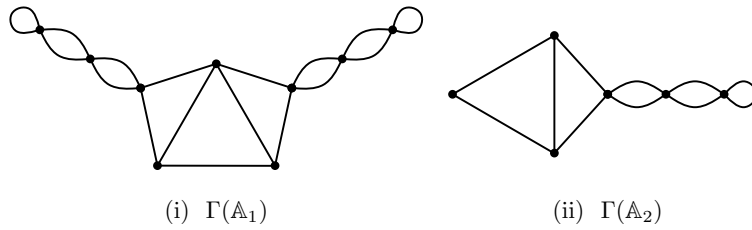


Figure 14

is denoted by  $A_2$  (resp.  $A_1$ ) and their dual graphs are shown in Figure 13. Observe that they have treewidth two.

For  $A_r$  ( $r \geq 3$ ) take  $r - 2$  copies of  $A_3$ , denote them by  $A_3^i$ ,  $1 \leq i \leq r - 2$  with tetrahedra  $\Delta_0^i$ ,  $\Delta_1^i$  and  $\Delta_2^i$ ,  $1 \leq i \leq r - 2$ . Glue them together by mirroring them across one of their boundary components as shown by Equation (6) for  $1 \leq i \leq r - 3$  odd, and by Equation (7) for  $2 \leq i \leq r - 3$  even. See also Figure 15 on the top for the case  $r = 5$ .

The resulting complex, denoted by  $A_r$ , has  $2r$  boundary triangles which become  $r$  2-triangle torus boundary components once identifications of vertical edges are introduced by gluing complexes with 2-triangle torus boundary components to them.

$$i \text{ odd: } \Delta_1^i(123) \mapsto \Delta_1^{i+1}(123), \quad \Delta_2^i(123) \mapsto \Delta_2^{i+1}(123). \tag{6}$$

$$i \text{ even: } \Delta_0^i(023) \mapsto \Delta_0^{i+1}(023), \quad \Delta_1^i(023) \mapsto \Delta_1^{i+1}(023). \tag{7}$$

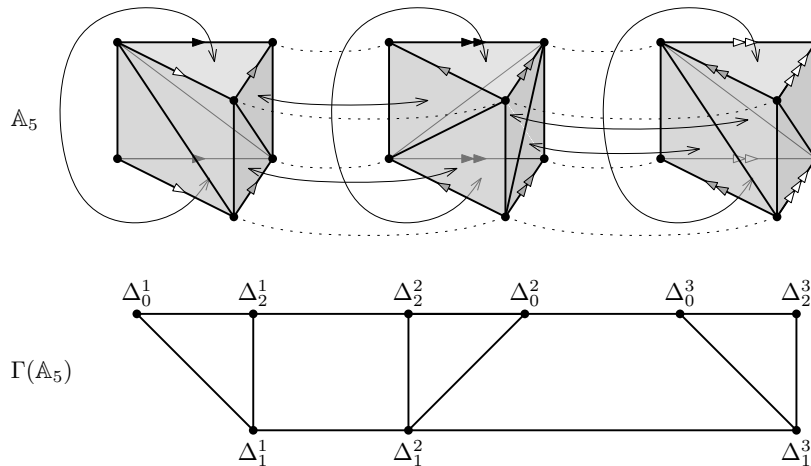
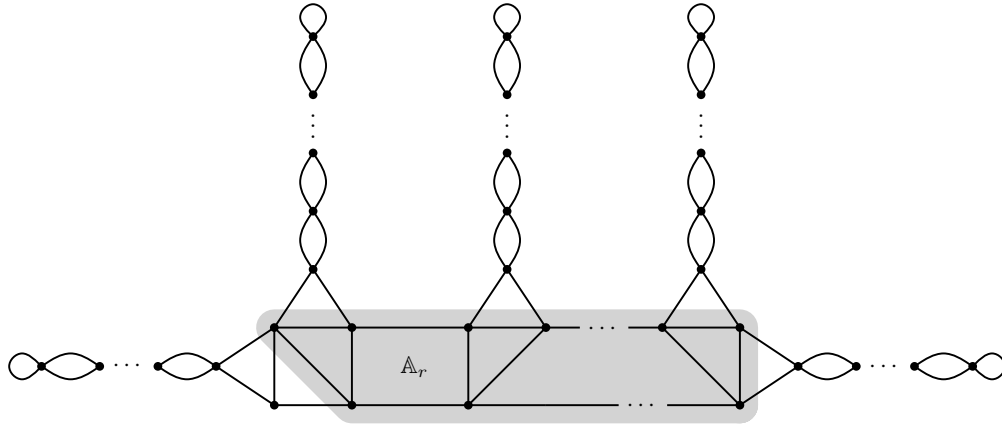


Figure 15

**Möbius laboratory module.** Such a complex, denoted by  $M$ , is given by

$$\begin{aligned} T_0(123) \mapsto T_1(123), \quad T_0(023) \mapsto T_1(031), \quad T_1(012) \mapsto T_2(201) \\ T_1(023) \mapsto T_2(023), \quad T_0(013) \mapsto T_2(132). \end{aligned} \tag{8}$$

Its dual graph is a triangle with two double edges, and hence of treewidth two (see, for instance, Figure 17).  $M$  has one torus boundary component given by the two triangles  $T_0(012)$  and  $T_2(013)$  with edges  $T_0(01) = T_2(13)$ ,  $T_0(02) = T_2(03)$ , and  $T_0(12) = T_2(01)$ .



■ **Figure 16** Dual graph of treewidth two triangulation of an orientable SFS over  $\mathbb{S}^2$ .

► **Theorem 15.** *Orientable Seifert fibered spaces over  $\mathbb{S}^2$  have treewidth at most two.*

**Proof.** To obtain a treewidth two triangulation of  $\text{SFS}[\mathbb{S}^2 : (a_1, b_1), \dots, (a_r, b_r)]$ , start with the core assembly  $\mathbb{A}_r$  and a collection of robotic arms  $\text{LST}(a_i, \pm|b_i|, -a_i \mp |b_i|)$ ,  $1 \leq i \leq r$ . The robotic arms are then glued to the  $r$  docking sites (2-triangle torus boundary components, separated by the vertical boundary edges) of  $\mathbb{A}_r$ , such that boundary edges of type  $a_i$  are glued to vertical boundary edges, and edges of type  $\pm|b_i|$  are glued to horizontal boundary edges. The sign in  $\text{LST}(a_i, \pm|b_i|, -a_i \mp |b_i|)$  is then determined by the type of diagonal edge in the  $i^{\text{th}}$  docking site of  $\mathbb{A}_r$  and by the sign of  $b_i$ . ◀

See Figure 16 for a picture of the dual graph of the resulting complex. It is of treewidth two. Note that, in some cases, a fibre of type  $(2, 1)$  can be realized by directly identifying the two triangles of a torus boundary component of  $\mathbb{A}_r$  with a twist. In the dual graph this then appears as a double edge rather than the attachment of a thick path. See Figure 18 on the right for an example in the treewidth two triangulation of the Poincaré homology sphere.

► **Theorem 16.** *An orientable SFS over a non-orientable surface is of treewidth at most two.*

**Proof.** To obtain a treewidth two triangulation of the orientable Seifert fibered space  $\text{SFS}[\mathcal{N}_g : (a_1, b_1), \dots, (a_r, b_r)]$  over the non-orientable surface  $\mathcal{N}_g$  of genus  $g$ , start with a core assembly  $\mathbb{A}_{r+g}$  and attach  $g$  copies  $\mathbb{M}_j$ ,  $1 \leq j \leq g$ , of the Möbius laboratory module via

$$T_0^j(012) \mapsto \Delta_2^j(201) \quad \text{and} \quad T_2^j(013) \mapsto \Delta_0^j(013), \tag{9}$$

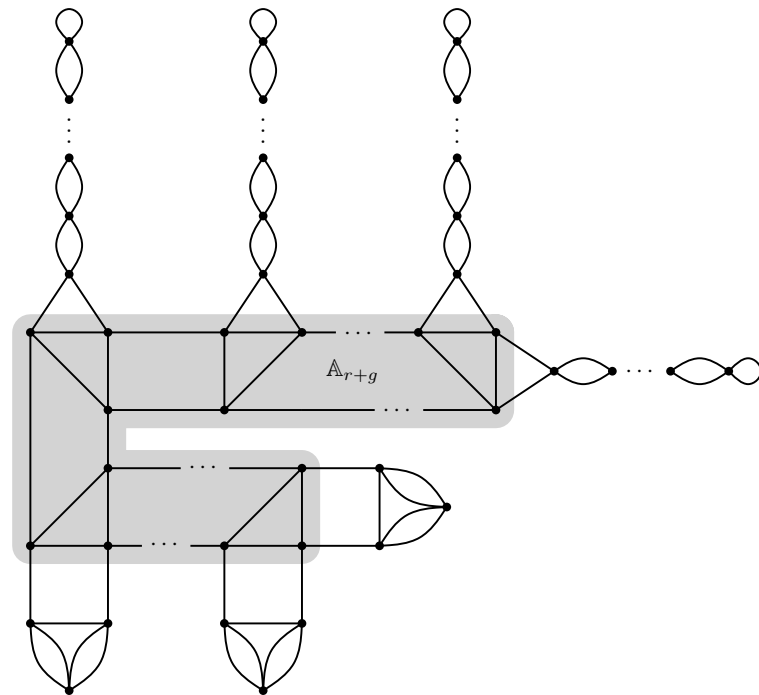
where  $T_0^j$ ,  $T_1^j$  and  $T_2^j$  are the tetrahedra comprising  $\mathbb{M}_j$ , and  $\Delta_0^j$ ,  $\Delta_1^j$  and  $\Delta_2^j$  denote the tetrahedra making up the first  $g$  core units (each being a copy of  $\mathbb{A}_3$ ) in  $\mathbb{A}_{r+g}$ .

Proceed by attaching a robotic arm of type  $\text{LST}(a_i, \pm|b_i|, -a_i \mp |b_i|)$ ,  $1 \leq i \leq r$ , to each of the remaining  $r$  docking sites. Again, for the gluings between the robotic arms and the core assembly  $\mathbb{A}_{r+g}$ , the edges of type  $a_i$  must be glued to the vertical boundary edges, the edges of type  $b_i$  must be glued to the horizontal boundary edges, and attention has to be paid to the signs of the  $b_i$  and to how exactly diagonal edges run. See Figure 17 for a picture of the dual graph of the resulting complex, which is of treewidth two by inspection. ◀

Combining Theorems 2, 15 and 16 immediately gives the following statement.

► **Corollary 17.** *An orientable Seifert fibered space  $\mathcal{M}$  over  $\mathbb{S}^2$  or a non-orientable surface is of treewidth one, if  $\mathcal{M}$  is a lens space or  $\text{SFS}[\mathbb{S}^2 : (2, 1), (2, 1), (2, -1)]$ , and two otherwise.*





■ **Figure 17** Dual graph of a treewidth two triangulation of an orientable SFS over  $\mathcal{N}_g$ .

► **Corollary 18.** *Orientable spherical or “ $\mathbb{S}^2 \times \mathbb{R}$ ” 3-manifolds are of treewidth at most two.*

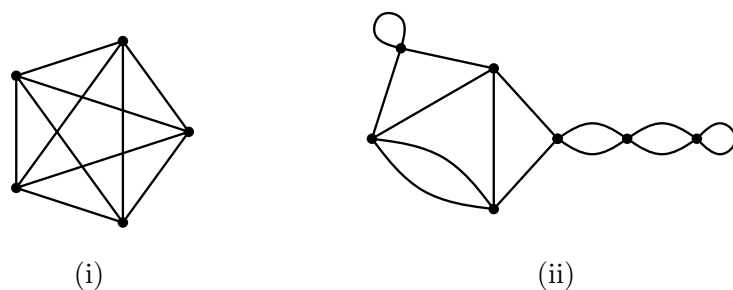
This follows directly from Theorems 15 and 16, see also [24].

► **Corollary 19.** *Graph manifolds  $\mathcal{M}_T$  modeled on a tree  $T$  with nodes being orientable Seifert fibered spaces over  $\mathbb{S}^2$  or  $\mathcal{N}_g$ ,  $g > 0$ , have treewidth at most two.*

See [24] for a proof of this statement and an example.

► **Corollary 20.** *Minimal triangulations are not always of minimum treewidth.*

**Proof.** The Poincaré homology sphere  $\Sigma^3 = \text{SFS}[\mathbb{S}^2 : (2, 1), (3, 1), (5, -4)]$  has treewidth two but its minimal triangulation has treewidth four, see Figure 18. ◀



■ **Figure 18** Dual graph of the minimal (i), and of a treewidth two (ii) triangulation of  $\Sigma^3$ .

► **Corollary 21.** *There exist irreducible 3-manifolds with treewidth two, but arbitrarily high Heegaard genus.*

This now follows from [9, Theorem 1.1 (i)], see the full version [24] for a proof.

Using Theorems 2, 15 and 16 we can now determine the treewidth of most of the 3-manifolds from the 10-tetrahedra census, see Table 1.

■ **Table 1** The 4979 3-manifolds triangulable with  $\leq 10$  tetrahedra and their treewidths.

| $n$      | # mfd. $M$ | $\text{tw}(M) = 0$ | $\text{tw}(M) = 1$ | $\text{tw}(M) = 2$ | unknown |
|----------|------------|--------------------|--------------------|--------------------|---------|
| 1        | 3          | 3                  | 0                  | 0                  | 0       |
| 2        | 7          | 0                  | 7                  | 0                  | 0       |
| 3        | 7          | 0                  | 6                  | 1                  | 0       |
| 4        | 14         | 0                  | 10                 | 4                  | 0       |
| 5        | 31         | 0                  | 20                 | 11                 | 0       |
| 6        | 74         | 0                  | 36                 | 36                 | 2       |
| 7        | 175        | 0                  | 72                 | 100                | 3       |
| 8        | 436        | 0                  | 136                | 297                | 3       |
| 9        | 1154       | 0                  | 272                | 861                | 21      |
| 10       | 3078       | 0                  | 528                | 2489               | 61      |
| $\Sigma$ | 4979       | 3                  | 1087               | 3799               | 90      |

► **Remark 22.** Using similar constructions it can be shown that orientable Seifert fibered spaces over orientable surfaces have treewidth at most four. However, since this only provides an upper bound rather than determining the treewidth of some family of 3-manifolds, we refer the reader to the full version of this article for a detailed description of these triangulations.

---

## References

- 1 D. Bachman, R. Derby-Talbot, and E. Sedgwick. Computing Heegaard genus is NP-hard. In M. Loebli, J. Nešetřil, and R. Thomas, editors, *A Journey Through Discrete Mathematics: A Tribute to Jiří Matoušek*, pages 59–87. Springer, 2017. doi:10.1007/978-3-319-44479-6.
- 2 M. C. Bell. *Recognising mapping classes*. PhD thesis, University of Warwick, June 2015. URL: <https://wrap.warwick.ac.uk/77123>.
- 3 L. Bessières, G. Besson, S. Maillot, M. Boileau, and J. Porti. *Geometrisation of 3-manifolds*, volume 13 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2010. doi:10.4171/082.
- 4 R. H. Bing. An alternative proof that 3-manifolds can be triangulated. *Ann. of Math. (2)*, 69:37–65, 1959. doi:10.2307/1970092.
- 5 H. L. Bodlaender. A Tourist Guide through Treewidth. *Acta Cybern.*, 11(1-2):1–21, 1993. URL: [https://www.inf.u-szeged.hu/actacybernetica/edb/vol11n1\\_2/Bodlaender\\_1993\\_ActaCybernetica.xml](https://www.inf.u-szeged.hu/actacybernetica/edb/vol11n1_2/Bodlaender_1993_ActaCybernetica.xml).
- 6 H. L. Bodlaender. A Partial  $k$ -Arboretum of Graphs with Bounded Treewidth. *Theor. Comput. Sci.*, 209(1-2):1–45, 1998. doi:10.1016/S0304-3975(97)00228-4.
- 7 H. L. Bodlaender. Discovering Treewidth. In *SOFSEM 2005: Theory and Practice of Computer Science, 31st Conference on Current Trends in Theory and Practice of Computer Science, Liptovský Ján, Slovakia, January 22–28, 2005, Proceedings*, pages 1–16, 2005. doi:10.1007/978-3-540-30577-4\_1.
- 8 H. L. Bodlaender and A. M. C. A. Koster. Combinatorial Optimization on Graphs of Bounded Treewidth. *Comput. J.*, 51(3):255–269, 2008. doi:10.1093/comjnl/bxm037.

- 9 M. Boileau and H. Zieschang. Heegaard genus of closed orientable Seifert 3-manifolds. *Invent. Math.*, 76(3):455–468, 1984. doi:10.1007/BF01388469.
- 10 B. A. Burton. *Minimal triangulations and normal surfaces*. PhD thesis, University of Melbourne, September 2003. URL: <https://people.smp.uq.edu.au/BenjaminBurton/papers/2003-thesis.html>.
- 11 B. A. Burton. Computational topology with Regina: algorithms, heuristics and implementations. In *Geometry and topology down under*, volume 597 of *Contemp. Math.*, pages 195–224. Amer. Math. Soc., Providence, RI, 2013. doi:10.1090/conm/597/11877.
- 12 B. A. Burton, R. Budney, W. Pettersson, et al. Regina: Software for low-dimensional topology, 1999–2017. URL: <https://regina-normal.github.io>.
- 13 B. A. Burton and R. G. Downey. Courcelle’s theorem for triangulations. *J. Comb. Theory, Ser. A*, 146:264–294, 2017. doi:10.1016/j.jcta.2016.10.001.
- 14 B. A. Burton, T. Lewiner, J. Paixão, and J. Spreer. Parameterized Complexity of Discrete Morse Theory. *ACM Trans. Math. Softw.*, 42(1):6:1–6:24, 2016. doi:10.1145/2738034.
- 15 B. A. Burton, C. Maria, and J. Spreer. Algorithms and complexity for Turaev–Viro invariants. *J. Appl. Comput. Topol.*, 2(1–2):33–53, 2018. doi:10.1007/s41468-018-0016-2.
- 16 B. A. Burton and J. Spreer. The complexity of detecting taut angle structures on triangulations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6–8, 2013*, pages 168–183, 2013. doi:10.1137/1.9781611973105.13.
- 17 A. de Mesmay, J. Purcell, S. Schleimer, and E. Sedgwick. On the tree-width of knot diagrams, 2018. 13 pages, 4 figures. arXiv:1809.02172.
- 18 J. Díaz, J. Petit, and M. J. Serna. A survey of graph layout problems. *ACM Comput. Surv.*, 34(3):313–356, 2002. doi:10.1145/568522.568523.
- 19 R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Berlin, fifth edition, 2017. doi:10.1007/978-3-662-53622-3.
- 20 R. G. Downey and M. R. Fellows. *Fundamentals of parameterized complexity*. Texts in Computer Science. Springer, London, 2013. doi:10.1007/978-1-4471-5559-1.
- 21 R. S. Hamilton. Three-manifolds with positive Ricci curvature. *J. Differential Geom.*, 17(2):255–306, 1982. doi:10.4310/jdg/1214436922.
- 22 A. Hatcher. Notes on Basic 3-Manifold Topology, 2007. URL: <https://pi.math.cornell.edu/~hatcher/3M/3Mfds.pdf>.
- 23 J. Hempel. *3-manifolds*. AMS Chelsea Publishing, Providence, RI, 2004. Reprint of the 1976 original. doi:10.1090/chel/349.
- 24 K. Huszár and J. Spreer. 3-Manifold triangulations with small treewidth, 2018. 34 pages, 30 figures, 1 table. arXiv:1812.05528.
- 25 K. Huszár, J. Spreer, and U. Wagner. On the treewidth of triangulated 3-manifolds, 2017. 25 pages, 6 figures, 1 table. An extended abstract of this paper appeared in the Proceedings of the 34th International Symposium on Computational Geometry (SoCG 2018), Budapest, June 11–14 2018. arXiv:1712.00434.
- 26 W. Jaco. *Lectures on three-manifold topology*, volume 43 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, Providence, R.I., 1980. doi:10.1090/cbms/043.
- 27 W. Jaco and J. H. Rubinstein. Layered-triangulations of 3-manifolds, 2006. 97 pages, 32 figures. arXiv:math/0603601.
- 28 B. Kleiner and J. Lott. Notes on Perelman’s papers. *Geom. Topol.*, 12(5):2587–2855, 2008. doi:10.2140/gt.2008.12.2587.
- 29 L. Lovász. Graph minor theory. *Bull. Amer. Math. Soc. (N.S.)*, 43(1):75–86, 2006. doi:10.1090/S0273-0979-05-01088-8.
- 30 F. H. Lutz. Triangulated Manifolds with Few Vertices: Geometric 3-Manifolds, 2003. 48 pages, 18 figures. arXiv:math/0311116.
- 31 C. Maria and J. Purcell. Treewidth, crushing, and hyperbolic volume, 2018. 20 pages, 12 figures. arXiv:1805.02357.

- 32 C. Maria and J. Spreer. A polynomial time algorithm to compute quantum invariants of 3-manifolds with bounded first Betti number. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16–19*, pages 2721–2732, 2017. doi:10.1137/1.9781611974782.180.
- 33 J. Milnor. Towards the Poincaré conjecture and the classification of 3-manifolds. *Notices Amer. Math. Soc.*, 50(10):1226–1233, 2003.
- 34 E. E. Moise. Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. *Ann. of Math. (2)*, 56:96–114, 1952. doi:10.2307/1969769.
- 35 J. M. Montesinos. *Classical tessellations and three-manifolds*. Universitext. Springer-Verlag, Berlin, 1987. doi:10.1007/978-3-642-61572-6.
- 36 P. Orlik. *Seifert manifolds*, volume 291 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin-New York, 1972. doi:10.1007/BFb0060329.
- 37 G. Perelman. The entropy formula for the Ricci flow and its geometric applications, 2002. 39 pages. arXiv:math/0211159.
- 38 G. Perelman. Finite extinction time for the solutions to the Ricci flow on certain three-manifolds, 2003. 7 pages. arXiv:math/0307245.
- 39 G. Perelman. Ricci flow with surgery on three-manifolds, 2003. 22 pages. arXiv:math/0303109.
- 40 J. Porti. Geometrization of three manifolds and Perelman’s proof. *Rev. R. Acad. Cienc. Exactas Fís. Nat. Ser. A Math. RACSAM*, 102(1):101–125, 2008. doi:10.1007/BF03191814.
- 41 N. Robertson and P. D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-Width. *J. Algorithms*, 7(3):309–322, 1986. doi:10.1016/0196-6774(86)90023-4.
- 42 N. Saveliev. *Lectures on the topology of 3-manifolds*. De Gruyter Textbook. Walter de Gruyter & Co., Berlin, 2nd edition, 2012. doi:10.1515/9783110250367.
- 43 M. Scharlemann. Heegaard splittings of compact 3-manifolds. In *Handbook of geometric topology*, pages 921–953. North-Holland, Amsterdam, 2002. doi:10.1016/B978-044482432-5/50019-6.
- 44 J. Schultens. *Introduction to 3-manifolds*, volume 151 of *Graduate Studies in Mathematics*. Amer. Math. Soc., Providence, RI, 2014. doi:10.1090/gsm/151.
- 45 P. Scott. The geometries of 3-manifolds. *Bull. London Math. Soc.*, 15(5):401–487, 1983. doi:10.1112/blms/15.5.401.
- 46 H. Seifert. Topologie Dreidimensionaler Gefaserner Räume. *Acta Math.*, 60(1):147–238, 1933. doi:10.1007/BF02398271.
- 47 W. P. Thurston. Three-dimensional manifolds, Kleinian groups and hyperbolic geometry. *Bull. Amer. Math. Soc. (N.S.)*, 6(3):357–381, 1982. doi:10.1090/S0273-0979-1982-15003-0.
- 48 W. P. Thurston. *Three-dimensional geometry and topology. Vol. 1*, volume 35 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1997. Edited by S. Levy. doi:10.1515/9781400865321.

# Algorithms for Metric Learning via Contrastive Embeddings

Diego Ihara

University of Illinois at Chicago, USA  
<https://dihara2.people.uic.edu/>  
dihara2@uic.edu

Neshat Mohammadi

University of Illinois at Chicago, USA  
nmoham24@uic.edu

Anastasios Sidiropoulos

University of Illinois at Chicago, USA  
<http://sidiropoulos.org>  
sidiropo@uic.edu

---

## Abstract

We study the problem of supervised learning a metric space under *discriminative* constraints. Given a universe  $X$  and sets  $\mathcal{S}, \mathcal{D} \subset \binom{X}{2}$  of *similar* and *dissimilar* pairs, we seek to find a mapping  $f : X \rightarrow Y$ , into some target metric space  $M = (Y, \rho)$ , such that similar objects are mapped to points at distance at most  $u$ , and dissimilar objects are mapped to points at distance at least  $\ell$ . More generally, the goal is to find a mapping of maximum *accuracy* (that is, fraction of correctly classified pairs). We propose approximation algorithms for various versions of this problem, for the cases of Euclidean and tree metric spaces. For both of these target spaces, we obtain fully polynomial-time approximation schemes (FPTAS) for the case of perfect information. In the presence of imperfect information we present approximation algorithms that run in quasi-polynomial time (QPTAS). We also present an exact algorithm for learning line metric spaces with perfect information in polynomial time. Our algorithms use a combination of tools from metric embeddings and graph partitioning, that could be of independent interest.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** metric learning, contrastive distortion, embeddings, algorithms

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.45

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1807.04881>.

**Funding** Supported by NSF under award CAREER 1453472, and grants CCF 1815145 and CCF 1423230.

## 1 Introduction

Geometric algorithms have given rise to a plethora of tools for data analysis, such as clustering, dimensionality reduction, nearest-neighbor search, and so on; we refer the reader to [20, 19, 25] for an exposition. A common aspect of these methods is that the underlying data is interpreted as a metric space. That is, each object in the input is treated as a point, and the pairwise dissimilarity between pairs of objects is encoded by a distance function on pairs of points. An important element that can critically determine the success of this data analytic framework, is the choice of the actual metric. Broadly speaking, the area of *metric learning* is concerned with methods for recovering an underlying metric space that agrees with a given set of observations (we refer the reader to [29, 23] for a detailed exposition). The problem of learning the distance function is cast as an optimization problem, where the objective function quantifies the extend to which the solution satisfies the input constraints.



© Diego Ihara, Neshat Mohammadi, and Anastasios Sidiropoulos;  
licensed under Creative Commons License CC-BY

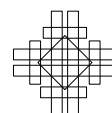
35th International Symposium on Computational Geometry (SoCG 2019).

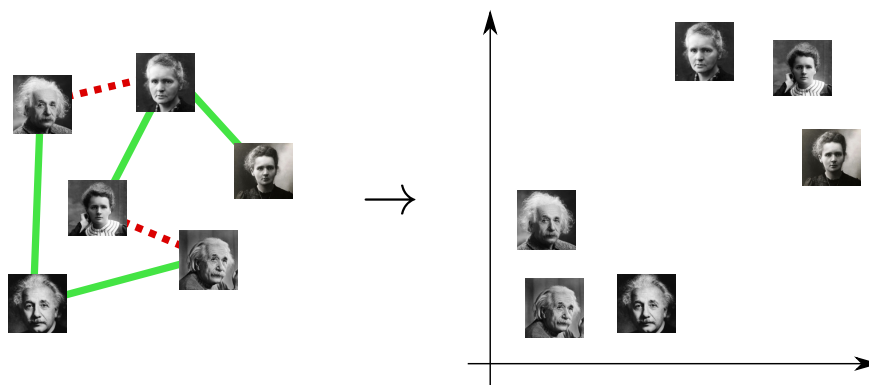
Editors: Gill Barequet and Yusu Wang; Article No. 45; pp. 45:1–45:14

Leibniz International Proceedings in Informatics



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





■ **Figure 1** An illustration of the metric learning framework. The input consists of a set of objects (here, a set of photos), with some pairs labeled as “similar” (depicted in green), and some pairs labeled “dissimilar” (depicted in red). The output is an embedding into some metric space (here, the Euclidean plane), such that similar objects are mapped to nearby points, while dissimilar objects are mapped to points that are far from each other.

**Supervised vs. unsupervised metric learning.** The problems studied in the context of metric learning generally fall within two main categories: *supervised* and *unsupervised* learning. In the case of unsupervised metric learning, the goal is to discover the intrinsic geometry of the data, or to fit the input into some metric space with additional structure. A prototypical example of an unsupervised metric learning problem is dimensionality reduction, where one is given a high-dimensional point set and the goal is to find a mapping into some space of lower dimension, or with a special structure, that approximately preserves the geometry of the input (see e.g. [15, 26, 12]).

In contrast, the input in a supervised metric learning problem includes *label constraints*, which encode some prior knowledge about the ground truth. As an example, consider the case of a data set where experts have labeled some pairs of points as being “similar” and some pairs as being “dissimilar”. Then, a metric learning task is to find a transformation of the input such that similar points end up close together, while dissimilar points end up far away from each other. As an illustrative example of the above definition, consider the problem of recognizing a face from a photo. More concretely, a typical input consists of some *universe*  $X$ , which is a set of photos of faces, together with some  $\mathcal{S}, \mathcal{D} \subseteq \binom{X}{2}$ . The set  $\mathcal{S}$  consists of pairs of photos that correspond to the same person, while  $\mathcal{D}$  consists of pairs of photos from different people. A typical approach for addressing this problem (see e.g. [14]) is to find a mapping  $f : X \rightarrow \mathbb{R}^d$ , for some  $d \in \mathbb{N}$ , such that for all similar pairs  $\{x, y\} \in \mathcal{S}$  we have  $\|f(x) - f(y)\|_2 \leq u$ , and for all dissimilar pairs  $\{x, y\} \in \mathcal{D}$  we have  $\|f(x) - f(y)\|_2 \geq \ell$ , for some  $u, \ell > 0$ . More generally, one seeks to find a mapping  $f$  that maximizes the fraction of correctly classified pairs of photos (see Figure 1).

## 1.1 Problem formulation

At the high level, an instance of a metric learning problem consists of some universe of objects  $X$ , together with some similarity information on subsets of these objects. Here, we focus on similarity and dissimilarity constraints. Specifically, an instance is a tuple  $\phi = (X, \mathcal{S}, \mathcal{D}, u, \ell)$ , where  $X$  is a finite set, with  $|X| = n$ ,  $\mathcal{S}, \mathcal{D} \subseteq \binom{X}{2}$ , which are sets of pairs of objects that are labeled as “similar” and “dissimilar” respectively, and  $u, \ell > 0$ . We refer to the elements of  $\mathcal{S} \cup \mathcal{D}$  as *constraints*. We focus on the case where  $\mathcal{S} \cap \mathcal{D} = \emptyset$ , and  $\mathcal{S} \cup \mathcal{D} = \binom{X}{2}$ . Let

$f : X \rightarrow Y$  be a mapping into some target metric space  $(Y, \rho)$ . As it is typical with geometric realization problems, we relax the definition to allow for a small multiplicative error  $c \geq 1$  in the embedding. We say that  $f$  *satisfies*  $\{x, y\} \in \mathcal{S}$ , if

$$\rho(f(x), f(y)) \leq u \cdot c, \quad (1)$$

and we say that it satisfies  $\{x, y\} \in \mathcal{D}$  if

$$\rho(f(x), f(y)) \geq \ell/c. \quad (2)$$

If  $f$  does not satisfy some  $\{x, y\} \in \mathcal{S} \cup \mathcal{D}$ , then we say that it *violates* it. We refer to the parameter  $c$  as the *contrastive distortion* of  $f$ . We also refer to  $f$  as a *contrastive embedding*, or *c-embedding*.

## 1.2 Our contribution

We now briefly discuss our main contributions.

We focus on the problem of computing an embedding with low contrastive distortion, and with maximum *accuracy*, which is defined to be the fraction of satisfied constraints. Since we are assuming *complete information*, that is  $\mathcal{S} \cup \mathcal{D} = \binom{X}{2}$ , it follows that the accuracy is equal to  $k/\binom{n}{2}$ , where  $k$  is the number of satisfied constraints. We remark that the setting of complete information requires a *dense* set of constraints. This case is important in applications when learning a metric space based on a set of objects with fully-labeled pairs. This is also the setting of other important machine learning primitives, such as Correlation Clustering (see [7]), which we discuss in Section 1.4.

Our results are concerned with two main cases: *perfect* and *imperfect* information. Here, the case of perfect information corresponds to the promise problem where there exists an embedding that satisfies all constraints. On the other hand, in the case of imperfect information, no such promise is given. As we shall see, the latter scenario appears to be significantly more challenging. Our results are concerned with two different families of target spaces:  $d$ -dimensional Euclidean space, and trees.

**Learning Euclidean metric spaces.** We begin our investigation by observing that the problem of computing a contrastive embedding into  $d$ -dimensional Euclidean space with perfect information is polynomial-time solvable for  $d = 1$ , and it becomes NP-hard even for  $d = 2$ . The result for  $d = 1$  is obtained via a simple greedy algorithm, while the NP-hardness proof uses a standard reduction from the problem of recognizing unit disk graphs (see [11]).

► **Theorem 1** (Learning the line with perfect information). *Let  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the metric learning problem. Then there exists a polynomial-time algorithm which given  $\gamma$ , either computes an 1-embedding  $\hat{f} : X \rightarrow \mathbb{R}$ , with accuracy 1, or correctly decides that no such embedding exists.*

► **Theorem 2** (Hardness of learning the plane with perfect information). *Given an instance  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  of the metric learning problem, it is NP-hard to decide whether there exists an 1-embedding  $\hat{f} : X \rightarrow \mathbb{R}^2$ , with accuracy 1.*

The above two results indicate that, except for what is essentially its simplest possible case, the problem is generally intractable. This motivates the study of approximation algorithms. Our first main result in this direction is a FPTAS for the case of perfect information, summarized in the following.



► **Theorem 3** (Learning Euclidean metric spaces with perfect information). *Let  $u, \ell > 0$  be fixed constants. Let  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the problem of learning a  $d$ -dimensional Euclidean metric space with perfect information, with  $|X| = n$ , for some  $d \geq 2$ . Suppose that  $\gamma$  admits an 1-embedding  $f^* : X \rightarrow \mathbb{R}^d$  with accuracy 1. Then for any  $\varepsilon, \varepsilon' > 0$ , there exists a randomized algorithm which given  $\gamma, \varepsilon$ , and  $\varepsilon'$ , computes a  $(1 + \varepsilon')$ -embedding  $\hat{f} : X \rightarrow \mathbb{R}^d$ , with accuracy at least  $1 - \varepsilon$ , in time  $n^{O(1)}g(\varepsilon, \varepsilon', d) = n^{O(1)}2^{(\frac{d}{\varepsilon\varepsilon'})^{O(d^2)}}$ , with high probability. In particular, for any fixed  $d, \varepsilon$ , and  $\varepsilon'$ , the running time is polynomial.*

We also obtain the following QPTAS for the case of imperfect information.

► **Theorem 4** (Learning Euclidean metric spaces with imperfect information). *Let  $d \geq 1$ . Let  $u, \ell > 0$  be fixed constants. Let  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the problem of learning a  $d$ -dimensional Euclidean metric space, with  $|X| = n$ , for some  $d \geq 1$ . Suppose that  $\gamma$  admits an 1-embedding  $f^* : X \rightarrow \mathbb{R}^d$  with accuracy  $1 - \zeta$ , for some  $\zeta > 0$ . Then for any  $\varepsilon, \varepsilon' > 0$ , there exists an algorithm which given  $\gamma, \varepsilon, \varepsilon'$ , and  $\zeta$ , computes a  $(1 + \varepsilon')$ -embedding  $\hat{f} : X \rightarrow \mathbb{R}^d$ , with accuracy at least  $1 - O(\zeta^{1/2} \log^{3/4} n (\log \log n)^{1/2}) - \varepsilon$ , in time  $n^{O(1)}2^{\varepsilon^{-2}(\frac{d \log n}{\zeta \varepsilon'})^{O(d)}}$ . In particular, for any fixed  $d, \varepsilon, \varepsilon'$ , and  $\zeta$ , the running time is quasi-polynomial.*

**Learning tree metric spaces.** We next consider the case of embedding into tree metric spaces. More precisely, we are given an instance  $(X, \mathcal{S}, \mathcal{D}, u, \ell)$  of the metric learning problem, and we wish to find a tree  $\hat{T}$ , and an embedding  $\hat{f} : X \rightarrow \hat{T}$ , with maximum accuracy. Note that the tree  $\hat{T}$  is not part of the input. Our results closely resemble the ones from the case of learning Euclidean metric spaces. Specifically, for the case of perfect information we obtain a PTAS, summarized in the following.

► **Theorem 5** (Learning tree metric spaces with perfect information). *Let  $u, \ell > 0$  be fixed constants. Let  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the problem of learning a tree metric space with perfect information, with  $|X| = n$ . Suppose that  $\gamma$  admits an 1-embedding  $f^* : X \rightarrow V(T^*)$ , for some tree  $T^*$ , with accuracy 1. Then for any  $\varepsilon, \varepsilon' > 0$ , there exists a randomized algorithm which given  $\gamma, \varepsilon$ , and  $\varepsilon'$ , computes some tree  $\hat{T}$ , and a  $(1 + \varepsilon')$ -embedding  $\hat{f} : X \rightarrow V(\hat{T})$ , with accuracy at least  $1 - \varepsilon$ , in time  $n^{O(1)}g(\varepsilon, \varepsilon') = n^{O(1)}2^{1/(\varepsilon\varepsilon')^{O(1/\varepsilon)}}$ , with high probability. In particular, for any fixed  $\varepsilon$  and  $\varepsilon'$ , the running time is polynomial.*

As in the case of learning Euclidean metric spaces, we also obtain a QPTAS for learning tree metric spaces in the presence of imperfect information, summarized in the following.

► **Theorem 6** (Learning tree metric spaces with imperfect information). *Let  $u, \ell > 0$  be fixed constants. Let  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the problem of learning a tree metric space with imperfect information, with  $|X| = n$ . Suppose that  $\gamma$  admits an 1-embedding  $f^* : X \rightarrow V(T^*)$ , for some tree  $T^*$ , with accuracy  $1 - \zeta$ , for some  $\zeta > 0$ . Then for any  $\varepsilon, \varepsilon' > 0$ , there exists a randomized algorithm which given  $\gamma, \varepsilon, \varepsilon'$ , and  $\zeta$ , computes some tree  $\hat{T}$ , and a  $(1 + \varepsilon')$ -embedding  $\hat{f} : X \rightarrow V(\hat{T})$ , with accuracy at least  $1 - O(\zeta^{1/2} \log^{3/4} n (\log \log n)^{1/2}) - \varepsilon$ , in time  $2^{((\log n)/(\zeta^{1/2} \varepsilon \varepsilon'))^{O(1/\varepsilon)}}$ . In particular, for any fixed  $\varepsilon, \varepsilon'$ , and  $\zeta$ , the running time is quasi-polynomial.*

### 1.3 Overview of our techniques

We now give a brief overview of the techniques used in obtaining our approximation algorithms for the metric learning problem. Interestingly, our algorithms for the Euclidean case closely resemble our algorithms for tree metric spaces.



**Learning Euclidean metric spaces.** At the high level, our algorithms for learning Euclidean metric spaces consist of the following steps:

**Step 1: Partitioning.** We partition the instance into sub-instances, each admitting an embedding into a ball of small diameter. For the case of perfect information, it is easy to show that such a partition exists, using a Lipschitz partition of  $\mathbb{R}^d$  (see [13]). Such a partition can be chosen so that only a small fraction of similarity constraints are “cut” (that is, their endpoints fall in different clusters of the partition). However, computing such a partition is a difficult task, since we do not have an optimal embedding (which is what we seek to compute). We overcome this obstacle by using a result of Krauthgamer and Roughgarden [22], which allows us to compute such a partition, without access to an optimal embedding.

For the case of imperfect information, the situation is somewhat harder since there might be no embedding that satisfies all the constraints. This implies that the partition that we use in the case of perfect information, is not guaranteed to exist anymore. Therefore, instead of using a geometric partitioning procedure, we resort to graph-theoretic methods. We consider the graph  $G_{\mathcal{S}}$ , with  $V(G) = X$ , and with  $E(G) = \mathcal{S}$ . Roughly speaking, we partition  $G_{\mathcal{S}}$  into expanding subgraphs. This can be done by deleting only a relatively small fraction of edges. For each expanding subgraph, we can show that the corresponding sub-instance admits an embedding into a subspace of small diameter.

**Step 2: Embedding each sub-instance.** Once we have partitioned the instance into sub-instances, with each one admitting an embedding into a ball of small diameter, it remains to find such an embedding. This is done by first discretizing the target ball, and then using the theory of pseudoregular partitions (see e.g. [17]) to exhaustively find a good embedding. The discretization step introduces contrastive distortion  $1 + \varepsilon'$ , for some  $\varepsilon' > 0$  that can be made arbitrarily small in expense of the running time.

**Step 3: Combining the embeddings.** Finally, we need to combine the embeddings of the sub-instances to an embedding of the original instance. This can easily be done by ensuring that the images of different clusters are sufficiently far from each other. Since only a small fraction of similarity constraints are cut by the original partition, it follows that the resulting accuracy is high.

**Learning tree metric spaces.** Surprisingly, the above template is also used, almost verbatim, for the case of learning tree metric spaces. The only difference is that the partitioning step now resembles the random partitioning scheme of Klein, Plotkin and Rao [21] for minor-free graphs. The embedding step requires some modification, since the space of trees of bounded diameter is infinite, and thus exhaustive enumeration is not directly possible. We resolve this issue by first showing that if there exists an embedding into a tree of small diameter, then there also exists an embedding into a *single* tree, which we refer to as *canonical*, and with only slightly worse accuracy.

## 1.4 Related work

**Metric embeddings.** The theory of metric embeddings is the source of several successful methods for processing metrical data sets, which have found applications in metric learning (see, e.g. [29, 23]). Despite the common aspects of the use of metric embeddings in metric learning and in algorithm design, there are some key differences. Perhaps the most important difference is that the metric embedding methods being used in algorithm design often correspond to unsupervised metric learning tasks. Consequently, problems and methods that are studied in the context of metric learning, have not received much attention from the

theoretical computer science community. Many of the existing methods used in supervised metric learning tasks are based on convex optimization, and are thus limited to specific kinds of objective functions and constraints (see e.g. [30, 16]).

The problems considered in this paper are closely related to questions studied in the context of computing low-distortion embeddings. For example, the problem of learning a tree metric space is closely related to the problem of embedding into a tree, which has been studied under various classical notions of distortion (see e.g. [12, 9, 28, 2, 1]). Similarly, several algorithms have been proposed for the problem of computing low-distortion embeddings into Euclidean space (see e.g. [24, 3, 26, 27, 5]). Various extensions and generalizations have also been considered for maps that approximately preserve the relative ordering of distances (see e.g. [10, 2, 6]). However, we remark that all of the above results correspond to the unsupervised version of the metric learning problem, and are thus not directly applicable in the supervised setting considered here.

**Other types of supervision and embeddings.** We note that other notions of supervision have also been considered in the metric learning literature. For example, instead of pairs of objects that are labeled either “similar” or “dissimilar”, another possibility is to have triple constraints of the form  $(x, y, z) \in \binom{X}{3}$ , encoding the fact that “ $x$  is more similar to  $y$  than  $z$ ”. In this case, one seeks to find a mapping  $f : X \rightarrow Y$ , such that  $\rho(f(x), f(y)) < \rho(f(x), f(z)) - m$ , for some *margin* parameter  $m > 0$  (see e.g. [30]). However, the form of supervision that we consider is one of the most popular ones in practice. Furthermore, it is often desirable that the mapping  $f$  has a special form, such as linearity (when  $X$  is a subset of some linear space), or being specified implicitly via a set of parameters, as is the case of mappings computed by neural networks. We believe that our work can lead to further theoretical understanding of the metric learning problem under different types of supervision, and for specific classes of embeddings.

**Correlation Clustering.** The supervised metric learning problem considered here can be thought of as a generalization of the classical Correlation Clustering problem [7]. More specifically, the Correlation Clustering problem is precisely the metric learning problem with finite contrastive distortion, for the special case when the host is the uniform metric space, with  $u = 0$  and  $\ell = 1$ .

## 1.5 Organization

The rest of the paper is organized as follows. Section 2 introduces some notation and definitions. Section 3 shows how pseudoregular partitions can be used to compute near-optimal embeddings into spaces of bounded cardinality. Section 4 presents the algorithm for learning Euclidean spaces with perfect information. The algorithm for learning Euclidean spaces with imperfect information is given in Section 5.

The algorithms for learning tree metric spaces under perfect and imperfect information, the exact algorithm for learning line metric spaces with perfect information, and the NP-hardness proof of learning the plane with perfect information can be found in the full version of this paper.

## 2 Preliminaries

For any non-negative integer  $n$ , we use the notation  $[n] = \{1, \dots, n\}$ , with the convention  $[0] = \emptyset$ . Let  $M = (X, \rho)$  be some metric space. We say that  $M$  is a *line* metric space if it can be realized as a submetric of  $\mathbb{R}$ , endowed with the standard distance. For a graph  $G$  and some  $v \in V(G)$ , we write

$$N_G(v) = \{u \in V(G) : \{v, u\} \in E(G)\}.$$

For some  $U \subset V(G)$ , we denote by  $E(U)$  the set of edges in  $G$  that have both endpoints in  $U$ ; that is  $E(U) = \{\{u, v\} \in E(G) : \{u, v\} \subseteq U\}$ . For some  $U, U' \subset V(G)$ , we also write  $E(U, U') = \{\{u, v\} \in E(G) : u \in U, v \in U'\}$ .

## 3 Pseudoregular partitions and spaces of bounded cardinality

In this Section we present an algorithm for computing a nearly-optimal embedding, provided that there exists a solution contained inside a ball of small cardinality.

Let  $G$  be a  $n$ -vertex graph. For any disjoint  $A, B \subseteq V(G)$ , let  $e(A, B) = |E(A, B)|$ , where  $E(A, B)$  denotes the set of edges with one endpoint in  $A$  and one in  $B$ , and let  $\mathbf{d}(A, B) = \frac{e(A, B)}{|A| \cdot |B|}$ . Let  $\mathcal{V} = V_1, \dots, V_k$  be a partition of  $V(G)$ . For any  $i, j \in [k]$ , let  $\mathbf{d}_{i,j} = \mathbf{d}(V_i, V_j)$ . For any  $U \subseteq V(G)$ , let  $U_i = U \cap V_i$ . The partition  $\mathcal{V}$  is called  $\varepsilon$ -*pseudoregular* if for all disjoint  $S, T \subseteq V(G)$ , we have  $\left| e(S, T) - \sum_{i \in [k]} \sum_{j \in [k]} \mathbf{d}_{i,j} |S \cap V_i| |T \cap V_j| \right| \leq \varepsilon n^2$ . It is also called *equitable* if for all  $i, j \in [k]$ , we have  $||V_i| - |V_j|| \leq 1$ . We recall the following result due to Frieze and Kannan [18] on computing pseudoregular partitions (see also [17, 8]).

► **Theorem 7** ([18]). *There exists a randomized algorithm which given an  $n$ -vertex graph  $G$ , and  $\varepsilon, \delta > 0$ , computes an equitable  $\varepsilon$ -pseudoregular partition of  $G$  with at most  $k = 2^{O(\varepsilon^{-2})}$  parts, in time  $2^{O(\varepsilon^{-2})} n^2 / (\varepsilon^2 \delta^3)$ , with probability at least  $1 - \delta$ .*

The following is the main result of this Section.

► **Lemma 8** (Pseudoregular partitions and embeddings into spaces of small cardinality). *Let  $\varepsilon, \varepsilon' > 0$ . Let  $\gamma_C = (C, \mathcal{S}, \mathcal{D}, u, \ell)$  be an instance of the metric learning problem, with  $|C| = n$ . Let  $M = (N, \rho)$  be some metric space. Suppose that  $\gamma_C$  admits a  $(1 + \varepsilon')$ -embedding  $g^* : C \rightarrow N$  with accuracy  $r^*$ . Then, there exists an algorithm which given  $\gamma_C$  and  $M$ , computes a  $(1 + \varepsilon')$ -embedding  $\hat{g} : C \rightarrow N$ , with accuracy at least  $r^* - \varepsilon$ . The running time is  $n^{O(1)} 2^{O(|N|^5 / \varepsilon^2)}$ .*

Due to lack of space, the proof of Lemma 8 is available in the full version of this paper.

## 4 Learning Euclidean metric spaces with perfect information

In this section we describe our algorithm for learning  $d$ -dimensional Euclidean metric spaces with perfect information. First, we present some tools from the theory of random metric partitions, and show how they can be used to partition an instance to smaller ones, each corresponding to a subspace of bounded diameter. The final algorithm combines this decomposition with the algorithm from Section 3 for learning metric spaces of bounded diameter.

### 4.1 Euclidean spaces and Lipschitz partitions

We introduce the necessary tools for randomly partitioning a metric space, and use them to obtain a decomposition of the input into pieces, each admitting an embedding into a ball in  $\mathbb{R}^d$  of small diameter.

Let  $M = (X, \rho)$  be a metric space. A partition  $P$  of  $X$  is called  $\Delta$ -bounded, for some  $\Delta > 0$ , if every cluster in  $P$  has diameter at most  $\Delta$ . Let  $\mathcal{P}$  be a probability distribution over partitions of  $X$ . We say that  $\mathcal{P}$  is  $(\beta, \Delta)$ -Lipschitz, for some  $\beta, \Delta > 0$ , if the following conditions are satisfied:

- Any partition in the support of  $\mathcal{P}$ , is  $\Delta$ -bounded.
- For all  $p, q \in X$ ,  $\Pr_{P \sim \mathcal{P}}[P(x) \neq P(y)] \leq \beta \cdot \rho(x, y)$ .

► **Lemma 9** ([13]). *For any  $d \geq 1$ , and any  $\Delta > 0$ , we have that  $d$ -dimensional Euclidean space admits a  $(O(\sqrt{d}/\Delta), \Delta)$ -Lipschitz distribution.*

Let  $G_S$  be the graph with vertex set  $V(G_S) = X$ , and edge set  $E(G_S) = \mathcal{S}$ . We use  $\rho_S$  to denote the shortest-path distance of  $G_S$ , where the length of each edge is set to  $u$ .

► **Lemma 10.** *Let  $X' \subset X$ , such that the subgraph of  $G_S$  induced on  $X'$  (i.e.  $G_S[X']$ ) is connected. Suppose further that  $\text{diam}(f^*(X')) \leq \Delta$ , for some  $\Delta > 0$ . Then, for any  $p, q \in X'$ , we have  $\rho_S(p, q) \leq 2u(1 + 4\Delta/u)^d$ .*

**Proof.** Let  $G' = G_S[X']$ . Let  $Q = x_1, \dots, x_L$  be the shortest path between  $p$  and  $q$  in  $G'$ , with  $x_1 = p$ ,  $x_L = q$ . We first claim that for all  $i, j \in \{1, \dots, L\}$ , with  $i < j - 1$ , we have that

$$\|f^*(x_i), f^*(x_j)\|_2 > u. \quad (3)$$

Indeed, note that if  $\|f^*(x_i), f^*(x_j)\|_2 \leq u$ , then since  $\gamma$  has full information, i.e.  $\mathcal{S} \cup \mathcal{D} = \binom{X}{2}$ , it must be that  $\{x_i, x_j\} \in \mathcal{S}$ , and thus the edge  $\{x_i, x_j\}$  is present in the graph  $G'$ . This implies that  $\rho_{G'}(x_i, x_j) = u$ , which contradicts the fact that  $Q$  is a shortest path. We have thus established (3).

For each  $i \in \{1, \dots, L\}$ , let  $B_i$  be the ball in  $\mathbb{R}^d$  centered at  $f^*(x_i)$  and of radius  $u/2$ . By (3) we get that the balls  $B_2, B_4, \dots, B_{2\lfloor L/2 \rfloor}$  are pairwise disjoint. Since  $\text{diam}(f^*(X')) \leq \Delta$ , it follows that there exists some  $x^* \in \mathbb{R}^d$ , such that  $f^*(X') \subset \text{ball}(x^*, 2\Delta)$ . Therefore

$$\bigcup_{i=1}^{\lfloor L/2 \rfloor} B_{2i} \subseteq \bigcup_{i=1}^{\lfloor L/2 \rfloor} \text{ball}(x_{2i}, u/2) \subseteq \text{ball}(x^*, 2\Delta + u/2). \quad (4)$$

Recall that the volume of the ball of radius  $r$  in  $\mathbb{R}^d$  is equal to  $V_d(r) = \frac{\pi^{d/2}}{\Gamma(1+d/2)} r^d$ . From (4) we get  $\lfloor L/2 \rfloor \cdot V_d(u/2) \leq V_d(2\Delta + u/2)$ , and thus  $\rho_S(p, q) \leq \rho_{G'}(p, q) = u \cdot (L - 1) \leq 2u \frac{V_d(2\Delta + u/2)}{V_d(u/2)} = 2u(1 + 4\Delta/u)^d$ , which concludes the proof. ◀

We now show that the shortest-path metric of the similarity constraint graph admits a Lipschitz distribution.

► **Lemma 11.** *For any  $\Delta > 0$ , the metric space  $(X, \rho_S)$  admits a  $(O(\sqrt{d}/\Delta), u(1 + 4\Delta/u)^d)$ -Lipschitz distribution.*

**Proof.** Let  $f^* : X \rightarrow \mathbb{R}^d$  be some mapping with accuracy 1. By Lemma 9 there exist some  $(O(\sqrt{d}/\Delta), \Delta)$ -Lipschitz distribution,  $\mathcal{P}$ , of  $(\mathbb{R}^d, \|\cdot\|_2)$ . Let  $P$  be a random partition sampled from  $\mathcal{P}$ . We define a partition  $P'$  of  $X$  as follows. Let  $G'_S$  be the graph obtained from  $G_S$  by deleting all edges whose endpoints under  $f^*$  are in different clusters of  $P$ . That is,

$V(G'_S) = X$ , and  $E(G'_S) = \{\{p, q\} \in \mathcal{S} : P(f^*(p)) = P(f^*(q))\}$ , where  $P(p)$  denotes the cluster of  $P$  containing  $p$ . We define  $P'$  to be the set of connected components of  $G'_S$ . Let  $\mathcal{P}'$  be the induced distribution over partitions of  $(X, \rho_S)$ . It remains to show that  $\mathcal{P}'$  is  $(O(\sqrt{d}/\Delta), u(1 + 4\Delta/u)^d)$ -Lipschitz.

Let us first bound the probability of separation. Let  $p, q \in X$ . Let  $x_1, \dots, x_t$ , with  $x_1 = p, x_t = q$ , be a shortest-path in  $G_S$  between  $p$  and  $q$ . Since  $\mathcal{P}$  is  $(O(\sqrt{d}/\Delta), \Delta)$ -Lipschitz, it follows that  $\Pr[f^*(p) \neq f^*(q)] \leq \frac{O(\sqrt{d})}{\Delta} \|f^*(p) - f^*(q)\|_2 \leq \frac{O(\sqrt{d})}{\Delta} \sum_{i=1}^{t-1} \|f^*(x_i) - f^*(x_{i+1})\|_2 \leq \frac{O(\sqrt{d})}{\Delta} \sum_{i=1}^{t-1} \rho_S(x_i, x_{i+1}) = \frac{O(\sqrt{d})}{\Delta} \rho_S(p, q)$ .

Finally, let us bound the diameter of the clusters in  $P'$ . Let  $C$  be a cluster in  $P'$ . By construction,  $\text{diam}(f^*(C)) \leq \Delta$ , and  $G_S[C]$  is a connected subgraph, thus by Lemma 10 we obtain that the diameter of  $C$  in the metric space  $(X, \rho)$  is at most  $u(1 + 4\Delta/u)^d$ . Thus  $\mathcal{P}'$  is  $(O(\sqrt{d}/\Delta), u(1 + 4\Delta/u)^d)$ -Lipschitz, concluding the proof. ◀

The following result allows us to sample from a Lipschitz distribution, without additional information about the geometry of the underlying space.

► **Lemma 12** ([22]). *Let  $M = (X, \rho)$  be a metric space, and let  $\Delta > 0$ . Suppose that  $M$  admits a  $(\beta, \Delta)$ -Lipschitz distribution, for some  $\beta > 0$ . Then there exists a randomized polynomial-time algorithm, which given  $M$  and  $\Delta$ , outputs a random partition  $P$  of  $X$ , such that  $P$  is sampled from a  $(2\beta, \Delta)$ -Lipschitz distribution  $\mathcal{P}$ .*

## 4.2 The algorithm: Combining Lipschitz and pseudoregular partitions

We now present our final algorithm for learning Euclidean metric spaces with perfect information. First, we show that, using the algorithm from Section 3, we can learn  $d$ -dimensional Euclidean metric spaces of bounded diameter. The final algorithm combines this result with an efficient procedure for decomposing the instance using a random Lipschitz partition.

► **Lemma 13** (Computing Euclidean embeddings of small diameter). *Let  $\varepsilon, \varepsilon' > 0$ . Let  $\gamma_C = (C, \mathcal{S}, \mathcal{D}, u, \ell)$ , with  $|C| = n$ , be an instance of the metric learning problem. Suppose that  $\gamma_C$  admits an embedding  $g^* : C \rightarrow \mathbb{R}^d$  with accuracy  $r^*$ , such that  $\text{diam}(g^*(C)) \leq \Delta$ , for some  $\Delta > 0$ . Then, there exists an algorithm which given  $\gamma_C$  and  $\Delta$ , computes a  $(1 + \varepsilon')$ -embedding  $\hat{g} : C \rightarrow \mathbb{R}^d$ , with accuracy at least  $r^* - \varepsilon$ . Furthermore for any fixed  $u$  and  $\ell$ , the running time is  $T(n, d, \Delta, \varepsilon, \varepsilon') = 2^{\varepsilon^{-2}(\Delta\sqrt{d}/\varepsilon')^{O(d)}}$ .*

**Proof.** Let  $c_1 = \frac{\varepsilon'}{2\sqrt{d}} \min\{\ell, u\}$ . Since  $\text{diam}(g^*(C)) \leq \Delta$ , it follows that there exists some ball  $B \subset \mathbb{R}^d$  of radius  $2\Delta$  such that  $g^*(C) \subset B$ . Let  $c_1\mathbb{Z}^d$  denote the integer lattice scaled by a factor of  $c_1$ . Let  $N = B \cap (c_1\mathbb{Z}^d)$ . Note that  $|N| \leq (2\Delta/c_1)^d$ . Let  $g' : C \rightarrow \mathbb{R}^d$  be defined such that for each  $p \in C$ , we have that  $g'(p)$  is a nearest neighbor of  $g^*(p)$  in  $N$ , breaking ties arbitrarily. For any  $\{p, q\} \in \mathcal{S} \cap \binom{C}{2}$ , we have  $\|g'(p) - g'(q)\|_2 \leq \|g'(p) - g^*(p)\|_2 + \|g^*(p) - g^*(q)\|_2 + \|g^*(q) - g'(q)\|_2 \leq u + \sqrt{d}c_1 \leq u(1 + \varepsilon')$ . Similarly, for any  $\{p, q\} \in \mathcal{D} \cap \binom{C}{2}$ , we have  $\|g'(p) - g'(q)\|_2 \geq -\|g'(p) - g^*(p)\|_2 + \|g^*(p) - g^*(q)\|_2 - \|g^*(q) - g'(q)\|_2 \geq \ell - \sqrt{d}c_1 \leq \ell/(1 + \varepsilon')$ , and thus  $g'$  is a  $(1 + \varepsilon')$ -embedding, with accuracy  $r^*$ .

Let  $M = (N, \|\cdot\|_2)$ ; that is, the metric space on  $N$  endowed with the Euclidean distance. By Lemma 8 we can therefore compute a  $(1 + \varepsilon')$ -embedding  $\hat{g} : C \rightarrow N$ , with accuracy  $r^* - \varepsilon$ , in time  $n^{O(1)} 2^{O(|N|^5/\varepsilon^2)} = 2^{\varepsilon^{-2}(\Delta\sqrt{d}/\varepsilon')^{O(d)}}$ . ◀

We are now ready to prove the main result in this Section.

**Proof of Theorem 3.** We first construct the graph  $G_S$  as above. Let  $\Delta = c\sqrt{d}u/\varepsilon$ , for some universal constant  $c > 0$ . By Lemma 11 the metric space  $(X, \rho_S)$  admits some  $(O(\sqrt{d}/\Delta), \Delta')$ -Lipschitz distribution, for some  $\Delta' = u(1 + 4\Delta/u)^d$ . By Lemma 12 we can compute, in polynomial time, some random partition  $P$  of  $X$ , such that  $P$  is distributed according to some  $(O(\sqrt{d}/\Delta), \Delta')$ -Lipschitz distribution  $\mathcal{P}$ .

Let  $\mathcal{S}' = \{\{p, q\} \in \mathcal{S} : P(p) \neq P(q)\}$ . Since  $\mathcal{P}$  is  $(O(\sqrt{d}/\Delta), \Delta')$ -Lipschitz, it follows by the linearity of expectation that

$$\mathbb{E}[|\mathcal{S}'|] = \sum_{\{p, q\} \in \mathcal{S}} \Pr[P(p) \neq P(q)] \leq \sum_{\{p, q\} \in \mathcal{S}} O(\sqrt{d}) \frac{\rho_S(p, q)}{\Delta} = O(\sqrt{d}) \frac{u}{\Delta} |\mathcal{S}| \leq \varepsilon |\mathcal{S}| / 4, \quad (5)$$

where the last inequality holds for some large enough constant  $c > 0$ .

Fix some optimal embedding  $f^* : X \rightarrow \mathbb{R}^d$ , that satisfies all the constraints in  $\gamma$ . Let  $C$  be a cluster in  $P$ . Since  $P$  is  $\Delta'$ -bounded, it follows that the diameter of  $C$  in  $(X, \rho_S)$  is at most  $\Delta'$ . Thus, for any  $p, q \in C$ , there exists in  $G_S$  some path  $Q = x_0, \dots, x_L$  between  $p$  and  $q$ , with  $L \leq \Delta'/u$  edges. Thus  $\|f^*(p) - f^*(q)\|_2 \leq \sum_{i=0}^{L-1} \|f^*(x_i) - f^*(x_{i+1})\|_2 \leq L \cdot u \leq \Delta'$ , which implies that  $\text{diam}(f^*(C)) \leq \Delta'$ . Furthermore  $f^*$  has accuracy 1.

Let  $\mathcal{S}_C = \mathcal{S} \cap \binom{C}{2}$ ,  $\mathcal{D}_C = \mathcal{D} \cap \binom{C}{2}$ . Then  $\gamma_C = (C, \mathcal{S}_C, \mathcal{D}_C, u, u)$  is an instance of non-linear metric learning in  $d$ -dimensional Euclidean space that admits a solution  $f^* : C \rightarrow \mathbb{R}^d$  with accuracy 1, with  $\text{diam}(f^*(C)) \leq \Delta'$ . Therefore, for each  $C \in P$ , using Lemma 13 we can compute some  $(1 + \varepsilon')$ -embedding  $f_C : C \rightarrow \mathbb{R}^d$ , with accuracy at least  $1 - \varepsilon/2$ , in time  $T(n, d, \Delta', \varepsilon/2, \varepsilon') = n^{O(1)} 2^{(\frac{d}{\varepsilon'})^{O(d^2)}}$ . We can now combine all these maps  $f_C$  into a single map  $\hat{f} : X \rightarrow \mathbb{R}^d$ , by translating each image  $f_C(C)$  such that for any distinct  $C, C' \in P$ , for any  $p \in C, p' \in C'$ , we have  $\|f_C(p) - f_{C'}(p')\|_2 \geq u$ . For any  $p \in C$ , we set  $f(p) = f_C(p)$ , where  $C$  is the unique cluster in  $P$  containing  $p$ . It is immediate that any constraint  $\{p, q\} \in \mathcal{S}$ , with  $p$  and  $q$  in different clusters is violated by  $f$ . The total number of such violations is  $|\mathcal{S}'|$ . By Markov's inequality and (5), these violations are at most  $\varepsilon |\mathcal{S}| / 2$ , with probability at least  $1/2$ . Similarly, any  $\{p, q\} \in \mathcal{D}$ , with  $p$  and  $q$  in different clusters is satisfied by  $f$ . All remaining violations are on constraints with both endpoints in the same cluster of  $P$ . Since the accuracy of each  $f_C$  is at least  $1 - \varepsilon$ , it follows that the total number of these violations is at most  $\varepsilon(|\mathcal{S}| + |\mathcal{D}|) / 2$ . Therefore, the total number of violations among all constraints is at most  $\varepsilon(|\mathcal{S}| + |\mathcal{D}|) / 2 + \varepsilon |\mathcal{S}| / 2 \leq \varepsilon(|\mathcal{S}| + |\mathcal{D}|)$ , with probability at least  $1/2$ . In other words, the accuracy of  $f$  is at least  $1 - \varepsilon$ , with probability at least  $1/2$ . The success probability can be increased to  $1 - 1/n^c$ , for any constant  $c > 0$ , by repeating the algorithm  $O(\log n)$  times and returning the best embedding found. Finally, the running time is dominated by the computation of the maps  $f_C$ , for all clusters  $C$  in  $P$ . Since there are at most  $n$  such clusters, the total running time is  $n^{O(1)} 2^{(\frac{d}{\varepsilon'})^{O(d^2)}}$ , concluding the proof.  $\blacktriangleleft$

## 5 Learning Euclidean metric spaces with imperfect information

In this Section we describe our algorithm for learning  $d$ -dimensional Euclidean metric spaces with imperfect information. We first obtain some preliminary results on graph partitioning via sparse cuts. We next show that for any instance whose similarity constraints induce an expander graph, the optimal solution must be contained inside a ball of small diameter. Our final algorithm combines these results with the algorithm from Section 3 for embedding into host metric spaces of bounded cardinality.

Fix some instance  $\gamma = (X, \mathcal{S}, \mathcal{D}, u, \ell)$ . For the remainder of this Section we define the graphs  $G_S = (X, E_S)$ ,  $G_D = (X, E_D)$ , and  $G_{S \cup D} = (X, E_{S \cup D})$ , where  $E_S = \mathcal{S}$ ,  $E_D = \mathcal{D}$ , and  $E_{S \cup D} = \mathcal{S} \cup \mathcal{D}$ .

### 5.1 Well-linked decompositions

We recall some results on partitioning graphs into well-connected components. An instance to the Sparsest-Cut problem consists of a graph  $G$ , with each edge  $\{x, y\} \in E(G)$  having capacity  $\text{cap}(x, y) \geq 0$ , and each pair  $x, y \in V(G)$  having demand  $\text{dem}(x, y) \geq 0$ . The goal is to find a cut  $(U, \bar{U})$  of minimum sparsity, denoted by  $\phi(U)$ , which is defined as  $\phi(U) = \frac{\text{cap}(U, \bar{U})}{\text{dem}(U, \bar{U})}$ , where  $\text{cap}(U, \bar{U}) = \sum_{\{x, y\} \in E(U, \bar{U})} \text{cap}(x, y)$ ,  $\text{dem}(U, \bar{U}) = \sum_{\{x, y\} \in E(U, \bar{U})} \text{dem}(x, y)$ , and  $E(U, \bar{U})$  denotes the set of edges between  $U$  and  $\bar{U}$ . We recall the following result of Arora, Lee and Naor [3] on approximating the Sparsest-Cut problem (see also [4]).

► **Theorem 14 ([3]).** *There exists a polynomial-time  $O(\sqrt{\log n} \log \log n)$ -approximation for the Sparsest-Cut problem on  $n$ -vertex graphs.*

► **Lemma 15.** *Let  $\alpha > 0$ . There exists a polynomial-time algorithm which computes some  $E' \subset \mathcal{S} \cup \mathcal{D}$ , with  $|E'| \leq \alpha |\mathcal{S} \cup \mathcal{D}|$ , such that for every connected component  $C$  of  $G_{\mathcal{S}} \setminus E'$ , and any  $U \subset C$ , we have  $|E_{\mathcal{S}}(U, C \setminus U)| = \Omega\left(\frac{\alpha}{\log^{3/2} n \log \log n}\right) |U| \cdot |C \setminus U|$ .*

**Proof.** We compute  $E'$  inductively starting with  $E' = \emptyset$ . We construct an instance of the Sparsest-Cut problem on graph  $G_{\mathcal{S}}$ , where for any  $\{x, y\} \in \mathcal{S}$  we have  $\text{cap}(x, y) = 1$ , and for any  $x, y \in X$ , we have  $\text{dem}(x, y) = 1$ . Let  $\chi = \frac{\alpha}{c \log^{3/2} n \log \log n}$ , for some universal constant  $c > 0$  to be specified. If there exists a cut of sparsity at most  $\chi$ , then by Theorem 14 we can compute a cut  $(U, \bar{U})$  of sparsity  $O(\chi \sqrt{\log n} \log \log n)$ . We add to  $E'$  all the edges in  $E_{\mathcal{S}}(U, \bar{U})$ , and we recurse on the subgraphs  $G_{\mathcal{S}}[U]$  and  $G_{\mathcal{S}}[\bar{U}]$ . If no cut with the desired sparsity exists, then we terminate the recursion. For each cut  $(U, \bar{U})$  found, the edges in  $E_{\mathcal{S}}(U, \bar{U})$  that were added to  $E'$  are charged to the edges in  $E_{\mathcal{S} \cup \mathcal{D}}(U, \bar{U})$ . In total, we charge only  $O(\chi \sqrt{\log n} \log \log n)$ -fraction of all edges, and thus  $|E'| = O(\chi n \log^{3/2} n \log \log n) \leq \alpha |\mathcal{S} \cup \mathcal{D}|$ , where the last inequality follows for some sufficiently large constant  $c > 0$ , concluding the proof. ◀

### 5.2 Isoperimetry and the diameter of embeddings

Here we show that if the graph  $G_{\mathcal{S}}$  of similarity constraints is “expanding” relative to  $G_{\mathcal{S} \cup \mathcal{D}}$ , then there exists an embedding with near-optimal accuracy, that has an image of small diameter. Technically, we require that in any cut in  $G_{\mathcal{S} \cup \mathcal{D}}$ , a significant fraction of its edges are in  $\mathcal{S}$ . Intuitively, this condition forces almost all points to be mapped within a small ball. The next Lemma formalizes this statement.

► **Lemma 16.** *Let  $F \subset \mathcal{S}$ , with  $|F| \leq \zeta \binom{|X|}{2}$ , for some  $\zeta > 0$ . Suppose that for all  $U \subset X$ , we have*

$$|E_{\mathcal{S}}(U, X \setminus U)| \geq \alpha' \cdot |U| \cdot |X \setminus U|, \tag{6}$$

for some  $\alpha' > 0$ . Then there exists  $J \subseteq X$ , satisfying the following conditions:

- (1) No edge in  $F$  has both endpoints in  $J$ ; that is,  $F \cap E_{\mathcal{S}}(J) = \emptyset$ .
- (2)  $|E_{\mathcal{S} \cup \mathcal{D}}(J)| \geq (1 - \zeta/\alpha') \binom{|X|}{2}$ .
- (3) For all  $U \subset J$ , we have

$$|E_{\mathcal{S}}(U, J \setminus U)| = \Omega(\alpha') |U| \cdot |J \setminus U|. \tag{7}$$

**Proof.** Let  $C$  be the largest connected component of  $G_{\mathcal{S}} \setminus F$ , and let  $C' = X \setminus C$ . By (6) we get

$$|E_{\mathcal{S} \cup \mathcal{D}}(C, C')| = |C| \cdot |C'| = O(1/\alpha') |E_{\mathcal{S}}(C, C')| = O(1/\alpha') |F| = O(\zeta/\alpha') |X|^2. \tag{8}$$



## 45:12 Algorithms for Metric Learning via Contrastive Embeddings

Since  $G_{\mathcal{S} \cup \mathcal{D}}$  is a complete graph, it follows from (8) that the number of edges not in  $G_{\mathcal{S} \cup \mathcal{D}}[C]$  is at most  $O(\zeta/\alpha')|X|^2$ , and therefore

$$|E_{\mathcal{S} \cup \mathcal{D}}(C)| \geq (1 - O(\zeta/\alpha')) \binom{|X|}{2}. \quad (9)$$

Next, we compute some  $J \subseteq C$  such that for all  $U \subset C$ ,

$$E_{\mathcal{S}}(U, J \setminus U) \geq c' \alpha' \cdot |U| \cdot |J \setminus U|, \quad (10)$$

for some universal constant  $c'$  to be determined. This is done as follows. If there is no  $U \subset C$  violating (10), then we set  $J = C$ . Otherwise, pick some  $U \subset C$ , such that

$$E_{\mathcal{S}}(U, C \setminus U) < c' \alpha' \cdot |U| \cdot |C \setminus U|. \quad (11)$$

Assume w.l.o.g. that  $|U| \leq |C \setminus U|$ . We delete  $U$  and we recurse on  $C \setminus U$ . All deleted edges are charged to the edges in  $F$  that are incident to  $U$ . It follows by (11) and (6) that, for some sufficiently small constant  $c' > 0$ , at least a  $2/3$ -fraction of the edges incident to  $U$  are in  $F$ . Thus  $|F \cap E_{\mathcal{S}}(U, X \setminus U)| = \Omega(\alpha')|U| \cdot |X \setminus U|$ . We charge the deleted edges (that is, the edges inside the deleted component  $U$  and in  $E_{\mathcal{S}}(U, X \setminus U)$ ) to the edges in  $F \cap E_{\mathcal{S}}(U, X \setminus U)$ , with each edge thus receiving  $O(1/\alpha')$  units of charge. When we recurse on  $C \setminus U$ , the part of  $F$  that was incident to  $U$  is replaced by  $E_{\mathcal{S}}(U, C \setminus U)$ , and thus it decreases in size by at least a factor of 2. Therefore, the total charge that we pay throughout the construction is at most  $O(1/\alpha')|F| = O(\zeta/\alpha')|X|^2$ , which is an upper bound on the number of all deleted edges. This completes the construction of  $J$ . Combining with (9) we get  $|E_{\mathcal{S} \cup \mathcal{D}}(J)| \geq |E_{\mathcal{S} \cup \mathcal{D}}(C)| - O(\zeta/\alpha')|X|^2 \geq (1 - O(\zeta/\alpha')) \binom{|X|}{2}$ , which concludes the proof. ◀

► **Lemma 17.** *Let  $M = (Y, \rho)$  be any metric space. Suppose that  $\gamma$  admits an embedding  $f^* : X \rightarrow Y$  with accuracy  $1 - \zeta$ , for some  $\zeta > 0$ . Suppose further that for all  $U \subset X$ , we have*

$$E_{\mathcal{S}}(U, X \setminus U) \geq \alpha' \cdot |U| \cdot |X \setminus U|, \quad (12)$$

for some  $\alpha' > 0$ . Then there exists an embedding  $f' : X \rightarrow Y$ , with accuracy at least  $1 - O(\zeta/\alpha')$ , such that  $\text{diam}_M(f'(X)) = O\left(\frac{u \log n}{\alpha'}\right)$ .

**Proof.** Let  $J \subseteq X$  be given by Lemma 16, where we set  $F$  to be the set of constraints in  $\mathcal{S}$  that are violated by  $f^*$ . We define  $f' : X \rightarrow Y$  by mapping each  $x \in J$  to  $f^*(x)$ , and each  $x \in X \setminus J$  to some arbitrary point in  $f^*(J)$ . All constraints in  $E_{\mathcal{S}}(J)$  are satisfied by  $f^*$ , and thus also by  $f'$ . It follows that  $f'$  can only violate constraints that are violated by  $f^*$ , and constraints that are not in  $E_{\mathcal{S} \cup \mathcal{D}}(J)$ . Therefore the accuracy of  $f'$  is at least  $1 - \zeta - O(\zeta/\alpha') = 1 - O(\zeta/\alpha')$ .

By Lemma 16 we have that for all  $U \subset J$ ,  $|E_{\mathcal{S}}(U, J \setminus U)| = \Omega(\alpha')|U| \cdot |J \setminus U|$ . Therefore, the combinatorial diameter (that is, the maximum number of edges in any shortest path) of  $G_{\mathcal{S}}[J]$  is at most  $O((\log n)/\alpha')$ . For all  $\{x, y\} \in E_{\mathcal{S}}(J)$  we have  $\rho(f'(x), f'(y)) = \rho(f^*(x), f^*(y)) \leq u$ , it follows that  $\text{diam}_M(f'(X)) = \text{diam}_M(f'(J)) = O\left(\frac{u \log n}{\alpha'}\right)$ . ◀

### 5.3 The algorithm

We are now ready to prove the main result of this Section.

**Proof of Theorem 4.** Fix some optimal embedding  $f^* : X \rightarrow \mathbb{R}^d$ , with accuracy  $1 - \zeta$ . Let  $E' \subset \mathcal{S} \cup \mathcal{D}$  be the set of constraints computed by the algorithm in Lemma 15. We have  $|E'| \leq \alpha |\mathcal{S} \cup \mathcal{D}|$ , for some  $\alpha > 0$  to be determined.

For each connected component  $C$  of  $G_{\mathcal{S}} \setminus E'$ , let  $\gamma_C = (C, \mathcal{S}_C, \mathcal{D}_C, u, \ell)$  be the restriction of  $\gamma$  on  $C$ ; that is,  $\mathcal{S}_C = \mathcal{S} \cap \binom{C}{2}$ , and  $\mathcal{D}_C = \mathcal{D} \cap \binom{C}{2}$ . Let  $f_C^*$  be the restriction of  $f^*$  on  $C$ , and let the accuracy of  $f_C^*$  be  $1 - \zeta_C$ , for some  $\zeta_C \in [0, 1]$ . Let  $F_C$  be the set of constraints in  $E_{\mathcal{S}}(C)$  that are violated by  $f_C^*$ . By Lemma 17 it follows that there exists an embedding  $f'_C : X \rightarrow \mathbb{R}^d$ , with accuracy  $1 - O(\zeta_C/\alpha')$ , and such that  $\text{diam}(f'_C(C)) = O(\frac{u \log n}{\alpha'})$ , for some  $\alpha' = \Omega(\frac{\alpha}{\log^{3/2} n \log \log n})$ .

Using Lemma 13 we can compute a  $(1 + \varepsilon')$ -embedding  $\widehat{f}_C : C \rightarrow \mathbb{R}^d$  with accuracy at least  $1 - O(\zeta_C/\alpha) - \varepsilon$ , in time  $n^{O(1)} 2^{\varepsilon^{-2} (\frac{d \sqrt{d}}{\varepsilon'})^{O(d)}}$ . We can combine all the embeddings  $\widehat{f}_C$  into a single embedding  $\widehat{f} : X \rightarrow \mathbb{R}^d$  by translating the images of  $\widehat{f}_C(C)$  and  $\widehat{f}_{C'}(C')$  in  $\mathbb{R}^d$  to that their distance is at least  $\ell$ , for all distinct components  $C, C'$ . It is immediate that the resulting embedding  $\widehat{f}$  can only violate the constraints in  $E'$ , and the constraints violated by all the  $\widehat{f}_C$ . Thus, the accuracy of  $\widehat{f}$  is at least  $1 - \alpha - O(\zeta/\alpha') - \varepsilon$ . Setting  $\alpha = \zeta^{1/2} \log^{3/4} n (\log \log n)^{1/2}$ , we get that the accuracy of  $\widehat{f}$  is at least  $1 - O(\zeta^{1/2} \log^{3/4} n (\log \log n)^{1/2}) - \varepsilon$ . The running time is dominated by the at most  $n$  executions of the algorithm from Lemma 13, and thus it is at most  $n^{O(1)} 2^{\varepsilon^{-2} (\frac{d \log n}{\zeta \varepsilon'})^{O(d)}}$ , for any fixed  $u > 0$ , concluding the proof.  $\blacktriangleleft$

---

#### References

- 1 Nir Ailon and Moses Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 73–82. IEEE, 2005.
- 2 Noga Alon, Mihai Bădoiu, Erik D Demaine, Martin Farach-Colton, MohammadTaghi Hajiaghayi, and Anastasios Sidiropoulos. Ordinal embeddings of minimum relaxation: general properties, trees, and ultrametrics. *ACM Transactions on Algorithms (TALG)*, 4(4):46, 2008.
- 3 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.
- 4 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.
- 5 Mihai Badoiu. Approximation algorithm for embedding metrics into a two-dimensional space. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2003.
- 6 Mihai Bădoiu, Erik D Demaine, MohammadTaghi Hajiaghayi, Anastasios Sidiropoulos, and Morteza Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 21–34. Springer, 2008.
- 7 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.
- 8 Nikhil Bansal and Ryan Williams. Regularity lemmas and combinatorial algorithms. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 745–754. IEEE, 2009.
- 9 Yair Bartal. Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193. IEEE Computer Society, 1996. doi: 10.1109/SFCS.1996.548477.

- 10 Yonatan Bilu and Nati Linial. Monotone maps, sphericity and bounded second eigenvalue. *Journal of Combinatorial Theory, Series B*, 95(2):283–299, 2005.
- 11 Heinz Breu and David G Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry*, 9(1-2):3–24, 1998.
- 12 Mihai Bădoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 512–521. Society for Industrial and Applied Mathematics, 2007.
- 13 Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 379–388. IEEE, 1998.
- 14 Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.
- 15 Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.
- 16 Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.
- 17 Alan Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 12–20. IEEE, 1996.
- 18 Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.
- 19 Sarel Har-Peled. *Geometric approximation algorithms*, volume 173. American mathematical society Boston, 2011.
- 20 Piotr Indyk, Jiří Matoušek, and Anastasios Sidiropoulos. Low-Distortion Embeddings of Finite Metric Spaces. In Jacob E. Goodman, Joseph O’Rourke, and Csaba D. Toth, editors, *Handbook of Discrete and Computational Geometry, Second Edition*. Chapman and Hall/CRC, 2017.
- 21 Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 682–690. ACM, 1993.
- 22 Robert Krauthgamer and Tim Roughgarden. Metric Clustering via Consistent Labeling. *Theory OF Computing*, 7:49–74, 2011.
- 23 Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.
- 24 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.
- 25 Jiří Matoušek. *Lectures on discrete geometry*, volume 212. Springer Science & Business Media, 2002.
- 26 Jiří Matoušek and Anastasios Sidiropoulos. Inapproximability for metric embeddings into  $\mathbb{R}^d$ . *Transactions of the American Mathematical Society*, 362(12):6341–6365, 2010.
- 27 Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 729–747. IEEE, 2015.
- 28 Amir Nayyeri and Benjamin Raichel. A Treehouse with Custom Windows: Minimum Distortion Embeddings into Bounded Treewidth Graphs. 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 724–736. SIAM, 2017. doi:10.1137/1.9781611974782.46.
- 29 Gregory Shakhnarovich. *Learning task-specific similarity*. PhD thesis, Massachusetts Institute of Technology, 2005.
- 30 Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

# Exact Computation of the Matching Distance on 2-Parameter Persistence Modules

Michael Kerber 

Graz University of Technology, Graz, Austria  
kerber@tugraz.at

Michael Lesnick 

University at Albany, SUNY, United States  
mlesnick@albany.edu

Steve Oudot 

Inria Saclay – Île-de-France, Palaiseau, France  
steve.oudot@inria.fr

---

## Abstract

The matching distance is a pseudometric on multi-parameter persistence modules, defined in terms of the weighted bottleneck distance on the restriction of the modules to affine lines. It is known that this distance is stable in a reasonable sense, and can be efficiently approximated, which makes it a promising tool for practical applications. In this work, we show that in the 2-parameter setting, the matching distance can be computed exactly in polynomial time. Our approach subdivides the space of affine lines into regions, via a line arrangement. In each region, the matching distance restricts to a simple analytic function, whose maximum is easily computed. As a byproduct, our analysis establishes that the matching distance is a rational number, if the bigrades of the input modules are rational.

**2012 ACM Subject Classification** Mathematics of computing → Algebraic topology; Mathematics of computing → Mathematical optimization

**Keywords and phrases** Topological Data Analysis, Multi-Parameter Persistence, Line arrangements

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.46

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1812.09085>.

**Funding** *Michael Kerber*: Supported by Austrian Science Fund (FWF) grant number P 29984-N35.

**Acknowledgements** This work was initiated at the BIRS workshop “Multiparameter Persistent Homology” (18w55140) in Oaxaca, Mexico (Aug. 2018). We thank Jan Reininghaus and the other members of the discussion group on this topic for fruitful initial exchanges. We thank Matthew Wright for helpful discussions about line arrangements, slices, and the computational aspects of 2-parameter persistence.

## 1 Introduction

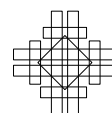
Multi-parameter persistent homology is receiving growing attention, both from the theoretical and computational points of view. Its motivation lies in the possibility of extending the success of topological data analysis to settings where the structure of data is best captured by 2-parameter rather than 1-parameter constructions. The basic algebraic objects of study in multi-parameter persistence are certain commutative diagrams of vector spaces called *persistence modules*. In the 1-parameter setting, persistence modules decompose in an essentially unique way into simple summands called *interval modules*. The decomposition is specified by a discrete invariant called a *persistence diagram*. In contrast, the algebraic



© Michael Kerber, Michael Lesnick, and Steve Oudot;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 46; pp. 46:1–46:15  
Leibniz International Proceedings in Informatics



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



structure of a 2-parameter persistence module (henceforth, *bipersistence module*) can be far more complex. As a result, a good definition of persistence diagram is unavailable for bipersistence modules [5].

Nevertheless, it is still possible to define meaningful notions of distance between multi-parameter persistence modules. Distances on 1-parameter persistence modules play an essential role in both theory and applications. To extend such theory and applications to the multi-parameter setting, one needs to select a suitable distance on multi-parameter persistence modules. However, progress on finding well-behaved, efficiently computable distances on multi-parameter persistence modules has been slow, and this is been an impediment to progress in practical applications.

The most widely studied and applied distances in the 1-parameter setting are the *bottleneck distance* and the *Wasserstein distance* [11, 21]. Both can be efficiently computed via publicly available code [15]. In the multi-parameter setting, the distance that has received the most attention is the *interleaving distance*. On 1-parameter persistence modules, the interleaving and bottleneck distances are equal [17]. The interleaving distance is theoretically well-behaved; in particular, on modules over prime fields, it is the most discriminative distance among the ones satisfying a certain stability condition [17]. However, it was proven recently to be NP-hard to compute or even approximate to any constant factor less than three on bipersistence modules, even under the restriction that the modules decompose as direct sums of *interval modules* [4]. While the problem lies in  $P$  when further restricting the focus to the sub-class of interval modules themselves [10], in practice one often encounters modules that are not interval modules (nor even direct sums thereof).

This motivates the search for a more computable surrogate for the interleaving distance. One natural candidate is the *matching distance*, introduced by Cerri et al. [6]. It is a lower bound for the interleaving distance; this is implicit in [6] and shown explicitly in [16]. In the 2-parameter setting, the matching distance is defined as follows: Given a pair of bipersistence modules, we call an affine line  $\ell$  in parameter space with positive slope a *slice*. Restricting the modules to  $\ell$  yields a pair of 1-parameter persistence modules, which we call *slice modules*. These slice modules have a well-defined bottleneck distance, which we multiply by a positive weight to ensure that the matching distance will be a lower bound for the interleaving distance. The matching distance is defined as the supremum of these weighted bottleneck distances over all slices. See Section 3 for the precise definition. The definition generalizes readily to  $n$ -parameter persistence modules, for any  $n \geq 1$ ; when  $n = 1$ , the matching distance is equal to the bottleneck distance.

As Cerri et al. have observed, to approximate the matching distance up to any (absolute) precision, it suffices to sample the space of slices sufficiently densely and to return the maximum weighted bottleneck distance encountered. For a constant number of scale parameters and approximation quality  $\epsilon$ , a polynomial number of slices are sufficient in terms of module size and  $\frac{1}{\epsilon}$ , yielding a polynomial time approximation algorithm. [3]. This approach has been recently applied to the virtual ligand screening problem in computational chemistry [13]. To the best of our knowledge, there is no other previous work in which the problem of computing the matching distance has been considered.

**Our contribution.** We give an algorithm that computes the exact matching distance between a pair of bipersistence modules in time polynomial with respect to the size of the input. We assume that each persistence module is specified by a *presentation*. Concretely, this means that the module is specified by a matrix encoding the generators and relations, with each row and each column labeled by a point in  $\mathbb{R}^2$ ; see Section 2.

To explain our strategy for computing the matching distance, consider the function  $F$  that assigns a slice to its weighted bottleneck distance. The matching distance is then simply the supremum of  $F$ , taken over all slices.  $F$  has a rather complicated structure, since it depends on the longest edge of a perfect matching in a bipartite graph whose edges lengths depend on both the slice and the two modules given as input. When the slice changes, the matching realizing the bottleneck distance undergoes combinatorial changes, making the function  $F$  difficult to treat analytically.

We show, however, that the space of slices can be divided into polynomially many regions so that the restriction of  $F$  to each region takes a simple closed form. Perhaps surprisingly, if we parameterize the space of slices by the right half plane  $\Omega \subset \mathbb{R}^2$ , the boundary between these regions can be expressed by the union of polynomially many lines in  $\Omega$ , making each region convex and bounded by (possibly unbounded) line segments. (This is analogous to the observation of [18] that for a single persistence module, the locus of lines where the combinatorial structure underlying the slice module can change is described by a line arrangement.) Moreover, the restriction of  $F$  to each cell attains its supremum at a boundary vertex of the cell, or as the limit of an unbounded line segment in  $\Omega$ ; this follows from a straightforward case analysis. These observations together lead to a simple polynomial time algorithm to compute the matching distance.

The characterization of the matching distance underlying our algorithm also makes clear that if the row and column labels of the presentations of the input modules have rational coordinates, then the matching distance is rational as well. We are not aware of a simpler argument for this property.

**Outline.** We introduce the underlying topological concepts in Section 2, and introduce the matching distance in Section 3. We define the line arrangement subdividing the slice space in Section 4 and give the algorithm to maximize each cell of the arrangement in Section 5. We conclude in Section 6.

## 2 Persistence modules

**Single-parameter modules.** Let  $\mathbb{K}$  be a fixed finite field throughout. A *persistence module*  $M$  over  $\mathbb{R}$  is an assignment of  $\mathbb{K}$ -vector spaces  $M_x$  to real numbers  $x$ , and linear maps  $M_{x \rightarrow y} : M_x \rightarrow M_y$  to a pair of real numbers  $x \leq y$ , such that  $M_{x \rightarrow x}$  is the identity and  $M_{y \rightarrow z} \circ M_{x \rightarrow y} = M_{x \rightarrow z}$ . Equivalently, in categorical terms, a persistence module is a functor from  $\mathbb{R}$  (considered as a poset category) to the category of  $\mathbb{K}$ -vector spaces.

A common way to arrive at a persistence module is to consider a nested sequence

$$X_1 \subseteq X_2 \subseteq \dots \subseteq X_n$$

of simplicial complexes and to apply homology with respect to a fixed dimension and base field  $\mathbb{K}$ . This yields a sequence

$$H_p(X_1, \mathbb{K}) \rightarrow H_p(X_2, \mathbb{K}) \rightarrow \dots \rightarrow H_p(X_n, \mathbb{K})$$

of vector spaces and linear maps. To obtain a persistence module over  $\mathbb{R}$ , we pick *grades*  $s_1 < s_2 < \dots < s_n$  and set  $M_x := 0$  if  $x < s_1$  and  $M_x := H_p(X_i, \mathbb{K})$  with  $i = \max\{j \mid s_j \leq x\}$  otherwise. For  $y \geq x$  and  $M_y = H_p(X_j, \mathbb{K})$ , we define  $M_{x \rightarrow y}$  as the map  $H_p(X_i, \mathbb{K}) \rightarrow H_p(X_j, \mathbb{K})$  induced by the inclusion map  $X_i \rightarrow X_j$ .

**Finite presentations.** In this work, we restrict our attention to persistence modules that are finitely presented in the following sense. A *finite presentation* is an  $k \times m$  matrix  $P$  over  $\mathbb{K}$ , where each row and each column is labeled by a number in  $\mathbb{R}$ , called the *grade*, such that if  $P_{ij} \neq 0$ , then  $\text{gr}(\text{row}_i) \leq \text{gr}(\text{col}_j)$ ; here  $\text{gr}(-)$  denotes the grade of a row or column. We refer to the multiset of grades of all rows and columns of  $P$  simply as the *set of grades of  $P$* , and denote this set as  $\text{gr}(P)$ . A finite presentation gives rise to a persistence module, as we describe next. The rows of  $P$  represent the generators of the module, while the columns of  $P$  encode relations (or syzygies) on the generators. Concretely, let  $e_1, \dots, e_k$  denote the standard basis of  $\mathbb{K}^k$ , and for  $x \in \mathbb{R}$  define the subset

$$\text{Gen}_x := \{e_i \mid \text{gr}(\text{row}_i) \leq x\}$$

Likewise, define

$$\text{Rel}_x := \{\text{col}_j \mid \text{gr}(\text{col}_j) \leq x\}$$

Then, we define

$$M_x^P := \text{span}(\text{Gen}_x) / \text{span}(\text{Rel}_x)$$

and  $M_{x \rightarrow y}^P$  simply as the map induced by the inclusion map  $\text{span}(\text{Gen}_x) \rightarrow \text{span}(\text{Gen}_y)$ . It can be checked easily that this indeed defines a persistence module  $M^P$ . If a persistence module  $N$  is isomorphic to  $M^P$ , we say that  $P$  is a *presentation of  $N$* . We call a persistence module  $N$  *finitely presented* if there exists a finite presentation of  $N$ . For instance, persistence modules as above arising from a finite simplicial filtration are finitely presented. Also, the representation theorem of persistence [22, 7] states that the category of persistence modules over  $\mathbb{K}$  is isomorphic to the category of graded  $\mathcal{R}$ -modules with an appropriately chosen ring  $\mathcal{R}$ . With that, a persistence module is finitely presented if and only if the corresponding  $\mathcal{R}$ -module is finitely presented (in the sense of a module).

We assume for concreteness that a finite presentation is encoded in terms of a sparse matrix representation [11]; the size of the presentation is understood to be the bitsize of this sparse matrix. In what follows, for finite a presentation with  $k$  generators and  $m$  relations, we will express complexity bounds in terms of  $n := k + m$ , the number of generators and relations. Note that an algorithm polynomial in  $n$  is also polynomial in the size of the presentation.

**Persistence diagrams.** A *persistence diagram* is a finite multi-set of points of the form  $(b, d) \in \mathbb{R} \times (\mathbb{R} \cup \{\infty\})$  with  $b < d$ . A well-known structure theorem tells us that we can associate to any finitely presented persistence module  $M$  a persistence diagram  $D(M)$ , and that this determines  $M$  up to isomorphism [8].

Given a presentation of  $M$ , the persistence diagram can be computed by bringing the presentation matrix into echelon form. This process takes cubic time in  $n$  using Gaussian elimination [11, 22], or  $O(n^\omega)$  time using fast matrix multiplication, where  $\omega \leq 2.373$  [20].

► **Lemma 1.** *For  $P$  a finite presentation of a persistence module  $M$  and  $(b, d) \in D(M)$ ,*

1.  *$b$  is the grade of some row of  $P$ ,*
2. *if  $d < \infty$ , then  $d$  is the grade of some column of  $P$ .*

**Proof.** This follows from the correctness of the basic matrix reduction algorithm for computing persistent homology, as described in [22]. ◀



**Bottleneck distance.** Consider two persistence diagrams  $D_1$  and  $D_2$  and a bijection  $\sigma : D'_1 \rightarrow D'_2$  for  $D'_1 \subseteq D_1$  and  $D'_2 \subseteq D_2$ . For  $\delta > 0$ , we define  $\text{cost}(\sigma) := \max(A, B)$ , where

$$A = \max \{ \max(|a - c|, |b - d|) \mid (a, b) \in D'_1, \sigma(a, b) = (c, d) \},$$

$$B = \max \{ (b - a)/2 \mid (a, b) \in (D_1 \setminus D'_1) \cup (D_2 \setminus D'_2) \},$$

and it is understood that  $\infty - \infty = 0$ . We define the bottleneck distance  $d_B$  by

$$d_B(D_1, D_2) = \min \{ \epsilon \mid \text{there exists a matching of cost } \epsilon \text{ between } D_1 \text{ and } D_2 \}.$$

For persistence modules  $M$  and  $N$ , we write  $d_B(D(M), D(N))$  simply as  $d_B(M, N)$ .

► **Lemma 2.** *Let  $P^M$  and  $P^N$  be finite presentations of persistence modules  $M$  and  $N$ , respectively.  $d_B(M, N)$  is equal to one of the following:*

1. *The difference of a grade of  $P^M$  and a grade of  $P^N$ ,*
2. *half the difference of two grades in  $P^M$ ,*
3. *half the difference of two grades in  $P^N$ .*

**Proof.** This follows immediately from Lemma 1 and the definition of  $d_B$ . ◀

Given two finite persistence diagrams  $D, D'$  with  $m$  points each, we can compute  $d_B(D, D')$  in time to  $O(m^{1.5} \log m)$  [12]; see [15] for details, including a report on practical efficiency. The number of points in a diagram is at most the number of generators of the corresponding module, and hence upper-bounded by  $n$ . Thus, the complexity of computing the bottleneck distance of two persistence modules is dominated by the computation of the persistence diagrams, and has worst-case complexity  $O(n^\omega)$ .

**Bipersistence modules.** The definitions of persistence modules and presentations extend to higher dimensions without difficulty. In the 2-parameter setting, this goes as follows: Define a partial order  $\leq$  on  $\mathbb{R}^2$  by  $p \leq q$  if  $p_x \leq q_x$  and  $p_y \leq q_y$ . A *bipersistence module* is an assignment of  $\mathbb{K}$ -vector spaces  $M_p$  to points  $p \in \mathbb{R}^2$ , and linear maps  $M_{p \rightarrow q} : M_p \rightarrow M_q$  to pairs of points  $p \leq q \in \mathbb{R}^2$ , such that  $M_{p \rightarrow p}$  is the identity and  $M_{q \rightarrow r} \circ M_{p \rightarrow q} = M_{p \rightarrow r}$  whenever  $p \leq q \leq r$ . In topological data analysis, 2-dimensional persistence modules typically arise by applying homology to a bifiltered simplicial complex

A *finite presentation* of a bipersistence module  $\mathbb{R}^2$  is defined in the same way as for one-parameter persistence modules, except that the labels of each row/column are elements of  $\mathbb{R}^2$ , and the  $\leq$  relation appearing in the definition means the partial order over  $\mathbb{R}^2$ . From now on, we will assume that all bipersistence modules considered are finitely presented.

In topological data analysis, we do not typically have immediate access to a presentation of a bipersistence module  $M$ , but rather to a chain complex of bipersistence modules for which  $M$  is a homology module. However, it has recently been observed that from such a chain complex, a (minimal) presentation of  $M$  can be computed in cubic time [19]. The algorithm for this is practical, and has been implemented in the software package RIVET [9].

### 3 The matching distance

**Slices.** We define a *slice* as a line  $\ell : y = sx + t$  where  $s$  and  $t$  are real numbers with  $s > 0$ . Let  $\lambda : \mathbb{R} \rightarrow \ell$  be an order-preserving, isometric parameterization of the slice. Concretely, such a parameterization is given by  $\lambda(x) = \frac{1}{\sqrt{1+s^2}}(x, sx + t)$ . Given a bipersistence module  $M$  and slice  $\ell$ , we define a (1-parameter) persistence module  $M^\ell$  via  $M_x^\ell := M_{\lambda(x)}$ , with its linear maps induced by  $M$ . We call  $M^\ell$  a *slice module*.

**Matching distance.** For a slice  $\ell : y = sx + t$ , we define a weight

$$w(\ell) := \begin{cases} \frac{1}{\sqrt{1+s^2}} & s \geq 1 \\ \frac{1}{\sqrt{1+\frac{1}{s^2}}} & 0 < s < 1 \end{cases}$$

$w(\ell)$  is maximized for slices with slope 1, and gets smaller when the slope goes to 0 or to  $\infty$ .

Let  $\Lambda$  denote the set of all slices. Writing  $\Omega := (0, \infty) \times \mathbb{R}$ , the map  $\alpha : \Omega \rightarrow \Lambda$  that sends  $(s, t)$  to the line  $y = sx + t$  is clearly a bijective parameterization of  $\Lambda$ . We equip  $\Lambda$  with the topology induced by  $\alpha$ , using the subset topology  $\Omega \subset \mathbb{R}^2$ .

For two persistence modules  $M, N$  over  $\mathbb{R}^2$ , we define a function  $F^{M,N} : \Lambda \rightarrow [0, \infty)$  by

$$F^{M,N}(\ell) := w(\ell) \cdot d_B(M^\ell, N^\ell).$$

and we define the *matching distance* between  $M$  and  $N$  as  $d_{\text{match}}(M, N) := \sup F^{M,N}$ . As noted in the introduction, the weights  $w(\ell)$  are chosen to ensure that  $d_{\text{match}}$  is a lower bound for the interleaving distance.

► **Lemma 3.** *Given two bipersistence modules  $M, N$ , the map  $F^{M,N}$  is continuous.*

**Proof.**  $w$  is clearly continuous, so it suffices to show that the function  $\ell \mapsto d_B(M^\ell, N^\ell)$  is continuous. Let  $\mathcal{D}$  denote the metric space of all finite persistence diagrams, with metric the bottleneck distance. It follows from [16, Theorem 2] that the map sending a slice  $\ell$  to the persistence diagram  $D(M^\ell)$  is continuous with respect to the topology on  $\mathcal{D}$ . Thus the map sending  $\ell$  to the pair  $(D(M^\ell), D(N^\ell))$  is also continuous. Moreover, the bottleneck distance is clearly continuous as a map  $\mathcal{D} \times \mathcal{D} \rightarrow [0, \infty)$ , thanks to the triangle inequality. Thus,  $\ell \mapsto d_B(M^\ell, N^\ell)$  is a composition of continuous functions, hence continuous. ◀

## 4 The arrangement

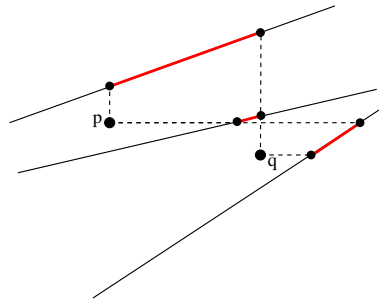
In what follows, we fix two bipersistence modules  $M, N$  (each with at most  $n$  generators and relations) and write  $F := F^{M,N}$ . Using the parameterization  $\alpha$  from above, we also have a function  $F \circ \alpha : \Omega \rightarrow [0, \infty)$ ; slightly abusing notation, we will also denote this map by  $F$ .

In this section, we construct a line arrangement in  $\Omega$  in such a way that it is simple to compute  $\sup F$  on each face. Recall that a *line arrangement* of  $\Omega$  is the subdivision of  $\Omega$  into vertices, edges, and faces induced by a finite set of distinct lines  $L_1, \dots, L_n$ . The vertices of the arrangement are the intersection points of (at least) two lines, the edges are maximal connected subsets of lines not containing any vertex, and the faces are the connected components of  $\Omega \setminus \bigcup_{i=1}^n L_i$ . Clearly, each vertex, edge, and face of the arrangement is a convex set. The boundary of each face consists of a finite number of edges and vertices.

**A first line arrangement.** For  $v \in \mathbb{R}^2$ , let  $L_v$  denote the line  $y = -v_x x + v_y$ . Note that  $L_v \cap \Omega$  is exactly the set of parameterizations of slices containing  $v$ . Now, fix presentations  $P^M$  and  $P^N$  of  $M$  and  $N$ . Let  $\mathcal{A}_0$  denote the arrangement in  $\Omega$  induced by the set of lines

$$\{L_v \mid v \in \text{gr}(P^M) \cup \text{gr}(P^N)\}.$$

In what follows, we will refine  $\mathcal{A}_0$  by adding more lines into the arrangement. For this we first need to introduce some definitions.



■ **Figure 1** The pushes of two points  $p$  and  $q$  to three different slices. The length of the thick (red) line corresponds to the  $\delta_{p,q}$  value of the corresponding slice.

**Pushes.** For a point  $p = (p_x, p_y) \in \mathbb{R}^2$  and a slice  $\ell : y = sx + t$ , we define the *push of  $p$  onto  $\ell$*  as

$$\text{push}(p, \ell) := \begin{cases} (p_x, s \cdot p_x + t) & \text{if } p \text{ lies below } \ell \\ (\frac{p_y - t}{s}, p_y) & \text{if } p \text{ lies on or above } \ell \end{cases}$$

Geometrically,  $\text{push}(p, \ell)$  gives the intersection point of  $\ell$  and a vertical upward ray emanating from  $p$  in the first case, and the intersection of  $\ell$  with a horizontal right ray emanating from  $p$  in the second case. See Figure 1 for an illustration.

► **Remark 4.** A finite presentation of  $M$  induces a finite presentation of  $M^\ell$  with the same underlying matrix, and each row or column grade  $p \in \mathbb{R}^2$  replaced with  $\lambda^{-1}(\text{push}(p, \ell))$ . Clearly, this presentation can be obtained in time linear in the size of the presentation of  $M$ .

For  $p, q \in \mathbb{R}^2$ , define  $\delta_{p,q} : \Omega \rightarrow [0, \infty)$  by

$$\delta_{p,q}(s, t) := \|\text{push}(p, \ell) - \text{push}(q, \ell)\|_2 = |\lambda^{-1} \circ \text{push}(p, \ell) - \lambda^{-1} \circ \text{push}(q, \ell)|$$

with  $\ell$  the slice defined by  $s$  and  $t$ . Again, see Figure 1 for an illustration.

We now give piecewise analytic formulae for  $\delta_{p,q}$ , which depend on whether the slice  $\ell$  is below or above  $p$  and  $q$ .

**(I) slice is below both  $p$  and  $q$ .**

$$\delta_{p,q}(s, t) = \|(\frac{p_y - t}{s}, p_y) - (\frac{q_y - t}{s}, q_y)\|_2 = \sqrt{\left(\frac{p_y - q_y}{s}\right)^2 + (p_y - q_y)^2} = |p_y - q_y| \sqrt{1 + \frac{1}{s^2}}.$$

**(II) slice is above both  $p$  and  $q$ .**

$$\delta_{p,q}(s, t) = \|(p_x, sp_x + t) - (q_x, sq_x + t)\|_2 = \sqrt{(p_x - q_x)^2 + (sp_x - sq_x)^2} = |p_x - q_x| \sqrt{1 + s^2}.$$

**(III) slice is between  $p$  and  $q$ .** There are two subcases, which we will call (IIIa) and (IIIb): Assuming  $p$  lies above the slice (IIIa), the formula is

$$\begin{aligned} \delta_{p,q}(s, t) &= \|(\frac{p_y - t}{s}, p_y) - (q_x, sq_x + t)\|_2 = \sqrt{\left(\frac{p_y - t}{s} - q_x\right)^2 + (p_y - sq_x - t)^2} \\ &= \sqrt{\frac{1}{s^2} (p_y - t - sq_x)^2 + (p_y - sq_x - t)^2} = |p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}}. \end{aligned}$$

If  $p$  lies below the slice (IIIb), the formula is the same, with the roles of  $p$  and  $q$  exchanged.

The push map is easily seen to be continuous with respect to the slice  $\ell$ , so these formulae also extend to boundaries of the cases, i.e., when the slice contains  $p$  or  $q$ .

► **Lemma 5.** *If  $p, q \in \text{gr}(P^M) \cup \text{gr}(P^N)$ , then in each face of  $\mathcal{A}_0$ , exactly one of the conditions (I), (II), (IIIa), (IIIb) holds everywhere. Hence,  $\delta_{p,q}$  can be expressed on the entire face by one of the analytic formulae above.*

**Proof.** Assume for a contradiction that two points  $x, x'$  in a face  $f$  of  $\mathcal{A}_0$  are of two different types among (I), (II), (IIIa), (IIIb). Let  $\ell = \alpha(x)$ ,  $\ell' = \alpha(x')$  denote the corresponding slices. Then either  $p$  or  $q$  (or both) switch sides from  $\ell$  to  $\ell'$ . Assume that  $p$  lies above  $\ell$  and  $p$  lies below  $\ell'$  – all other cases are analogous. Then, on the line segment from  $x$  to  $x'$ , there is some point  $x''$  such that  $p$  lies on the slice  $\alpha(x'')$ . Hence,  $x''$  lies on  $L_p$  and is therefore not in  $f$ . This contradicts the convexity of the faces. ◀

In view of Lemma 5, for  $p, q \in \text{gr}(P^M) \cup \text{gr}(P^N)$  we may define the *type* of  $\delta_{p,q}$  on a face of  $\mathcal{A}_0$  to be the case (I), (II), (IIIa), or (IIIb) which holds on that face.

**Refinement of the arrangement.** Now we further subdivide the arrangement  $\mathcal{A}_0$ . For that, consider the set of equations of the form

$$\begin{aligned} \delta_{p,q}(s, t) &= 0 && \text{for } p, q \in \text{gr}(P^M) \text{ or } p, q \in \text{gr}(P^N), \\ c_{pq}\delta_{p,q}(s, t) &= c_{p'q'}\delta_{p',q'}(s, t) && \text{for } p, q, p', q' \in \text{gr}(P^M) \sqcup \text{gr}(P^N), \end{aligned} \quad (1)$$

where

$$c_{pq} := \begin{cases} \frac{1}{2} & \text{if } p, q \in \text{gr}(P^M) \text{ or } p, q \in \text{gr}(P^N), \\ 1 & \text{otherwise.} \end{cases}$$

► **Lemma 6.** *The solution set of each of the above equations restricted to a face  $f$  is either the empty set, the entire face, the intersection of  $f$  with a line, or intersection of  $f$  with the union of two lines.*

**Proof.** First we show the statement for equations of type  $\delta_{p,q}(s, t) = 0$ . There are 3 cases:  $\delta_{p,q}$  is of type (I). the equation becomes

$$|p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = 0$$

for which either all  $(s, t) \in f$  are a solution (if  $p_y = q_y$ ), or no  $(s, t)$  is a solution.

$\delta_{p,q}$  is of type (II). the same argument holds for the equation

$$|p_x - q_x| \sqrt{1 + s^2} = 0.$$

$\delta_{p,q}$  is of type (III). Swapping  $p$  and  $q$  if necessary, we obtain the equation

$$|p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}} = 0$$

and the solution set is made of all  $(s, t) \in f$  for which  $p_y - t - sq_x = 0$ , which is the equation of a line.

For the remaining equations, we give the proof in the special case that  $c_{pq} = c_{p'q'}$ ; the proof in the other cases is essentially the same. For equations of the form  $\delta_{p,q}(s, t) = \delta_{p',q'}(s, t)$ , there are six cases to check, depending on the type of the  $\delta$ -functions on the left and right sides of the equation:

Both  $\delta_{p,q}(s, t)$  and  $\delta_{p',q'}(s, t)$  are of type (I). the equation is

$$|p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = |p'_y - q'_y| \sqrt{1 + \frac{1}{s^2}}$$

and the equation is satisfied if and only if  $|p_y - q_y| = |p'_y - q'_y|$ , independent of  $s$  and  $t$ . Hence, the solution set is either  $f$  or  $\emptyset$ .

Both  $\delta_{p,q}(s, t)$  and  $\delta_{p',q'}(s, t)$  are of type (II). the same argument as in the previous case applies, so the solution set is either  $f$  or  $\emptyset$ .

$\delta_{p,q}$  is of type (I) and  $\delta_{p',q'}$  of type (II). we get the equation

$$|p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = |p'_x - q'_x| \sqrt{1 + s^2}.$$

Since  $1 + \frac{1}{s^2} = \frac{1+s^2}{s^2}$ , this simplifies to

$$|p_y - q_y| = s|p'_x - q'_x|$$

and the solution set is either all of  $f$  (if both absolute values vanish), the empty set (if only  $p'_x - q'_x = 0$ ), or the intersection of  $f$  with the vertical line  $s = \frac{|p_y - q_y|}{|p'_x - q'_x|}$  (otherwise).

Both  $\delta_{p,q}$  and  $\delta_{p',q'}$  are of type (III). Swapping  $p, q$  or  $p', q'$  if necessary, we get

$$|p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}} = |p'_y - t - sq'_x| \sqrt{1 + \frac{1}{s^2}}$$

hence,  $(s, t)$  is a solution if and only if  $p_y - t - sq_x = p'_y - t - sq'_x$  or  $p_y - t - sq_x = -(p'_y - t - sq'_x)$ . The first equations yields again either  $f$ ,  $\emptyset$ , or a vertical line as solution set, the second equation always defines a line. Exchanging the roles of  $p$  and  $q$ , or the roles of  $p'$  and  $q'$ , or both, does not change the conclusion.

$\delta_{p,q}$  is of type (I) and  $\delta_{p',q'}$  is of type (III). Swapping  $p'$  and  $q'$  if necessary, the formula is

$$|p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = |p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}}.$$

$(s, t) \in f$  is a solution if  $p_y - t - sq_x = p_y - q_y$  or  $p_y - t - sq_x = q_y - p_y$ , which is a line equation in both cases.

$\delta_{p,q}$  is of type (II) and  $\delta_{p',q'}$  is of type (III). Swapping  $p'$  and  $q'$  if necessary, we get

$$|p_x - q_x| \sqrt{1 + s^2} = |p'_y - t - sq'_x| \sqrt{1 + \frac{1}{s^2}}$$

which simplifies to

$$s|p_x - q_x| = |p'_y - t - sq'_x|$$

Here  $(s, t) \in f$  is a solution if and only if  $s(p_x - q_x) = p'_y - t - sq'_x$  or  $s(p_x - q_x) = -(p'_y - t - sq'_x)$ . Again, we obtain a line in both cases. ◀

► **Definition 7.** Let  $\mathcal{A}$  denote the line arrangement in  $\Omega$  formed by the lines in  $\mathcal{A}_0$ , all lines from the case analysis above, and the vertical line  $s = 1$ .

► **Lemma 8.** The arrangement  $\mathcal{A}$  consists of  $O(n^4)$  lines.

**Proof.** The case analysis in the proof of Lemma 6 was performed relative to a choice of face  $f$  in  $\mathcal{A}_0$ . However, for a fixed choice of an equation in the set of equations (1), the lines which arise in the case analysis depend only on the types of  $\delta_{p,q}$  and  $\delta_{p',q'}$  on  $f$ . There are at most  $4 \times 4 = 16$  possible ways of jointly choosing those types, and for a given choice, at most two lines are added to the arrangement. Hence, each of the  $O(n^4)$  equations in the set of equations (1) contributes at most a constant number of lines to  $\mathcal{A}$ . The result now follows easily.  $\blacktriangleleft$

Note that the arrangement  $\mathcal{A}$  depends on the choice of presentations for  $M$  and  $N$ . The next statement says that within a face of the arrangement, the bottleneck distance is realized by the difference of the pushes of two fixed grades of the presentations.

► **Theorem 9.** *For any face  $f$  of  $\mathcal{A}$ , there is some choice of  $p, q \in \text{gr}(P^M) \cup \text{gr}(P^N)$  such that  $d_B(M^\ell, N^\ell) = c_{pq}\delta_{p,q}(\ell)$  for all  $\ell \in \alpha(f)$ .*

See the full version [14, App. A] for a complete proof. It can be summarized as follows: By Remark 4 and Lemma 1, for each  $\ell \in f$  there is a collection  $T_M^\ell$  of pairs

$$(b, d) \in \text{gr}(P^M) \times (\text{gr}(P^M) \cup \{(\infty, \infty)\})$$

such that  $D(M^\ell)$  is obtained by pushing the elements of  $T_M^\ell$  onto  $\ell$ . Call such  $T_M^\ell$  a *template* for  $D(M^\ell)$ . For  $\ell$  and  $\ell'$  such that the grades of  $P^M$  push onto  $\ell$  and  $\ell'$  in the same order, the templates for  $D(M^\ell)$  and  $D(M^{\ell'})$  are the same. But in fact, the grades of  $P^M$  push onto all lines  $\ell \in f$  in the same order, because whenever the order changes, we need to cross one of the lines of the arrangement  $\mathcal{A}$ . Hence, there exists  $T_M^f$  that is a template for  $D(M^\ell)$ , for all  $\ell \in f$ . Similarly for  $N$ .

By Lemma 2, for any fixed  $\ell' \in f$  we have that  $d_B(M^{\ell'}, N^{\ell'}) = c_{pq}\delta_{p,q}(\ell')$ , for  $p$  and  $q$  each some coordinate of a pair in  $T_M^f \cup T_N^f$ . The order of the values of the functions

$$\{c_{rs}\delta_{r,s} \mid r, s \in \text{gr}(P^M) \cup \text{gr}(P^N)\}$$

remains the same across  $f$ , because any change in the order will result again in crossing one of the lines of the arrangement  $\mathcal{A}$ . Thus, in fact  $d_B(M^\ell, N^\ell) = c_{pq}\delta_{p,q}(\ell)$  for all  $\ell \in f$ .

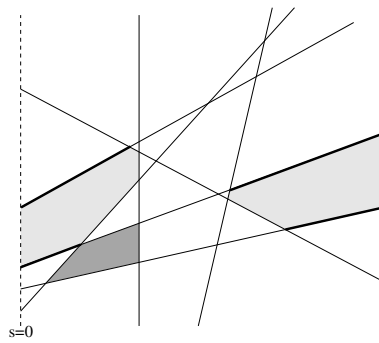
## 5 Maximization

We define a *region* of  $\mathcal{A}$  as the closure of a face of  $\mathcal{A}$  within  $\Omega$ . We can compute the matching distance by determining  $\sup F(s, t)$  separately for each region in  $\mathcal{A}$ . We will show now that in each region,  $\sup F(s, t)$  is either realized at a boundary vertex, or as the limit of an unbounded boundary edge, which can be computed easily.

We fix the following notation: A region  $R \subseteq \Omega$  is an *inner region* if it is bounded as a set in  $\mathbb{R}^2$  and it has a positive distance to the vertical line  $s = 0$  in  $\mathbb{R}^2$  (in other words,  $R$  does not reach the boundary of  $\Omega$ ). An inner region is a convex polygon. Regions that are not inner region are called *outer regions*. Outer regions have exactly two *outer segments* in their boundary, which are infinite or converge to a point on the vertical line  $s = 0$  (except for the empty arrangement, which only consists of one outer region). See Figure 2 for an illustration.

For a fixed region  $R$  of  $\mathcal{A}$ , Theorem 9 ensures that there is a pair of grades  $(p, q)$  whose  $\delta$ -function realizes  $d_B$  within the interior of  $R$  (a face of  $\mathcal{A}$ ). By continuity, this implies that  $\delta_{p,q}$  also realizes  $d_B$  on the entire region.

► **Lemma 10.** *The supremum of  $F$  within  $R$  is attained either at a vertex on the boundary of  $R$ , or as the limit of  $F$  along an outer segment. In the latter case, the limit can be expressed in simple terms based on the equation of the line segment and one of the functions  $\delta_{p,q}$ .*



■ **Figure 2** The lightly shaded regions show outer regions. The outer segments of these are drawn more thickly. The darkly shaded region is an example of an inner region.

**Proof.** We distinguish 6 cases based on the type of the  $\delta$ -function and on whether  $s \leq 1$  or  $s \geq 1$  (note that each cell belongs to one of these cases, because the line  $s = 1$  is in  $\mathcal{A}$ ).

$\delta_{p,q}$  of type (I),  $s \leq 1$ . In that case,

$$F(\ell) = w(\ell)\delta_{p,q}(\ell) = \frac{1}{\sqrt{1 + \frac{1}{s^2}}} |p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = |p_y - q_y|,$$

a constant function. Clearly, the supremum is attained everywhere, in particular at the boundary vertices of  $R$ .

$\delta_{p,q}$  of type (I),  $s \geq 1$ . We get

$$F(\ell) = \frac{1}{\sqrt{1 + s^2}} |p_y - q_y| \sqrt{1 + \frac{1}{s^2}} = \frac{1}{s} |p_y - q_y|$$

Clearly, this function becomes larger when  $s$  gets smaller. Moreover, because  $s \geq 1$  within the cell, there is a leftmost vertex on the boundary, which minimizes  $s$  and therefore attains the supremum within the cell.

$\delta_{p,q}$  of type (II),  $s \leq 1$ . We obtain

$$F(\ell) = \frac{1}{\sqrt{1 + \frac{1}{s^2}}} |p_x - q_x| \sqrt{1 + s^2} = s |p_x - q_x|.$$

Similarly to the previous case, there exists a rightmost boundary vertex in the cell (because  $s \leq 1$ ), which realizes the supremum.

$\delta_{p,q}$  of type (II),  $s \geq 1$ . The function simplifies to

$$F(\ell) = \frac{1}{\sqrt{1 + s^2}} |p_x - q_x| \sqrt{1 + s^2} = |p_x - q_x|,$$

a constant function, which attains its supremum at any boundary vertex.

$\delta_{p,q}$  of type (III),  $s \leq 1$ . Assuming that  $p$  lies above the slice, we get

$$F(\ell) = \frac{1}{\sqrt{1 + \frac{1}{s^2}}} |p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}} = |p_y - t - sq_x|.$$

If  $R$  is an inner region, we are maximizing the above function over a closed convex polygon, and the maximum is achieved at a boundary vertex, because  $|p_y - t - sq_x|$  is the maximum of two linear functions in  $s$  and  $t$ .



It remains to analyze the case that  $R$  is an outer region. We argue first that  $R$  is bounded in  $t$ -direction from above and below: Since  $\delta_{p,q}$  is of type (III), with  $p$  lying above  $\ell$ ,  $(s, t)$  must be below the (non-vertical) line  $t = -sp_x + p_y$  in the dual space. Likewise, since  $q$  is below  $\ell$ ,  $(s, t)$  must be above  $t = -sq_x + q_y$ . Moreover, we have  $0 < s \leq 1$ . If the above lines intersect at a point  $r$  with  $s$ -value in  $(0, 1)$ ,  $R$  is contained in the triangle spanned by the two lines and the vertical line  $s = 0$ . Otherwise,  $R$  is contained in the trapezoid induced by these two lines and the vertical lines  $s = 0$  and  $s = 1$ .

It follows that the two outer segments of  $R$  converge to the vertical line  $s = 0$ . Let  $(0, t_1)$  denote the limit of the lower outer segment and  $(0, t_2)$  the limit of the upper outer segment. Clearly  $t_1 \leq t_2$ . Let  $\bar{R}$  denote the union of  $R$  with the vertical line segment from  $(0, t_1)$  to  $(0, t_2)$ ; note that  $\bar{R}$  is the closure of  $R$  considered as a subset of  $\mathbb{R}^2$ . Observe that  $|p_y - t - sq_x|$  is continuous over  $\mathbb{R}^2$ ; therefore  $F$  can be continuously extended to  $\bar{R}$ . It follows that the supremum of  $F$  over  $\bar{R}$  is attained at a boundary vertex, since  $\bar{R}$  is a convex closed polygon. There are two cases: either the maximum is attained at a vertex of  $\mathcal{A}$ , or at  $(0, t_1)$  or  $(0, t_2)$ . As we can readily see, the function values at the latter two points are  $|p_y - t_1|$  and  $|p_y - t_2|$ , respectively. The case where  $p$  is below the slice and  $q$  is above is analyzed in the same way, with the roles of  $p$  and  $q$  swapped.

$\delta_{p,q}$  of type (III),  $s \geq 1$  Assuming that  $p$  lies above the slice, we get

$$F(\ell) = \frac{1}{\sqrt{1+s^2}} |p_y - t - sq_x| \sqrt{1 + \frac{1}{s^2}} = \left| \frac{p_y}{s} - \frac{t}{s} - q_x \right|$$

We first consider the case where  $R$  is an inner region, and we show that the function is maximized at a boundary vertex of  $R$ . The function  $|\frac{p_y}{s} - \frac{t}{s} - q_x|$  has no local maximum over  $\mathbb{R}^2$  since it is the absolute value of a linear function in  $t$  for any fixed  $s$ . Hence, the supremum over  $R$  must be attained on the boundary. We have to exclude the case that the maximum lies in the interior of an edge. For vertical edges this is obvious, because for a constant  $s$ , the function simplifies to the absolute value of a linear function in  $t$  which must be maximized at a boundary vertex. For a non-vertical line of the form  $t = as + b$ , plugging in this equation for  $t$  yields a function of the form

$$\left| \frac{p_y}{s} - \frac{as + b}{s} - q_x \right| = \left| \frac{1}{s}(p_y - b) - a - q_x \right|.$$

This is the absolute value of a monotone function in  $s$  and hence has no local maximum. Again, this implies that it is maximized at a boundary vertex.

Consider now the case where  $R$  is an outer region. As in the previous case,  $R$  is upper and lower bounded by two non-vertical lines, because we assume type (III). Hence, the two outer segments of  $R$  cannot be vertical; the lower outer segment has a slope  $r_1$  and the upper outer segment has a slope  $r_2$  with  $r_1 < r_2$ . We argue next that the supremum of  $\frac{p_y}{s} - \frac{t}{s} - q_x$  is either attained at a boundary vertex, or equal to  $|r_1 + q_x|$ , or equal to  $|r_2 + q_x|$ . Let  $(s_i, t_i)$  denote a sequence of points in  $R$  such that  $F(s_i, t_i)$  converges to the supremum. If  $(s_i, t_i)$  converges to a point in  $R$  (or has at least a convergent subsequence), it follows (similarly to the case of an inner region) that the limit point is a boundary vertex. Otherwise, we can assume (by passing to a subsequence) that the sequence  $s_i$  is unbounded. Moreover, the sequence  $\frac{t_i}{s_i}$  is bounded by  $[r_1, r_2]$  and therefore has a convergent subsequence with limit  $r'$ . Passing to this subsequence, we obtain that

$$\lim_{i \rightarrow \infty} F(s_i, t_i) = \lim_{i \rightarrow \infty} \left| \frac{p_y}{s_i} - \frac{t_i}{s_i} - q_x \right| = |-r' - q_x| = |r' + q_x|$$

Hence, the supremum must be of the form  $|r' + q_x|$  for some  $r' \in [r_1, r_2]$ . On the other hand, this expression is clearly maximized for either  $|r_1 + q_x|$  or  $|r_2 + q_x|$ , and there exist sequences attaining these values, for instance when choosing  $(s_i, t_i)$  on either of the outer segments. The case where  $p$  lies below the slice and  $q$  lies above is treated similarly, with the roles of  $p$  and  $q$  exchanged. ◀

**The algorithm.** We now give the algorithm to compute the matching distance:

- Compute the arrangement induced by  $\mathcal{A}$  from Definition 7.
- For each vertex  $(s, t)$  in the arrangement, compute  $F(s, t)$ . Let  $m$  be the maximum among all the values.
- For each outer region  $R$ , pick a point  $(s, t)$  in the interior. Compute the bottleneck distance and identify a pair  $(p, q)$  of grades that realizes the bottleneck. Determine whether  $p$  and  $q$  are above or below the slice  $(s, t)$ . If the region is of type (IIIa) with respect to  $p$  and  $q$ , then do the following:
  - If  $R$  is to the left of  $s = 1$ , compute the intersections  $(0, t_1), (0, t_2)$  of the outer segments of  $R$  with the vertical line  $s = 0$ . Set  $m \leftarrow \max\{m, |p_y - t_1|, |p_y - t_2|\}$ .
  - If  $R$  is to the right of  $s = 1$ , let  $r_1, r_2$  denote the slopes of the outer segments of  $R$ . Set  $m \leftarrow \max\{m, |r_1 + q_x|, |r_2 + q_x|\}$ .
 If the region is of type (IIIb), proceed analogously, with the roles of  $p$  and  $q$  exchanged.
- Return  $m$ .

By “computing the arrangement”, we mean to store the planar subdivision induced by the lines of the arrangement (e.g.[2, Ch.2]). In fact, it is possible to implement the algorithm without explicitly constructing the line arrangement  $\mathcal{A}$  or even storing its entire set of vertices. This reduces the space complexity of the algorithm. See [14, App. B] for details.

► **Theorem 11.** *The above algorithm computes the matching distance in polynomial time.*

**Proof.** Correctness follows from Lemma 10: as we check all vertices of the arrangement, we cover the supremum of all inner regions. The outer regions are handled separately in the last steps of the algorithm.

Running time: recall from Lemma 8 that we have  $O(n^4)$  lines in the arrangement  $\mathcal{A}$ . Hence, the arrangement has  $O(n^8)$  vertices,  $O(n^4)$  outer regions, and can be computed in  $O(n^8 \log n)$  time using an extension of the Bentley-Ottman sweep-line algorithm [1]. For each vertex and each outer region, we have to compute two persistence diagrams, which can be done in  $O(n^3)$  time, and a bottleneck distance whose complexity can be neglected. The remaining computations are negligible. Hence, we arrive at a  $O(n^{11})$  algorithm. ◀

We remark that the algorithm can be realized entirely with rational arithmetic if all grades are rational numbers. Indeed, all lines in the arrangement have rational coefficients, and so do their intersection points. An intersection points corresponds to a slice along which we are required to compute the bottleneck distance. Recall from Section 3 that the definition of the slice  $\ell$  module introduces a grade of  $\lambda^{-1}(\text{push}(p, \ell))$  where  $\lambda^{-1}(p_x, p_y) = \sqrt{1 + s^2}p_x$ . Hence, these grades are not rational numbers. However, the bottleneck distance is multiplied with the weight  $w(\ell)$  of the slice afterwards, and instead of doing so, one can as well scale all grades with  $w(\ell)$  in advance. A simple calculation shows that this indeed turns the grades into rational values.

A simple analysis also reveals that if the input coordinates are rational and of bitsize  $\leq b$ , all intermediate computations in the algorithm can be performed with a bitsize of  $\leq cb$ , with  $c$  a (small) constant. Hence, the algorithm is strongly polynomial.

## 6 Discussion

We have presented the first polynomial time algorithm to exactly compute the matching distance for 2-parameter persistence modules. It is natural to ask about practicality of our approach. The large exponent of  $n^{11}$  seems discouraging at first, but we mention first that the worst-case running time of  $O(n^3)$  for persistent homology is usually not appearing for real instances; indeed an almost linear behavior can be expected. Still, the large number of  $O(n^4)$  lines in the arrangement constitutes a computational barrier in practice. There are several possibilities, however, to reduce this effect:

- Instead of computing the arrangement  $\mathcal{A}$  globally, we could compute the intermediate arrangement  $\mathcal{A}_0$  and refine each face of it separately, using only those lines that affect the  $\delta$ -functions within this face.
- As a follow-up to the previous point, it might be possible to compute a smaller arrangement per face adaptively. The idea is to start at some interior point in a face of  $\mathcal{A}_0$ , identifying a pair  $(p, q)$  that realizes the bottleneck distance and then to determine the boundary of the region where  $(p, q)$  realizes the bottleneck distance.
- As a preprocessing step, we can minimize the input presentations, yielding presentations for the same modules with the smallest possible number of generators and relations (hence minimizing  $n$ ) [19].

We pose the question of whether an implementation realizing the above ideas is competitive to an approximative, sampling-based approach for computing the matching distance.

Our algorithm needs to treat the outer edges of the arrangement  $\mathcal{A}$  separately since our analysis does not rule out the possibility that the supremum is realized at the boundary of  $\Omega$ . On the other hand, we are not aware of an example of two finite presentations whose matching distance is not realized by a particular slice in  $\Omega$ . A proof that the supremum in the definition of the matching distance is in fact a maximum would greatly simplify our algorithm, since it would boil down to computing the intersection points of all lines and searching for the maximal  $F$ -value among them.

Finally, we have restricted attention to the case of two-parameter persistence modules. It is natural to conjecture that our algorithm extends to more parameters by constructing a hyperplane arrangement. It would be worthwhile to check this conjecture in future work.

---

## References

- 1 J. Bentley and T. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Transactions on Computers*, 28:643–647, 1979.
- 2 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.
- 3 S. Biasotti, A. Cerri, P. Frosini, and D. Giorgi. A new algorithm for computing the 2-dimensional matching distance between size functions. *Pattern Recognition Letters*, 32(14):1735–1746, 2011.
- 4 H. Bjerkevik, M. Botnan, and M. Kerber. Computing the interleaving distance is NP-hard. *arXiv*, abs/1811.09165, 2018. [arXiv:1811.09165](https://arxiv.org/abs/1811.09165).
- 5 G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete & Computational Geometry*, 42(1):71–93, 2009.
- 6 A. Cerri, B. Di Fabio, M. Ferri, P. Frosini, and C. Landi. Betti numbers in multidimensional persistent homology are stable functions. *Mathematical Methods in the Applied Sciences*, 36(12):1543–1557, 2013.

- 7 R. Corbet and M. Kerber. The representation theorem of persistence revisited and generalized. *Journal of Applied and Computational Topology*, 2(1):1–31, 2018.
- 8 W. Crawley-Boevey. Decomposition of pointwise finite-dimensional persistence modules. *Journal of Algebra and its Applications*, 14(05):1550066, 2015.
- 9 The RIVET Developers. RIVET: Software for visualization and analysis of 2-parameter persistent homology. <http://repo.rivet.online/>, 2014–2018.
- 10 T. Dey and C. Xin. Computing Bottleneck Distance for 2-D Interval Decomposable Modules. In *International Symposium on Computational Geometry (SoCG)*, pages 32:1–32:15, 2018.
- 11 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, Providence, RI, USA, 2010.
- 12 A. Efrat, A. Itai, and M. Katz. Geometry Helps in Bottleneck Matching and Related Problems. *Algorithmica*, 31(1):1–28, 2001. doi:10.1007/s00453-001-0016-8.
- 13 B. Keller, M. Lesnick, and T. L. Willke. Persistent Homology for Virtual Screening. *ChemRxiv preprint*, October 2018.
- 14 M. Kerber, M. Lesnick, and S. Oudot. Exact computation of the matching distance on 2-parameter persistence modules. *CoRR*, abs/1812.09085, 2018.
- 15 M. Kerber, D. Morozov, and A. Nigmatov. Geometry Helps to Compare Persistence Diagrams. *Journal of Experimental Algorithms*, 22:1.4:1–1.4:20, September 2017.
- 16 C. Landi. The Rank Invariant Stability via Interleavings. In *Research in Computational Topology*, pages 1–10. Springer International Publishing, 2018.
- 17 M. Lesnick. The theory of the interleaving distance on multidimensional persistence modules. *Foundations of Computational Mathematics*, 15(3):613–650, 2015.
- 18 M. Lesnick and M. Wright. Interactive visualization of 2-D persistence modules. *arXiv:1512.00180*, 2015. arXiv:1512.00180.
- 19 M. Lesnick and M. Wright. Computing Minimal Presentations and Betti Numbers of 2-Parameter Persistent Homology. *arXiv:1902.05708*, 2019. arXiv:1902.05708.
- 20 N. Milosavljevic, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *ACM Symposium on Computational Geometry (SoCG)*, pages 216–225, 2011.
- 21 S. Oudot. *Persistence theory: From Quiver Representation to Data Analysis*, volume 209 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2015.
- 22 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005.



# Probabilistic Smallest Enclosing Ball in High Dimensions via Subgradient Sampling

Amer Krivošija

Department of Computer Science, TU Dortmund, Germany  
amer.krivosija@tu-dortmund.de

Alexander Munteanu

Department of Computer Science, TU Dortmund, Germany  
alexander.munteanu@tu-dortmund.de

---

## Abstract

---

We study a variant of the median problem for a collection of point sets in high dimensions. This generalizes the geometric median as well as the (probabilistic) smallest enclosing ball (pSEB) problems. Our main objective and motivation is to improve the previously best algorithm for the pSEB problem by reducing its exponential dependence on the dimension to linear. This is achieved via a novel combination of sampling techniques for clustering problems in metric spaces with the framework of stochastic subgradient descent. As a result, the algorithm becomes applicable to shape fitting problems in Hilbert spaces of unbounded dimension via kernel functions. We present an exemplary application by extending the support vector data description (SVDD) shape fitting method to the probabilistic case. This is done by simulating the pSEB algorithm implicitly in the feature space induced by the kernel function.

**2012 ACM Subject Classification** Theory of computation → Design and analysis of algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Computational geometry

**Keywords and phrases** geometric median, convex optimization, smallest enclosing ball, probabilistic data, support vector data description, kernel methods

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.47

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1902.10966>.

**Funding** This work was supported by the German Science Foundation (DFG) Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, projects A2 and C4.

## 1 Introduction

The (probabilistic) smallest enclosing ball (pSEB) problem in  $\mathbb{R}^d$  is to find a center that minimizes the (expected) maximum distance to the input points (see Definition 13). It occurs often as a building block for complex data analysis and machine learning tasks like estimating the support of high dimensional distributions, outlier detection, novelty detection, classification and robot gathering [9, 27, 28, 30]. It is thus very important to develop highly efficient approximation algorithms for the base problem. This involves reducing the number of points but also keeping the dependence on the dimension as low as possible. Both objectives will be studied in this paper. We will focus on a small dependence on the dimension. This is motivated as follows.

Kernel methods are a common technique in machine learning. These methods implicitly project the  $d$ -dimensional input data into much larger dimension  $D$  where simple linear classifiers or spherical data fitting methods can be applied to obtain a non-linear separation or non-convex shapes in the original  $d$ -dimensional space. The efficiency of kernel methods is usually not harmed. Despite the large dimension  $D \gg d$ , most important kernels, and thus inner products and distances in the  $D$ -dimensional space, can be evaluated in  $O(d)$  time [25].



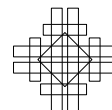
© Amer Krivošija and Alexander Munteanu;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 47; pp. 47:1–47:14

Leibniz International Proceedings in Informatics



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



In some cases, however, a proper approximation relying on sampling and discretizing the ambient solution space may require a polynomial or even exponential dependence on  $D$ . The algorithm of Munteanu et al. [22] is the only fully polynomial time approximation scheme (FPTAS) and the fastest algorithm to date for the pSEB problem in fixed dimension. However, it suffers from the stated problems. In particular, the number of realizations sampled by their algorithm had a linear dependence on  $D$  stemming from a ball-cover decomposition of the solution space. The actual algorithm made a brute force evaluation (on the sample) of all centers in a grid of exponential size in  $D$ . This is prohibitive in the setting of kernel methods since the implicit feature space may have infinite dimension. Even if it is possible to exploit the up to  $n$ -dimensional subspace spanned by  $n$  points in infinite dimensions, we would still have  $D = n \gg d$  leading to exponential time algorithms.

To make the probabilistic smallest enclosing ball algorithm viable in the context of kernel methods and generally in high dimensions, it is highly desirable to reduce the dependence on the dimension to a small polynomial occurring only in evaluations of inner products and distances between two (low dimensional) vectors.

## 1.1 Related work

**Probabilistic smallest enclosing ball.** The study of *probabilistic clustering problems* was initiated by Cormode and McGregor [11]. They developed approximation algorithms for the probabilistic settings of  $k$ -means,  $k$ -median as well as  $k$ -center clustering. For the metric 1-center clustering problem their results are bi-criteria  $O(1)$ -approximation algorithms with a blow-up on the number of centers. Guha and Munagala [14] improved the previous work by giving  $O(1)$ -approximations that preserve the number of centers. Munteanu et al. [22] gave the first fully polynomial time  $(1 + \varepsilon)$ -approximation scheme (FPTAS) for the probabilistic Euclidean 1-center, i.e., the probabilistic smallest enclosing ball problem, in fixed dimensions. The algorithm runs in linear time for sampling a constant number of realizations. Solving the subsampled problem takes only constant time, though exponential in the dimension. This yields a total running time of roughly  $O(nd/\varepsilon^{O(1)} + 1/\varepsilon^{O(d)})$ . Based on  $\varepsilon$ -kernels for probabilistic data [16], Huang and Li [15] generalized the  $(1 + \varepsilon)$ -approximation of [22] to a polynomial time approximation scheme (PTAS) for Euclidean  $k$ -center in  $\mathbb{R}^d$  for fixed constants  $k$  and  $d$ . We note that the running time of the algorithm in [15] grows as a double exponential function of the dimension and it is unclear how to reduce this. We thus base our work on the FPTAS of [22] and reduce its exponential dependence to linear.

**Sampling techniques for 1-median.** The work of [22] showed a reduction from the probabilistic smallest enclosing ball to 1-median problems on (*near*)-*metric* spaces. We thus review relevant results on sampling techniques for metric 1-median. Bădoiu et al. [7] showed that with constant probability, the span of a uniform sample of a constant number of input points contains a  $(1 + \varepsilon)$ -approximation for the 1-median. This was used to construct a set of candidate solutions of size  $O(2^{1/\varepsilon^{O(1)}} \log n)$ . A more refined estimation procedure based again on a small uniform sample was used by Kumar et al. [19] to reduce this to  $O(2^{1/\varepsilon^{O(1)}})$ . Indyk and Thorup [17, 29] showed that a uniform sample of size  $O(\log n/\varepsilon^2)$  is sufficient to approximate the discrete metric 1-median on  $n$  points within a factor of  $(1 + \varepsilon)$ . Ackermann et al. [1] showed how this argument can be adapted to doubling spaces which include the continuous Euclidean space. We adapt these ideas to find a  $(1 + \varepsilon)$ -approximation to the best center in our setting.

**Stochastic subgradient descent.** One quite popular and often only implicitly used technique in the coresets literature is derived from convex optimization, see [21]. One of the first results of that kind is given in the uniform sampling algorithm of Bădoiu et al. [7] for 1-median stated above. In each iteration a single point is sampled uniformly. With high



probability moving the current center towards that point for a carefully chosen step size improves the solution. Each step can be seen as taking a descent towards a uniformly random direction from the subgradient which equals roughly the sum of directions to all points. Another example is the coresets construction for the smallest enclosing ball problem by Bădoiu and Clarkson [5], where the next point included in the coresets is the one maximizing the distance to the current best center. The direction taken towards that point is again related to the subgradient of the objective function at the current center position. The authors also gave a more explicit application of subgradient descent to the problem with a slightly larger number of iterations. More recently, Cohen et al. [10] developed one of the fastest  $(1 + \varepsilon)$ -approximation algorithms to date for the geometric median problem via stochastic subgradient methods.

**Kernel methods in machine learning.** Kernel functions simulate an inner product space in large or even unbounded dimensions but can be evaluated via simple low dimensional vector operations in the original dimension of input points [26]. This enables simple spherical shape fitting via a smallest enclosing ball algorithm in the high dimensional feature space, which implicitly defines a more complex and even non-convex shape in the original space. The smallest enclosing ball problem in kernel spaces is well-known as the support vector data description (SVDD) by Tax and Duin [28]. A more subtle connection between (the dual formulations of) several kernel based methods in machine learning and the smallest enclosing ball problem was established by Tsang et al. [30].

## 1.2 Our contributions and outline

- We extend in Section 2 the geometric median in Euclidean space to the more general set median problem. It consists of finding a center  $c \in \mathbb{R}^d$  that minimizes the sum of maximum distances to sets of points in a given collection of  $N$  point sets. We show how to solve this problem via estimation and sampling techniques combined with a stochastic subgradient descent algorithm.
- The elements in the collection are sets of up to  $n$  points in  $\mathbb{R}^d$ . We discuss in Section 2.1 the possibility of further reducing their size. In the previous work [22] they were summarized via strong coresets of size  $1/\varepsilon^{\Theta(d)}$  for constant dimension  $d$ . This is not an option in high dimensions where, e.g.  $d \approx n$ . Reviewing the techniques of Agarwal and Sharathkumar [2] we can show that no reduction below  $\min\{n, \exp(d^{1/3})\}$  is possible unless one is willing to sacrifice an additional approximation factor of roughly  $\sqrt{2}$ . However, we discuss the possibility to achieve roughly a factor  $(\sqrt{2} + \varepsilon)$ -approximation in streaming via the *blurred-ball-cover* [2] of size  $O(1/\varepsilon^3 \cdot \log 1/\varepsilon)$ , and in an off-line setting via weak coresets [5, 6] of size  $O(1/\varepsilon)$ .
- We show in Section 3.1 how this improves the previously best FPTAS for the probabilistic smallest enclosing ball problem from  $O(dn/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^{O(d)})$  to  $O(dn/\varepsilon^4 \cdot \log^2 1/\varepsilon)$ . In particular the dependence on the dimension  $d$  is reduced from exponential to linear and more notably occurs only in distance evaluations between points in  $d$ -dimensional Euclidean space, but not in the number of sampled points nor in the number of candidate centers to evaluate.
- This enables in Section 3.2 working in very high  $D$ -dimensional Hilbert spaces whose inner products and distances are given implicitly via positive semidefinite kernel functions. These functions can be evaluated in  $O(d)$  time although  $D$  is large or even unbounded depending on the kernel function. As an example we extend the well-known support vector data description (SVDD) method to the probabilistic case. SVDD is equivalent to the smallest enclosing ball problem in the implicit high-dimensional feature space.

Please find the missing proofs in the full version of this paper.

### 1.3 General notation

We denote the set of positive integers up to  $n \in \mathbb{N}$  by  $[n] = \{1, \dots, n\}$ . For any vectors  $x, y \in \mathbb{R}^d$  we denote their inner product by  $\langle x, y \rangle = x^T y = \sum_{i=1}^d x_i y_i$  and the Euclidean norm by  $\|x\| = \sqrt{\langle x, x \rangle} = (\sum_{i=1}^d x_i^2)^{1/2}$ . The Cauchy-Schwarz inequality (CSI) states that  $\langle x, y \rangle \leq \|x\| \|y\|$ . For any convex function  $f: \mathbb{R}^d \rightarrow \mathbb{R}$  we denote by  $\partial f(x) = \{g \in \mathbb{R}^d \mid \forall y \in \mathbb{R}^d: f(x) - f(y) \leq \langle g, x - y \rangle\}$  the subdifferential, i.e., the set of subgradients of  $f$  at  $x$ . For any event  $\mathcal{E}$  let the indicator function be  $\mathbb{1}_{\mathcal{E}} = 1$  if  $\mathcal{E}$  happens and 0 otherwise. We denote by  $B(c, r) = \{x \mid \|c - x\| \leq r\}$  the Euclidean ball centered at  $c \in \mathbb{R}^d$  with radius  $r \geq 0$ . We assume the error parameter satisfies  $0 < \varepsilon < 1/9$ . All our results hold with constant probability, say  $1/8$ , which can be amplified to arbitrary  $1 - \eta$ ,  $0 < \eta < 1$ , by running  $O(\log 1/\eta)$  independent repetitions and returning the minimum found.

## 2 A generalized median problem

The probabilistic smallest enclosing ball (pSEB) problem is to find a center that minimizes the expected maximum distance to points drawn from the input distributions (see Section 3.1 and Definition 13), and it can be reduced to two different types of 1-median problems [22]. One of them is defined on the set of all non-empty locations in  $\mathbb{R}^d$  where probabilistic points may appear, equipped with the Euclidean distance. The other is defined on the collection of all possible realizations of probabilistic point sets, and the distance measure between a center  $c \in \mathbb{R}^d$  and a realization  $P_i \subset \mathbb{R}^d$  is the maximum distance between the center and any of the realized points, i.e.,  $\max_{p \in P_i} \|c - p\|$ .

We begin our studies with a generalized median problem that we call the *set median problem* and covers both of these cases.

► **Definition 1** (set median problem). *Let  $\mathcal{P} = \{P_1, \dots, P_N\}$  be a family of finite non-empty sets where  $\forall i \in [N]: P_i \subset \mathbb{R}^d$  and  $n = \max\{|P_i| \mid i \in [N]\}$ . The set median problem on  $\mathcal{P}$  consists in finding a center  $c \in \mathbb{R}^d$  that minimizes the cost function*

$$f(c) = \sum_{i=1}^N m(c, P_i),$$

where  $m(c, P_i) = \max_{p \in P_i} \|c - p\|$ .

It was noted in [22] that the distance measure  $m$  between any two sets  $A, B \subset \mathbb{R}^d$  defined by the maximum distance  $m(A, B) = \max_{a \in A, b \in B} \|a - b\|$  is not a metric since for any non-singleton set  $C \subset \mathbb{R}^d$  it holds that  $m(C, C) > 0$ . We stress here that we consider only cases where, as in Definition 1, one of the sets  $A = \{c\}$  is a singleton, and  $B = P$  is an arbitrary non-empty set of points from  $\mathbb{R}^d$ . In order to directly apply results from the theory of metric spaces we thus simply define  $m(A, B) = 0$  whenever  $A = B$ .

► **Lemma 2.** *Let  $\mathcal{X}$  be the set of all finite non-empty subsets of  $\mathbb{R}^d$ . We define*

$$m(A, B) = \begin{cases} \max_{a \in A, b \in B} \|a - b\| & \text{if } A \neq B \\ 0 & \text{if } A = B \end{cases}$$

for any  $A, B \in \mathcal{X}$ . Then  $(\mathcal{X}, m)$  is a metric space.

Note that in case of singleton sets, the set median problem in Definition 1 is equivalent to the well-known Fermat-Weber problem also known as 1-median or geometric median. Also, for  $N = 1$  it coincides with the smallest enclosing ball or 1-center problem.

For both of these problems there are known algorithms based on the subgradient method from convex optimization. Bádoiu and Clarkson [5] gave a simple algorithm for approximating the 1-center within  $(1 + \varepsilon)$ -error. Starting from an initial center, it is iteratively moved a little towards the input point that is furthest away. Note that a suitable subgradient at the current center points exactly into the opposite direction. More precisely if  $q \in P$  is a point that is furthest away from the current center  $c$  then  $(c - q) / \|c - q\| \in \partial \max_{p \in P} \|c - p\|$ . The algorithm can thus be interpreted as a subgradient descent minimizing  $\max_{p \in P} \|c - p\|$ . Weiszfeld's algorithm [31, 32] was the first that solved the 1-median problem within additive  $O(\varepsilon)$ -error in  $O(1/\varepsilon)$  subgradient iterations. And indeed one of the fastest algorithms to approximate the geometric median problem to date relies on a stochastic subgradient descent [10]. The crucial ingredients to turn the additive error to a relative error are finding a suitable starting point that achieves a constant approximation and estimating its initial distance to the optimal solution. Another important step is a bound on the Lipschitz constant of the cost function, see also [3]. We will generalize these approaches to minimizing the cost function  $f$  of the set median problem from Definition 1.

First note that  $f$  is a convex function which implies that we can apply the theory of convex analysis and optimization. In particular, it implies that the subdifferential  $\partial f(c)$  is non-empty for any center  $c \in \mathbb{R}^d$  and  $c$  is locally optimal if and only if  $0 \in \partial f(c)$ . Moreover, any local optimum is also globally optimal by convexity [4, 23]. This implies that if we find a  $(1 + \varepsilon)$ -approximation to a local minimum of the convex function, the convexity implies that it is a  $(1 + \varepsilon)$ -approximation to the global minimum as well.

To see that  $f$  is convex, note that the Euclidean norm is a convex function. Therefore the Euclidean distance to some fixed point is a convex function since every translation of a convex function is convex. The maximum of convex functions is a convex function and finally the sum of convex functions is again convex. We prove this claim for completeness.

► **Lemma 3.** *The objective function  $f$  of the set median problem (see Definition 1) is convex.*

Next we bound the Lipschitz constant of the function  $f$  by  $N$ . We may get a better bound if we limit the domain of  $f$  to a ball of small radius centered at the optimal solution, but the Lipschitz constant cannot be bounded by  $o(N)$  in general. We will see later how we can remove the dependence on  $N$ .

► **Lemma 4.** *The objective function  $f$  of the set median problem (see Definition 1) is  $N$ -Lipschitz continuous, i.e.,  $|f(x) - f(y)| \leq N \cdot \|x - y\|$  for all  $x, y \in \mathbb{R}^d$ .*

We want to minimize  $f$  via the subgradient method, see [23]. For that sake we need to compute a subgradient  $g(c_i) \in \partial f(c_i)$  at the current center  $c_i$ . To this end we prove the following lemma.

► **Lemma 5.** *Let  $c_i \in \mathbb{R}^d$  be any center. For each set  $P_j \in \mathcal{P}$ , let  $p_j \in P_j$  be a point with  $\|c_i - p_j\| = m(c_i, P_j)$ . We have*

$$g(c_i) = \sum_{j=1}^N \frac{c_i - p_j}{\|c_i - p_j\|} \cdot \mathbf{1}_{c_i \neq p_j} \in \partial f(c_i),$$

*i.e.,  $g(c_i)$  is a valid subgradient of  $f$  at  $c_i$ .*

For brevity of presentation we omit the indicator function in any use of the above Lemma in the remainder of this paper and simply define  $(c_i - p_j) / \|c_i - p_j\| = 0$  whenever  $c_i = p_j$ .

The subgradient computation takes  $O(dnN)$  time to calculate, since in each of the  $N$  terms of the sum we maximize over  $|P_i| \leq n$  distances in  $d$  dimensions to find a point in  $P_i$  that is furthest away from  $c$ . We are going to discuss the possibility of reducing  $n$  later. For

now we focus on removing the dependence on  $N$ . To this end we would like to replace the exact subgradient  $g(c_i)$  by a uniform sample of only one nonzero term which points into the right direction in expectation. We formalize this in the following lemma.

► **Lemma 6.** *Let  $c_i \in \mathbb{R}^d$  be any fixed center. For each set  $P_j \in \mathcal{P}$ , let  $p_j \in P_j$  be a point with  $\|c_i - p_j\| = m(c_i, P_j)$ . Let  $\tilde{g}(c_i)$  be a random vector that takes the value  $\tilde{g}(c_i) = (c_i - p_j) / \|c_i - p_j\|$  for  $j \in [N]$  with probability  $1/N$  each. Then  $\mathbb{E}[\|\tilde{g}(c_i)\|^2] \leq 1$  and  $\mathbb{E}[\tilde{g}(c_i)] = g(c_i)/N$ , where  $g(c_i) \in \partial f(c_i)$  is the subgradient given in Lemma 5.*

We can now adapt the deterministic subgradient method from [23] using the random unbiased subgradient of Lemma 6 in such a way that the result is in expectation a  $(1 + \varepsilon)$ -approximation to the optimal solution. This method is presented in Algorithm 1. Given an initial center  $c_0$ , a fixed step size  $s$ , and a number of iterations  $\ell$ , the Algorithm iteratively picks a set  $P_j \in \mathcal{P}$  uniformly at random and chooses a point  $p_j \in P_j$  that attains the maximum distance to the current center. This point is used to compute an approximate subgradient via Lemma 6. The Algorithm finally outputs the best center found in all iterations.

---

**Algorithm 1:** Stochastic subgradient method.

---

**Data:** A family of non-empty sets  $\mathcal{P} = \{P_1, \dots, P_N\}$ , where  $P_i \subset \mathbb{R}^d$

**Result:** A center  $\tilde{c} \in \mathbb{R}^d$

- 1 Determine an initial center  $c_0$
  - 2 Fix a step size  $s$
  - 3 Fix the number of iterations  $\ell$
  - 4 **for**  $i \leftarrow 1$  **to**  $\ell$  **do**
  - 5     Choose an index  $j \in [N]$  uniformly at random and compute  $\tilde{g}(c_{i-1})$ , cf. Lemma 6
  - 6      $c_i = c_{i-1} - s \cdot \tilde{g}(c_{i-1})$
  - 7 **return**  $\tilde{c} \in \operatorname{argmin}_{c \in \{c_i | i=0, \dots, \ell\}} f(c)$
- 

The following theorem bounds in expectation the quality of the output that our subgradient algorithm returns. It is a probabilistic adaptation of a result in convex optimization [23].

► **Theorem 7.** *Consider Algorithm 1 on input  $\mathcal{P} = \{P_1, \dots, P_N\}$  for the set median problem with objective function  $f$ , see Definition 1. Let  $c^* \in \operatorname{argmin}_{c \in \mathbb{R}^d} f(c)$ . Let  $R = \|c_0 - c^*\|$ . Then*

$$\mathbb{E}_{\tilde{c}} [f(\tilde{c}) - f(c^*)] \leq N \cdot \frac{R^2 + (\ell + 1)s^2}{2(\ell + 1)s},$$

where the expectation is taken over the random variable  $\tilde{c} \in \operatorname{argmin}_{c \in \{c_i | i=0, \dots, \ell\}} f(c)$ , i.e., the output of Algorithm 1.

Our aim now is to choose the parameters  $\ell$ ,  $s$ , and  $c_0$  of Algorithm 1 in such a way that the bound given in Theorem 7 becomes at most  $\varepsilon f(c^*)$ . To this end we can choose the initial center  $c_0$  and bound its initial distance  $R = \|c_0 - c^*\|$  proportional to the average cost  $O(f(c^*)/N)$  with constant probability using a simple Markov argument.

► **Lemma 8.** *Choose a set  $P$  from  $\mathcal{P} = \{P_1, \dots, P_N\}$  uniformly at random and let  $c_0$  be an arbitrary point of  $P$ . Then for any constant  $0 < \delta_1 < 1$  it holds that  $R = \|c_0 - c^*\| \leq f(c^*)/(\delta_1 N)$  with probability at least  $1 - \delta_1$ .*

Assume that we know the value of  $R$ , and set the step size to  $s = R/\sqrt{\ell + 1}$ , then Theorem 7 and Lemma 8 imply that for some constant  $C$

$$\mathbb{E}_{\tilde{c}} [f(\tilde{c}) - f(c^*)] \leq \frac{NR}{\sqrt{\ell + 1}} \leq \frac{Cf(c^*)}{\sqrt{\ell + 1}}$$

holds with constant probability. We thus only need to run the algorithm for  $\ell \in O(1/\varepsilon^2)$  iterations to get within  $\varepsilon f(c^*)$  error. But choosing this particular step size requires to know the optimal center in advance. To get around this, we attempt to estimate the average cost. More formally, we are interested in a constant factor approximation of  $f(c^*)/N$ . It turns out that we can do this based on a small sample of the input sets unless our initial center is already a good approximation. But in the latter case we do not care about all the subsequent steps or step sizes, since we are already done after the initialization. The proof technique is originally from [19] and is adapted here to work in our setting with sets of points.

► **Lemma 9.** *There exists an algorithm that based on a sample  $\mathcal{S} \subseteq \mathcal{P}$  of size  $|\mathcal{S}| = 1/\varepsilon$  returns an estimate  $\tilde{R}$  and an initial center  $c_0$  in time  $O(dn/\varepsilon)$  such that with constant probability one of the following holds:*

- a)  $\varepsilon f(c^*)/N \leq \tilde{R} \leq (2/\varepsilon^3) \cdot f(c^*)/N$  and  $\|c_0 - c^*\| \leq 8f(c^*)/N$ ;
- b)  $\|c_0 - c^*\| \leq 4\varepsilon f(c^*)/N$ .

Lemma 9 has the following consequence. Either the initial center  $c_0$  is already a  $(1 + 4\varepsilon)$ -approximation, in which case we are done. Or we are close enough to an optimal solution and have a good estimate on the step size to find a  $(1 + 4\varepsilon)$ -approximation in a constant number of iterations.

Another issue that we need to take care of, is finding the best center in the last line of Algorithm 1 efficiently. We cannot do this exactly since evaluating the cost even for one single center takes time  $O(dnN)$ . However, we can find a point that is a  $(1 + \varepsilon)$ -approximation of the best center in a finite set of candidate centers using a result from the theory of discrete metric spaces.

To this end we can apply our next theorem which is originally due to Indyk and Thorup in [17, 29] and adapted here to work in our setting. The main difference is that in the original work the set of input points and the set of candidate solutions are identical. In our setting, however, we have that the collection of input sets and the set of candidate solutions may be completely distinct and the distance measure is the maximum distance (see Lemma 2).

► **Theorem 10.** *Let  $\mathcal{Q}$  be a set of uniform samples with repetition from  $\mathcal{P}$ . Let  $\mathcal{C}$  be a set of candidate solutions. Let  $a \in \mathcal{C}$  minimize  $\sum_{Q \in \mathcal{Q}} m(a, Q)$  and let  $\hat{c} = \operatorname{argmin}_{c \in \mathcal{C}} f(c)$ . Then*

$$\Pr \left[ \sum_{P \in \mathcal{P}} m(a, P) > (1 + \varepsilon) \sum_{P \in \mathcal{P}} m(\hat{c}, P) \right] \leq |\mathcal{C}| \cdot e^{-\varepsilon^2 |\mathcal{Q}|/64}.$$

Putting all pieces together we have the following Theorem.

► **Theorem 11.** *Consider an input  $\mathcal{P} = \{P_1, \dots, P_N\}$ , where for every  $i \in [N]$  we have  $P_i \subset \mathbb{R}^d$  and  $n = \max\{|P_i| \mid i \in [N]\}$ . There exists an algorithm that computes a center  $\tilde{c}$  that is with constant probability a  $(1 + \varepsilon)$ -approximation to the optimal solution  $c^*$  of the set median problem (see Definition 1). Its running time is  $O(dn/\varepsilon^4 \cdot \log^2 1/\varepsilon)$ .*

## 2.1 On reducing the size of the input sets

We would like to remove additionally the linear dependence on  $n$  for the maximum distance computations. This is motivated from the streaming extension of [22] where one aims at reading the input once in linear time and all subsequent computations should be sublinear,

or preferably independent of  $n$ . To this end a grid based strong coreset of size  $1/\varepsilon^{\Theta(d)}$  was used. However, here we focus on reducing the dependence on  $d$ , and exponential is not an option if we want to work in high dimensions. It turns out that without introducing an exponential dependence on  $d$ , we would have to lose a constant approximation factor. Pagh et al. [24] showed that if the coreset is a subset of the input and approximates furthest neighbor queries to within less than a factor of roughly  $\sqrt{2}$ , then it must consist of  $\Omega(\min\{n, \exp(d^{1/3})\})$  points. This was shown via a carefully constructed input point set of Agarwal and Sharathkumar [2] who used it to prove lower bounds on streaming algorithms for several extent problems.

In the next theorem, we review the techniques of the latter reference to show a slightly stronger result, namely no small data structure can exist for answering maximum distance queries to within a factor of less than roughly  $\sqrt{2}$ . In comparison to the previous results [2, 24], it is not limited to the streaming setting, and it is not restricted to subsets of the input.

► **Theorem 12.** *Any data structure that, with probability at least  $2/3$ ,  $\alpha$ -approximates maximum distance queries on a set  $S \subset \mathbb{R}^d$  of size  $|S| = n$ , for  $\alpha < \sqrt{2}(1 - 2/d^{1/3})$ , requires  $\Omega(\min\{n, \exp(d^{1/3})\})$  bits of storage.*

On the positive side it was shown by Goel et al. [13] that a  $\sqrt{2}$ -approximate furthest neighbor to the point  $c \in \mathbb{R}^d$  can always be found on the surface of the smallest enclosing ball of the sets  $P_i$ , using linear preprocessing time  $\tilde{O}(dn)$  and  $\tilde{O}(d^2)$  query time. Thus, if we plug in the coresets of Bădoiu and Clarkson [5, 6] of size  $O(1/\varepsilon)$  instead of the entire sets  $P_i \in \mathcal{P}$  to evaluate  $m(c, P_i)$ , we would have a sublinear time algorithm (in  $n$ ) after reading the input via a  $\sqrt{2}(1 + \varepsilon)$ -approximation to any query  $m(c, P_i)$ . In a streaming setting the same bound can be achieved via the *blurred-ball-cover* of Agarwal and Sharathkumar [2] of slightly larger size  $O(1/\varepsilon^3 \cdot \log 1/\varepsilon)$ . Otherwise, Goel et al. [13] have shown that using  $O(dn^{1+1/(1+\varepsilon)})$  preprocessing time and  $\tilde{O}(dn^{1/(1+\varepsilon)})$  query time, one can obtain a  $(1 + \varepsilon)$ -approximation for the furthest neighbor problem. Note that in this case the preprocessing time is already superlinear in  $n$ , and in particular the exponent is already larger than 1.7 for  $1 + \varepsilon < \sqrt{2}$ .

### 3 Applications

#### 3.1 Probabilistic smallest enclosing ball

We apply our result to the probabilistic smallest enclosing ball problem, as given in [22]. In such a setting, the input is a set  $\mathcal{D} = \{D_1, \dots, D_n\}$  of  $n$  discrete and independent probability distributions. The  $i$ -th distribution  $D_i$  is defined over a set of  $z$  possible locations  $q_{i,j} \in \mathbb{R}^d \cup \{\perp\}$ , for  $j \in [z]$ , where  $\perp$  indicates that the  $i$ -th point is not present in a sampled set, i.e.,  $q_{i,j} = \perp \Leftrightarrow \{q_{i,j}\} = \emptyset$ . We call these points *probabilistic points*. Each location  $q_{i,j}$  is associated with the probability  $p_{i,j}$ , such that  $\sum_{j=1}^z p_{i,j} = 1$ , for every  $i \in [n]$ . Thus the probabilistic points can be considered as independent random variables  $X_i$ .

A probabilistic set  $X$  consisting of probabilistic points is also a random variable, where for each random choice of indices  $(j_1, \dots, j_n) \in [z]^n$  there is a realization  $P_{(j_1, \dots, j_n)} = X(j_1, \dots, j_n) = (q_{1,j_1}, \dots, q_{n,j_n})$ . By independence of the distributions  $D_i$ ,  $i \in [n]$ , it holds that  $\Pr[X = P_{(j_1, \dots, j_n)}] = \prod_{i=1}^n p_{i,j_i}$ .

The probabilistic smallest enclosing ball problem is defined as follows. Here we may assume that the distance of any point  $c \in \mathbb{R}^d$  to the empty set is 0.

► **Definition 13** ([22]). Let  $\mathcal{D}$  be a set of  $n$  discrete distributions, where each distribution is defined over  $z$  locations in  $\mathbb{R}^d \cup \{\perp\}$ . The probabilistic smallest enclosing ball problem is to find a center  $c^* \in \mathbb{R}^d$  that minimizes the expected smallest enclosing ball cost, i.e.,

$$c^* \in \operatorname{argmin}_{c \in \mathbb{R}^d} \mathbb{E}_X [m(c, X)],$$

where the expectation is taken over the randomness of  $X \sim \mathcal{D}$ .

The authors of [22] showed a reduction of the probabilistic smallest enclosing ball problem to computing a solution for the set median problem of Definition 1. Their algorithm distinguishes between two cases. In the first case the probability of obtaining a nonempty realization  $P \neq \emptyset$  is small, more formally  $\sum_{q_{i,j} \in Q} p_{i,j} \leq \varepsilon$ , where  $Q = \{q_{i,j} \mid q_{i,j} \neq \perp, i \in [n], j \in [z]\}$ , and thus we have little chance of gaining information by sampling realizations. However, it was shown that

$$(1 - \varepsilon) \cdot \mathbb{E}_X \left[ \sum_{p \in X} \|c - p\| \right] \leq \mathbb{E}_X [m(c, X)] \leq \mathbb{E}_X \left[ \sum_{p \in X} \|c - p\| \right],$$

where  $\mathbb{E}_X \left[ \sum_{p \in X} \|c - p\| \right] = \sum_{i,j} p_{i,j} \cdot \|c - q_{i,j}\|$  is a weighted version of the deterministic 1-median problem, cf. [11], and thus also a weighted instance of the set median problem. In the second case, the probability that a realization contains at least one point is reasonably large. Therefore by definition of the expected value and  $m(c, \emptyset) = 0$  we have

$$\mathbb{E}_X [m(c, X)] = \sum_{P \neq \emptyset} \Pr[X = P] \cdot m(c, P),$$

which is a weighted version of the set median problem. Depending on these two cases, we sample a number of elements, non-empty locations or non-empty realizations, and solve the resulting set median problem using the samples in Theorem 11 for computing the approximate subgradients.

Algorithm 2 adapts this framework. It differs mainly in three points from the previous algorithm of [22]. First, the number of samples had a dependence on  $d$  hidden in the  $O$ -notation. This is not the case any more. Second, the sampled realizations are not sketched via coresets of size  $1/\varepsilon^{\Theta(d)}$  any more, as discussed in Section 2.1. Third, the running time of the actual optimization task is reduced via Theorem 11 instead of an exhaustive grid search.

► **Theorem 14.** Let  $\mathcal{D}$  be a set of  $n$  discrete distributions, where each distribution is defined over  $z$  locations in  $\mathbb{R}^d \cup \{\perp\}$ . Let  $\tilde{c} \in \mathbb{R}^d$  denote the output of Algorithm 2 on input  $\mathcal{D}$ , and let the approximation parameter be  $\varepsilon < 1/9$ . Then with constant probability the output is a  $(1 + \varepsilon)$ -approximation for the probabilistic smallest enclosing ball problem. I.e., it holds that

$$\mathbb{E}_X [m(\tilde{c}, X)] \leq (1 + \varepsilon) \min_{c \in \mathbb{R}^d} \mathbb{E}_X [m(c, X)].$$

The running time of Algorithm 2 is  $O(dn \cdot (z/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^4 \cdot \log^2 1/\varepsilon))$ .

Comparing to the result of [22], the running time is reduced from  $O(dnz/\varepsilon^{O(1)} + 1/\varepsilon^{O(d)})$  to  $O(dnz/\varepsilon^{O(1)})$ , i.e., our dependence on the dimension  $d$  is no longer exponential but only linear. Note, in particular, that the factor of  $d$  plays a role only in computations of distances between two points in  $\mathbb{R}^d$ . Further the sample size and the number of centers that need to be evaluated do not depend on the dimension  $d$  any more. This will be crucial in the next application.



---

**Algorithm 2:** Probabilistic smallest enclosing ball.

---

**Data:** A set  $\mathcal{D}$  of  $n$  point distributions over  $z$  locations in  $\mathbb{R}^d$ , a parameter  $\varepsilon < 1/9$

**Result:** A center  $\hat{c} \in \mathbb{R}^d$

- 1  $Q \leftarrow \{q_{i,j} \mid q_{i,j} \neq \perp, i \in [n], j \in [z]\}$       /\* the set of non-empty locations \*/
- 2 Set a sample size  $k \in O(1/\varepsilon^2 \log(1/\varepsilon))$
- 3 **if**  $\sum_{q_{i,j} \in Q} p_{i,j} \leq \varepsilon$  **then**
- 4     - Pick a random sample  $R$  of  $k$  locations from  $\mathcal{P} = Q$ , where for every  $r \in R$  we have  $r = q_{ij}$  with probability proportional to  $p_{ij}$
- 5     - Compute  $\hat{c} \in \mathbb{R}^d$  that is a  $(1 + \varepsilon)$ -approximation using the sampled points  $R$  one-by-one for computing the approximate subgradients in the algorithm of Theorem 11
- 6 **else**
- 7     - Sample a set  $R$  of  $k$  non-empty realizations from the input distributions  $\mathcal{D}$
- 8     - Compute  $\hat{c} \in \mathbb{R}^d$  that is a  $(1 + \varepsilon)$ -approximation using the sampled realizations  $R$  one-by-one for computing the approximate subgradients in the algorithm of Theorem 11
- 9 **return**  $\hat{c}$

---

### 3.2 Probabilistic support vector data description

Now we turn our attention to the support vector data description (SVDD) problem [28] and show how to extend it to its probabilistic version. To this end, let  $K: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  be a positive semidefinite kernel function. It is well known by Mercer's theorem, cf. [26], that such a function implicitly defines the inner product of a high dimensional Hilbert space  $\mathcal{H}$ , say  $\mathbb{R}^D$  where  $D \gg d$ . This means we have  $K(x, y) = \langle \varphi(x), \varphi(y) \rangle$ , where  $\varphi: \mathbb{R}^d \rightarrow \mathcal{H}$  is the so called feature mapping associated with the kernel. Examples for such kernel functions include polynomial transformations of the standard inner product in  $\mathbb{R}^d$  such as the constant, linear or higher order polynomial kernels. In these cases  $D$  remains bounded but grows as a function of  $d$  raised to the power of the polynomials' degree. Other examples are the exponential, squared exponential, Matérn, or rational quadratic kernels, which are transformations of the Euclidean distance between the two low dimensional vectors. The dimension  $D$  of their implicit feature space is in principle unbounded. Despite the large dimension  $D \gg d$ , all these kernels can be evaluated in time  $O(d)$  [25].

It is known that the SVDD problem is equivalent to the smallest enclosing ball problem in the feature space induced by the kernel function [30], i.e., given the kernel  $K$  with implicit feature mapping  $\varphi: \mathbb{R}^d \rightarrow \mathcal{H}$ , and an input set  $P \subseteq \mathbb{R}^d$ , the task is to find

$$c^* \in \operatorname{argmin}_{c \in \mathcal{H}} \max_{p \in P} \|c - \varphi(p)\| = \operatorname{argmin}_{c \in \mathcal{H}} m(c, \varphi(P)), \quad (1)$$

where  $\varphi(P) = \{\varphi(p) \mid p \in P\}$ .

Now we extend this to the probabilistic setting as we did in Section 3.1. The input is again a set  $\mathcal{D}$  of  $n$  discrete and independent probability distributions, where  $D_i \in \mathcal{D}$  is defined over a set of  $z$  locations  $q_{i,j} \in \mathbb{R}^d \cup \{\perp\}$ . Note that the mapping  $\varphi$  maps the locations  $q_{i,j}$  from  $\mathbb{R}^d$  to  $\varphi(q_{i,j})$  in  $\mathcal{H}$ , and we assume  $\varphi(\perp) = \perp$ . Then the probabilistic SVDD problem is given by the following adaptation of Definition 13.

► **Definition 15.** Let  $\mathcal{D}$  be a set of  $n$  discrete distributions, where each distribution is defined over  $z$  locations in  $\mathbb{R}^d \cup \{\perp\}$ . Let  $K: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$  be a kernel function with associated feature map  $\varphi: \mathbb{R}^d \rightarrow \mathcal{H}$ . The probabilistic support vector data description (pSVDD) problem is to find a center  $c^* \in \mathcal{H}$  that minimizes the expected SVDD cost, i.e.,

$$c^* \in \operatorname{argmin}_{c \in \mathcal{H}} \mathbb{E}_X [m(c, \varphi(X))],$$

where the expectation is taken over the randomness of  $X \sim \mathcal{D}$ .

Note that the deterministic problem, see Equation (1), is often stated with squared distances [28, 30]. It does not matter whether we minimize the maximum distance or any of its powers or any other monotone transformation. In the probabilistic case this is not true. Consider for instance squared distances. The resulting problem would be similar to a 1-means rather than a 1-median problem. Huang et al. [16] have observed that minimizing the expected maximum squared distance corresponds to minimizing the expected area of an enclosing ball in  $\mathbb{R}^2$ . This observation can be generalized to the expected volume of an enclosing ball in  $\mathbb{R}^p$  when the  $p$ -th powers of distances are considered. Considering  $p = 2$  might also have advantages when dealing with Gaussian input distributions due to their strong connection to squared Euclidean distances. In a general setting of the probabilistic smallest enclosing ball problem however, it is natural to minimize in expectation the maximum Euclidean distance, as in Definitions 13 and 15, since its radius is the primal variable to minimize.

Next we want to show how to find a  $(1 + \varepsilon)$ -approximation for the pSVDD problem. Explicitly computing any center  $c \in \mathcal{H}$  takes  $\Omega(D)$  time and space which is prohibitive not only when  $D = \infty$ . Note that for the SEB and SVDD problems, any reasonable center lies in the convex hull of the input points. Since taking the expectation is simply another linear combination over such centers, we can express any center  $c \in \mathcal{H}$  as a linear combination of the set of non-empty locations, i.e.,

$$c = \sum_{q_{u,v} \in Q} \gamma_{u,v} \varphi(q_{u,v})$$

The idea is to exploit this characterization to simulate Algorithm 1 and thereby Algorithm 2 to work in the feature space  $\mathcal{H}$  by computing the centers and distances only implicitly.

For now, assume that any distance computation can be determined. Note that sampling a set  $P_i \subset \mathbb{R}^d$  is the same as sampling the set  $\varphi(P_i)$  of corresponding points in  $\mathcal{H}$  from the same distribution. We assume that we have a set of locations or realizations  $\mathcal{P} = \{P_1, \dots, P_N\}$ , with  $P_i \subset \mathbb{R}^d$ . The remaining steps are passed to Theorem 11 which is based on Algorithm 1. First, we show the invariant that each center  $c_i$  reached during its calls to Algorithm 1 can be updated such that we maintain a linear combination  $c_i = \sum_{u,v} \gamma_{u,v} \varphi(q_{u,v})$ , where at most  $i + 1$  terms have  $\gamma_{u,v} \neq 0$ .

- The initial center  $c_0 \in \mathcal{H}$  is chosen by sampling uniformly at random a set  $P \in \mathcal{P}$  via Lemma 8. We take any point  $q \in P$ ,  $q \neq \perp$ , which maps to  $c_0 = \varphi(q)$ . Thus the invariant is satisfied at the beginning, where the corresponding coefficient is  $\gamma = 1$  and all other coefficients are zero.
- In each iteration we randomly sample a set  $P \in \mathcal{P}$  to simulate the approximate subgradient  $\tilde{g}(c_i)$  at the current point  $c_i$ . The vector  $\tilde{g}(c_i)$  is a vector between  $c_i$  and some point  $p_{j,k} = \varphi(q_{j,k}) \in \mathcal{H}$ , such that  $q_{j,k}$  maximizes  $\|c_i - \varphi(q')\|$  over all  $q' \in P$  (cf. Lemma 6). To implicitly update to the next center  $c_{i+1}$  note that (cf. Algorithm 1)

$$c_{i+1} = c_i - s \cdot \frac{c_i - \varphi(q_{j,k})}{\|c_i - \varphi(q_{j,k})\|} = \left(1 - \frac{s}{\|c_i - \varphi(q_{j,k})\|}\right) \cdot c_i + \frac{s}{\|c_i - \varphi(q_{j,k})\|} \cdot \varphi(q_{j,k}).$$

Assume the invariant was valid that  $c_i$  was represented as  $c_i = \sum_{u,v} \gamma_{u,v} \varphi(q_{u,v})$  with at most  $i + 1$  non-zero coefficients. Then it also holds for the point  $c_{i+1}$  since the previous non-zero coefficients of  $\varphi(q_{u,v})$  are multiplied by  $1 - s / \|c_i - \varphi(q_{j,k})\|$  and the newly added  $\varphi(q_{j,k})$  is assigned the coefficient  $s / \|c_i - \varphi(q_{j,k})\|$ . So there are at most  $i + 2$  non-zero coefficients.

Therefore, we do not have to store the points  $c_i$  explicitly while performing Algorithm 1. The implicit representation can be maintained via a list storing points that appear in the approximate subgradients and their corresponding non-zero coefficients.

To actually compute the coefficients, we need to be able to compute Euclidean distances as well as determine  $s$ . Using Lemma 9 we determined the step size  $s$  via an estimator  $\tilde{R} = \sum_{P \in \mathcal{S}} m(c_0, P)$  based on a small sample  $\mathcal{S}$ . In particular this requires distance computations again. To this end, we show how to compute  $\|c_i - \varphi(q)\|$  for any location  $q \in \mathbb{R}^d$ . Recall that the kernel function implicitly defines the inner product in  $\mathcal{H}$ . It thus holds that

$$\begin{aligned} \|c_i - \varphi(q)\|^2 &= \left\| \sum_{u,v} \gamma_{u,v} \varphi(q_{u,v}) - \varphi(q) \right\|^2 = \left\| \sum_{w=0}^i \gamma_w \varphi(q_w) - \varphi(q) \right\|^2 \\ &= \left\| \sum_{w=0}^i \gamma_w \varphi(q_w) \right\|^2 + \|\varphi(q)\|^2 - 2 \sum_{w=0}^i \gamma_w \langle \varphi(q_w), \varphi(q) \rangle \\ &= \sum_{w=0}^i \sum_{w'=0}^i \gamma_w \gamma_{w'} K(q_w, q_{w'}) + K(q, q) - 2 \sum_{w=0}^i \gamma_w K(q_w, q), \end{aligned} \quad (2)$$

where  $w, w' \in \{0, \dots, i\}$  index the locations  $q_w, q_{w'}$  with corresponding  $\gamma_w, \gamma_{w'} \neq 0$  in iteration  $i$ . Therefore, we have the following Theorem.

► **Theorem 16.** *Let  $\mathcal{D}$  be a set of  $n$  discrete distributions, where each distribution is defined over  $z$  locations in  $\mathbb{R}^d \cup \{\perp\}$ . There exists an algorithm that implicitly computes  $\tilde{c} \in \mathcal{H}$  that with constant probability is a  $(1 + \varepsilon)$ -approximation for the probabilistic support vector data description problem. I.e., it holds that*

$$\mathbb{E}_X [m(\tilde{c}, \varphi(X))] \leq (1 + \varepsilon) \min_{c \in \mathcal{H}} \mathbb{E}_X [m(c, \varphi(X))],$$

where the expectation is taken over the randomness of  $X \sim \mathcal{D}$ . The running time of the algorithm is  $O(dn \cdot (z/\varepsilon^3 \cdot \log 1/\varepsilon + 1/\varepsilon^8 \cdot \log^2 1/\varepsilon))$ .

## 4 Conclusion and open problems

We studied the set median problem in high dimensions that minimizes the sum of maximum distances to the furthest point in each input set. We presented a  $(1 + \varepsilon)$ -approximation algorithm whose running time is linear in  $d$  and independent of the number of input sets. We further discussed that in high dimensions the size of the input sets cannot be reduced sublinearly without losing a factor of roughly  $\sqrt{2}$ . Our work resolves an open problem of [22] and improves the previously best algorithm for the probabilistic smallest enclosing ball problem in high dimensions by reducing the dependence on  $d$  from exponential to linear. This enables running the algorithm in high dimensional Hilbert spaces induced by kernel functions which makes it more flexible and viable as a building block in machine learning and data analysis. As an example we transferred the kernel based SVDD problem of [28] to the probabilistic data setting. Our algorithms assume discrete input distributions. It would be interesting to extend them to various continuous distributions. The pSEB problem minimizes the expected maximum distance. When it comes to minimizing volumes of balls or

in the context of Gaussian distributions it might be interesting to study higher moments of this variable. This corresponds to a generalization of the set median problem to minimizing the sum of higher powers of maximum distances. Finally we hope that our methods may help to extend more shape fitting and machine learning problems to the probabilistic setting.

---

## References

- 1 Marcel R. Ackermann, Johannes Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms*, 6(4):59:1–59:26, 2010.
- 2 Pankaj K. Agarwal and R. Sharathkumar. Streaming Algorithms for Extent Problems in High Dimensions. *Algorithmica*, 72(1):83–98, 2015. doi:10.1007/s00453-013-9846-4.
- 3 Amir Beck and Shoham Sabach. Weiszfeld’s Method: Old and New Results. *Journal of Optimization Theory and Applications*, pages 1–40, 2014.
- 4 Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- 5 Mihai Bădoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 801–802, 2003.
- 6 Mihai Bădoiu and Kenneth L. Clarkson. Optimal core-sets for balls. *Computational Geometry*, 40(1):14–22, 2008.
- 7 Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th ACM Symposium on Theory of Computing, STOC*, pages 250–257, 2002.
- 8 M. T. Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.
- 9 Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed Computing by Mobile Robots: Gathering. *SIAM Journal on Computing*, 41(4):829–879, 2012. doi:10.1137/100796534.
- 10 Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the 48th ACM Symposium on Theory of Computing, STOC*, pages 9–21, 2016. doi:10.1145/2897518.2897647.
- 11 Graham Cormode and Andrew McGregor. Approximation algorithms for clustering uncertain data. In *Proceedings of the 27th ACM Symposium on Principles of Database Systems, PODS*, pages 191–200, 2008.
- 12 Pavlos S. Efrimidis. Weighted Random Sampling over Data Streams. In *Algorithms, Probability, Networks, and Games*, pages 183–195. Springer International, 2015. doi:10.1007/978-3-319-24024-4\_12.
- 13 Ashish Goel, Piotr Indyk, and Kasturi R. Varadarajan. Reductions among high dimensional proximity problems. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 769–778, 2001.
- 14 Sudipto Guha and Kamesh Munagala. Exceeding expectations and clustering uncertain data. In *Proceedings of the 28th ACM Symposium on Principles of Database Systems, PODS*, pages 269–278, 2009.
- 15 Lingxiao Huang and Jian Li. Stochastic  $k$ -Center and  $j$ -Flat-Center Problems. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 110–129, 2017. doi:10.1137/1.9781611974782.8.
- 16 Lingxiao Huang, Jian Li, Jeff M. Phillips, and Haitao Wang.  $\epsilon$ -Kernel Coresets for Stochastic Points. In *Proceedings of the 24th Annual European Symposium on Algorithms, ESA*, pages 50:1–50:18, 2016.
- 17 Piotr Indyk. *High-dimensional Computational Geometry*. PhD thesis, Stanford University, 2000.
- 18 Ilan Kremer, Noam Nisan, and Dana Ron. On Randomized One-Round Communication Complexity. *Computational Complexity*, 8(1):21–49, 1999. doi:10.1007/s000370050018.

- 19 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *Journal of the Association for Computing Machinery*, 57(2):5:1–5:32, 2010.
- 20 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- 21 Alexander Munteanu and Chris Schwiegelshohn. Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms. *Künstliche Intelligenz*, 32(1):37–53, 2018. doi:10.1007/s13218-017-0519-3.
- 22 Alexander Munteanu, Christian Sohler, and Dan Feldman. Smallest enclosing ball for probabilistic data. In *Proceedings of the 30th ACM Symposium on Computational Geometry, SoCG*, pages 214–223, 2014.
- 23 Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Applied Optimization. Springer, New York, 2004.
- 24 Rasmus Pagh, Francesco Silvestri, Johan Sivertsen, and Matthew Skala. Approximate furthest neighbor with application to annulus query. *Information Systems*, 64:152–162, 2017. doi:10.1016/j.is.2016.07.006.
- 25 Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. URL: <http://www.worldcat.org/oclc/61285753>.
- 26 Bernhard Schölkopf and Alexander Johannes Smola. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002. URL: <http://www.worldcat.org/oclc/48970254>.
- 27 Marco Stolpe, Kanishka Bhaduri, Kamalika Das, and Katharina Morik. Anomaly Detection in Vertically Partitioned Data by Distributed Core Vector Machines. In *Proceedings of Machine Learning and Knowledge Discovery in Databases, ECML/PKDD Part III*, pages 321–336, 2013. doi:10.1007/978-3-642-40994-3\_21.
- 28 David M. J. Tax and Robert P. W. Duin. Support Vector Data Description. *Machine Learning*, 54(1):45–66, 2004. doi:10.1023/B:MACH.0000008084.60811.49.
- 29 Mikkel Thorup. Quick  $k$ -Median,  $k$ -Center, and Facility Location for Sparse Graphs. *SIAM Journal on Computing*, 34(2):405–432, 2005.
- 30 Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core Vector Machines: Fast SVM Training on Very Large Data Sets. *Journal of Machine Learning Research*, 6:363–392, 2005.
- 31 Endre Weiszfeld. Sur le point pour lequel la somme des distances de  $n$  points donnés est minimum. *Tohoku Mathematical Journal*, 43(2):355–386, 1937.
- 32 Endre Weiszfeld. On the point for which the sum of the distances to  $n$  given points is minimum. *Annals of Operations Research*, 167:7–41, 2009. Translated from the French original and annotated by Frank Plastria.

# A Weighted Approach to the Maximum Cardinality Bipartite Matching Problem with Applications in Geometric Settings

Nathaniel Lahn

Virginia Tech, Blacksburg, VA, USA  
lahnn@vt.edu

Sharath Raghvendra

Virginia Tech, Blacksburg, VA, USA  
sharathr@vt.edu

---

## Abstract

---

We present a weighted approach to compute a maximum cardinality matching in an arbitrary bipartite graph. Our main result is a new algorithm that takes as input a weighted bipartite graph  $G(A \cup B, E)$  with edge weights of 0 or 1. Let  $w \leq n$  be an upper bound on the weight of any matching in  $G$ . Consider the subgraph induced by all the edges of  $G$  with a weight 0. Suppose every connected component in this subgraph has  $\mathcal{O}(r)$  vertices and  $\mathcal{O}(mr/n)$  edges. We present an algorithm to compute a maximum cardinality matching in  $G$  in  $\tilde{\mathcal{O}}(m(\sqrt{w} + \sqrt{r} + \frac{wr}{n}))$  time.<sup>1</sup>

When all the edge weights are 1 (symmetrically when all weights are 0), our algorithm will be identical to the well-known Hopcroft-Karp (HK) algorithm, which runs in  $\mathcal{O}(m\sqrt{n})$  time. However, if we can carefully assign weights of 0 and 1 on its edges such that both  $w$  and  $r$  are sub-linear in  $n$  and  $wr = \mathcal{O}(n^\gamma)$  for  $\gamma < 3/2$ , then we can compute maximum cardinality matching in  $G$  in  $o(m\sqrt{n})$  time. Using our algorithm, we obtain a new  $\tilde{\mathcal{O}}(n^{4/3}/\varepsilon^4)$  time algorithm to compute an  $\varepsilon$ -approximate bottleneck matching of  $A, B \subset \mathbb{R}^2$  and an  $\frac{1}{\varepsilon^{\mathcal{O}(d)}} n^{1+\frac{d-1}{2d-1}}$  poly log  $n$  time algorithm for computing  $\varepsilon$ -approximate bottleneck matching in  $d$ -dimensions. All previous algorithms take  $\Omega(n^{3/2})$  time. Given any graph  $G(A \cup B, E)$  that has an easily computable balanced vertex separator for every subgraph  $G'(V', E')$  of size  $|V'|^\delta$ , for  $\delta \in [1/2, 1)$ , we can apply our algorithm to compute a maximum matching in  $\tilde{\mathcal{O}}(mn^{\frac{\delta}{1+\delta}})$  time improving upon the  $\mathcal{O}(m\sqrt{n})$  time taken by the HK-Algorithm.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms; Theory of computation  $\rightarrow$  Graph algorithms analysis; Theory of computation  $\rightarrow$  Network flows

**Keywords and phrases** Bipartite matching, Bottleneck matching

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.48

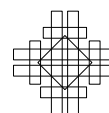
**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.10445>.

## 1 Introduction

We consider the classical matching problem in an arbitrary unweighted bipartite graph  $G(A \cup B, E)$  with  $|A| = |B| = n$  and  $E \subseteq A \times B$ . A *matching*  $M \subseteq E$  is a set of vertex-disjoint edges. We refer to a largest cardinality matching  $M$  in  $G$  as a *maximum matching*. A maximum matching is *perfect* if  $|M| = n$ . Now suppose the graph is weighted and every edge  $(a, b) \in E$  has a *weight* specified by  $c(a, b)$ . The weight of any subset of edges  $E' \subseteq E$  is given by  $\sum_{(a,b) \in E'} c(a, b)$ . A *minimum-weight maximum matching* is a maximum matching with the smallest weight. In this paper, we present an algorithm to compute a maximum matching faster by carefully assigning weights of 0 and 1 to the edges of  $G$ .

---

<sup>1</sup> We use  $\tilde{\mathcal{O}}$  to suppress poly-logarithmic terms.



**Maximum matching in graphs.** In an arbitrary bipartite graph with  $n$  vertices and  $m$  edges, Ford and Fulkerson's algorithm [7] iteratively computes, in each phase, an augmenting path in  $\mathcal{O}(m)$  time, leading to a maximum cardinality matching in  $\mathcal{O}(mn)$  time. Hopcroft and Karp's algorithm (HK-Algorithm) [10] reduces the number of phases from  $n$  to  $\mathcal{O}(\sqrt{n})$  by computing a maximal set of vertex-disjoint shortest augmenting paths in each phase. A single phase can be implemented in  $\mathcal{O}(m)$  time leading to an overall execution time of  $\mathcal{O}(m\sqrt{n})$ . In weighted bipartite graphs with  $n$  vertices and  $m$  edges, the well-known Hungarian method computes a minimum-weight maximum matching in  $\mathcal{O}(mn)$  time [11]. Gabow and Tarjan designed a weight-scaling algorithm (GT-Algorithm) to compute a minimum-weight perfect matching in  $\mathcal{O}(m\sqrt{n} \log(nC))$  time, provided all edge weights are integers bounded by  $C$  [8]. Their method, like the Hopcroft-Karp algorithm, computes a maximal set of vertex-disjoint shortest (for an appropriately defined augmenting path cost) augmenting paths in each phase. For the maximum matching problem in arbitrary graphs (not necessarily bipartite), a weighted approach has been applied to achieve a simple  $\mathcal{O}(m\sqrt{n})$  time algorithm [9].

Recently Lahn and Raghvendra [12] gave  $\tilde{\mathcal{O}}(n^{6/5})$  and  $\tilde{\mathcal{O}}(n^{7/5})$  time algorithms for finding a minimum-weight perfect bipartite matching in planar and  $K_h$ -minor<sup>2</sup> free graphs respectively, overcoming the  $\Omega(m\sqrt{n})$  barrier; see also Asathulla *et al.* [4]. Both these algorithms are based on the existence of an  $r$ -clustering which, for a parameter  $r > 0$ , is a partitioning of  $G$  into edge-disjoint clusters  $\{\mathcal{R}_1, \dots, \mathcal{R}_k\}$  such that  $k = \tilde{\mathcal{O}}(n/\sqrt{r})$ , every cluster  $\mathcal{R}_j$  has  $\mathcal{O}(r)$  vertices, and each cluster has  $\tilde{\mathcal{O}}(\sqrt{r})$  boundary vertices. A boundary vertex has edges from two or more clusters incident on it. Furthermore, the total number of boundary vertices, counted with multiplicity, is  $\tilde{\mathcal{O}}(n/\sqrt{r})$ . The algorithm of Lahn and Raghvendra extends to any graph that admits an  $r$ -clustering. There are also algebraic approaches for the design of fast algorithms for bipartite matching; see for instance [14, 15].

**Matching in geometric settings.** In geometric settings,  $A$  and  $B$  are points in a fixed  $d$ -dimensional space and  $G$  is a complete bipartite graph on  $A$  and  $B$ . For a fixed integer  $p \geq 1$ , the weight of an edge between  $a \in A$  and  $b \in B$  is  $\|a - b\|^p$ , where  $\|a - b\|$  denotes the Euclidean distance between  $a$  and  $b$ . The weight of a matching  $M$  is given by  $(\sum_{(a,b) \in M} \|a - b\|^p)^{1/p}$ . For any fixed  $p \geq 1$ , we wish to compute a perfect matching with the minimum weight. When  $p = 1$ , the problem is the well-studied *Euclidean bipartite matching* problem. A minimum-weight perfect matching for  $p = \infty$  will minimize the largest-weight edge in the matching and is referred to as a *bottleneck matching*. The Euclidean bipartite matching in a plane can be computed in  $\tilde{\mathcal{O}}(n^{3/2+\delta})$  [17] time for an arbitrary small  $\delta > 0$ ; see also Sharathkumar and Agarwal [19]. Efrat *et al.* present an algorithm to compute a bottleneck matching in the plane in  $\tilde{\mathcal{O}}(n^{3/2})$  [6] time. Both these algorithms use geometric data structures in a non-trivial fashion to speed up classical graph algorithms.

When  $p = 1$ , for any  $0 < \varepsilon \leq 1$ , there is an  $\varepsilon$ -approximation algorithm for the Euclidean bipartite matching problem that runs in  $\tilde{\mathcal{O}}(n/\varepsilon^d)$  time [18]. However, for  $p > 1$ , all known  $\varepsilon$ -approximation algorithms take  $\Omega(n^{3/2}/\varepsilon^d)$  time. We note that it is possible to find a  $\Theta(1)$ -approximate bottleneck matching in 2-dimensional space by reducing the problem to finding maximum flow in a planar graph and then finding the flow using an  $\tilde{\mathcal{O}}(n)$  time max-flow algorithm [5]. There are numerous other results; see also [2, 3, 16]. Designing exact and approximation algorithms that break the  $\Omega(n^{3/2})$  barrier remains an important research challenge in computational geometry.

---

<sup>2</sup> They assume  $h = O(1)$ .



**Our results.** We present a weighted approach to compute a maximum cardinality matching in an arbitrary bipartite graph. Our main result is a new matching algorithm that takes as input a weighted bipartite graph  $G(A \cup B, E)$  with every edge having a weight of 0 or 1. Let  $w \leq n$  be an upper bound on the weight of any matching in  $G$ . Consider the subgraph induced by all the edges of  $G$  with a weight 0. Let  $\{K_1, K_2, \dots, K_l\}$  be the connected components in this subgraph and let, for any  $1 \leq i \leq l$ ,  $V_i$  and  $E_i$  be the vertices and edges of  $K_i$ . We refer to each connected component  $K_i$  as a *piece*. Suppose  $|V_i| = \mathcal{O}(r)$  and  $|E_i| = \mathcal{O}(mr/n)$ . Given  $G$ , we present an algorithm to compute a maximum matching in  $G$  in  $\tilde{\mathcal{O}}(m(\sqrt{w} + \sqrt{r} + \frac{wr}{n}))$  time. Consider any graph in which removal of sub-linear number of “separator” vertices partitions the graph into connected components with  $\mathcal{O}(r)$  vertices and  $\mathcal{O}(mr/n)$  edges. We can apply our algorithm to any such graph by simply setting the weight of every edge incident on any separator vertex to 1 and weights of all other edges to 0.

When all the edge weights are 1 or all edge weights are 0, our algorithm will be identical to the HK-Algorithm algorithm and runs in  $\mathcal{O}(m\sqrt{n})$  time. However, if we can carefully assign weights of 0 and 1 on the edges such that both  $w$  and  $r$  are sub-linear in  $n$  and for some constant  $\gamma < 3/2$ ,  $wr = \mathcal{O}(n^\gamma)$ , then we can compute a maximum matching in  $G$  in  $o(m\sqrt{n})$  time. Using our algorithm, we obtain the following result for bottleneck matching:

- Given two point sets  $A, B \subset \mathbb{R}^2$  and an  $0 < \varepsilon \leq 1$ , we reduce the problem of computing an  $\varepsilon$ -approximate bottleneck matching to computing a maximum cardinality matching in a subgraph  $\mathcal{G}$  of the complete bipartite graph on  $A$  and  $B$ . We can, in  $\mathcal{O}(n)$  time assign 0/1 weights to the  $\mathcal{O}(n^2)$  edges of  $\mathcal{G}$  with so that any matching has a weight of  $\mathcal{O}(n^{2/3})$ . Despite possibly  $\Theta(n^2)$  edges in  $\mathcal{G}$ , we present an efficient implementation of our graph algorithm with  $\tilde{\mathcal{O}}(n^{4/3}/\varepsilon^4)$  execution time that computes an  $\varepsilon$ -approximate bottleneck matching for  $d = 2$ ; all previously known algorithms take  $\Omega(n^{3/2})$  time. Our algorithm, for any fixed  $d \geq 2$  dimensional space, computes an  $\varepsilon$ -approximate bottleneck matching in  $\frac{1}{\varepsilon^{\mathcal{O}(d)}} n^{1 + \frac{d-1}{2d-1}}$  poly log  $n$  time. (See Section 5).

The algorithm of Lahn and Raghvendra [12] for  $K_h$ -minor free graphs requires the clusters to have a small number of boundary vertices, which is used to create a compact representation of the residual network. This compact representation becomes prohibitively large as the number of boundary vertices increase. For instance, their algorithm has an execution time of  $\Omega(m\sqrt{n})$  for the case where  $G$  has a balanced vertex separator of  $\Theta(n^{2/3})$ . Our algorithm, on the other hand, extends to any graph with a sub-linear vertex separator. Given any graph  $G(A \cup B, E)$  that has an easily computable balanced vertex separator for every subgraph  $G'(V', E')$  of size  $|V'|^\delta$ , for  $\delta \in [1/2, 1)$ , there is a 0/1 weight assignment on edges of the graph so that the weight of any matching is  $\mathcal{O}(n^{\frac{2\delta}{1+\delta}})$  and  $r = \mathcal{O}(n^{\frac{1}{1+\delta}})$ . This assignment can be obtained by simply recursively sub-dividing the graph using balanced separators until each piece has  $\mathcal{O}(r)$  vertices and  $\mathcal{O}(mr/n)$  edges. All edges incident on the separator vertices are then assigned a weight of 1 and all other edges are assigned a weight of 0. As a result, we obtain an algorithm that computes the maximum cardinality matching in  $\tilde{\mathcal{O}}(mn^{\frac{\delta}{1+\delta}})$  time.

**Our approach.** Initially, we compute, in  $\mathcal{O}(m\sqrt{r})$  time, a maximum matching within all pieces. Similar to the GT-Algorithm, the rest of our algorithm is based on a primal-dual method and executes in phases. Each phase consists of two stages. The first stage conducts a Hungarian search and finds at least one augmenting path containing only zero slack (with respect to the dual constraints) edges. Let the admissible graph be the subgraph induced by the set of all zero slack edges. Unlike in the GT-Algorithm, the second stage of our algorithm computes augmenting paths in the admissible graph that are not necessarily vertex-disjoint. In the second stage, the algorithm iteratively initiates a DFS from every free vertex. When a

DFS finds an augmenting path  $P$ , the algorithm will augment the matching immediately and terminate this DFS. Let all pieces of the graph that contain the edges of  $P$  be *affected*. Unlike the GT-Algorithm, which deletes all edges visited by the DFS, our algorithm deletes only those edges that were visited by the DFS and did not belong to an affected piece. Consequently, we allow for visited edges from an affected piece to be reused in another augmenting path. As a result, our algorithm computes several more augmenting paths per phase than the GT-Algorithm, leading to a reduction of number of phases from  $\mathcal{O}(\sqrt{n})$  to  $\mathcal{O}(\sqrt{w})$ . Note, however, that the edges of an affected piece may now be visited multiple times by different DFS searches within the same phase. This increases the cumulative time taken by all the DFS searches in the second stage. However, we are able to bound the total number of affected pieces across all phases of the algorithm by  $\mathcal{O}(w \log w)$ . Since each piece has  $\mathcal{O}(mr/n)$  edges, the total time spent revisiting these edges is bounded by  $\mathcal{O}(mrw \log(w)/n)$ . The total execution time can therefore be bounded by  $\tilde{\mathcal{O}}(m(\sqrt{w} + \sqrt{r} + \frac{wr}{n}))$ .

## 2 Preliminaries

We are given a bipartite graph  $G(A \cup B, E)$ , where any edge  $(a, b) \in E$  has a weight  $c(a, b)$  of 0 or 1. Given a matching  $M$ , a vertex is *free* if it is not matched in  $M$ . An *alternating path* (resp. cycle) is a simple path (resp. cycle) that alternates between edges in  $M$  and not in  $M$ . An *augmenting path* is an alternating path that begins and ends at a free vertex.

A matching  $M$  and an assignment of dual weights  $y(\cdot)$  on the vertices of  $G$  is *feasible* if for any  $(a, b) \in A \times B$ :

$$y(b) - y(a) \leq c(a, b) \quad \text{if } (a, b) \notin M, \quad (1)$$

$$y(a) - y(b) = c(a, b) \quad \text{if } (a, b) \in M. \quad (2)$$

To assist in describing our algorithm, we first define a residual network and an augmented residual network with respect to a feasible matching  $M, y(\cdot)$ . A *residual network*  $G_M$  with respect to a feasible matching  $M$  is a directed graph where every edge  $(a, b)$  is directed from  $b$  to  $a$  if  $(a, b) \notin M$  and from  $a$  to  $b$  if  $(a, b) \in M$ . The weight  $s(a, b)$  of any edge is given by the slack of this edge with respect to feasibility conditions (1) and (2), i.e., if  $(a, b) \notin M$ , then  $s(a, b) = c(a, b) + y(a) - y(b)$  and  $s(a, b) = 0$  otherwise. An *augmented residual network* is obtained by adding to the residual network an additional vertex  $s$  and additional directed edges from  $s$  to every vertex in  $B_F$ , each of having a weight of 0. We denote the augmented residual network as  $G'_M$ .

## 3 Our algorithm

Throughout this section we will use  $M$  to denote the current matching maintained by the algorithm and  $A_F$  and  $B_F$  to denote the vertices of  $A$  and  $B$  that are free with respect to  $M$ . Initially  $M = \emptyset$ ,  $A_F = A$ , and  $B_F = B$ . Our algorithm consists of two steps. The first step, which we refer to as the *preprocessing step*, will execute the Hopcroft-Karp algorithm and compute a maximum matching within every piece. Any maximum matching  $M_{\text{OPT}}$  has at most  $w$  edges with a weight of 1 and the remaining edges have a weight of 0. Therefore,  $|M_{\text{OPT}}| - |M| \leq w$ . The time taken by the preprocessing step for  $K_i$  is  $\mathcal{O}(|E_i| \sqrt{|V_i|}) = \mathcal{O}(|E_i| \sqrt{r})$ . Since the pieces are vertex disjoint, the total time taken across all pieces is  $\mathcal{O}(m\sqrt{r})$ . After this step, no augmenting path with respect to  $M$  is completely contained within a single piece. We set the dual weight  $y(v)$  of every vertex  $v \in A \cup B$  to 0. The matching  $M$  along with the dual weights  $y(\cdot)$  satisfies (1) and (2) and is feasible.

The second step of the algorithm is executed in *phases*. We describe phase  $k$  of the algorithm. This phase consists of two stages.

**First stage.** In the first stage, we construct the augmented residual network  $G'_M$  and execute Dijkstra's algorithm with  $s$  as the source. Let  $\ell_v$  for any vertex  $v$  denote the shortest path distance from  $s$  to  $v$  in  $G'_M$ . If a vertex  $v$  is not reachable from  $s$ , we set  $\ell_v$  to  $\infty$ . Let

$$\ell = \min_{v \in A_F} \ell_v. \quad (3)$$

Suppose  $M$  is a perfect matching or  $\ell = \infty$ , then this algorithm returns with  $M$  as a maximum matching. Otherwise, we update the dual weight of any vertex  $v \in A \cup B$  as follows. If  $\ell_v \geq \ell$ , we leave its dual weight unchanged. Otherwise, if  $\ell_v < \ell$ , we set  $y(v) \leftarrow y(v) + \ell - \ell_v$ . After updating the dual weights, we construct the *admissible graph* which consists of a subset of edges in the residual network  $G_M$  that have zero slack. After the first stage, the matching  $M$  and the updated dual weights are feasible. Furthermore, there is at least one augmenting path in the admissible graph. This completes the first stage of the phase.

**Second stage:** In the second stage, we initialize  $G'$  to be the admissible graph and execute DFS to identify augmenting paths. For any augmenting path  $P$  found during the DFS, we refer to the pieces that contain its edges as *affected pieces* of  $P$ .

Similar to the HK-Algorithm, the second stage of this phase will initiate a DFS from every free vertex  $b \in B_F$  in  $G'$ . If the DFS does not lead to an augmenting path, we delete all edges that were visited by the DFS. On the other hand, if the DFS finds an augmenting path  $P$ , then the matching is augmented along  $P$ , all edges that are visited by the DFS and do not lie in an affected piece of  $P$  are deleted, and the DFS initiated at  $b$  will terminate.

Now, we describe in detail the DFS initiated for a free vertex  $b \in B_F$ . Initially  $P = \langle b = v_1 \rangle$ . Every edge of  $G'$  is marked unvisited. At any point during the execution of DFS, the algorithm maintains a simple path  $P = \langle b = v_1, v_2, \dots, v_k \rangle$ . The DFS search continues from the last vertex of this path as follows:

- If there are no unvisited edges that are going out of  $v_k$  in  $G'$ ,
  - If  $P = \langle v_1 \rangle$ , remove all edges that were marked as visited from  $G'$  and terminate the execution of DFS initiated at  $b$ .
  - Otherwise, delete  $v_k$  from  $P$  and continue the DFS search from  $v_{k-1}$ ,
- If there is an unvisited edge going out of  $v_k$ , let  $(v_k, v)$  be this edge. Mark  $(v_k, v)$  as visited. If  $v$  is on the path  $P$ , continue the DFS from  $v_k$ . If  $v$  is not on the path  $P$ , add  $(v_k, v)$  to  $P$ , set  $v_{k+1}$  to  $v$ , and,
  - Suppose  $v \in A_F$ , then  $P$  is an augmenting path from  $b$  to  $v$ . Execute the AUGMENT procedure which augments  $M$  along  $P$ . Delete from  $G'$  every visited edge that does not belong to any affected piece of  $P$  and terminate the execution of DFS initiated at  $b$ .
  - Otherwise,  $v \in (A \cup B) \setminus A_F$ . Continue the DFS from  $v_{k+1}$ .

The AUGMENT procedure receives a feasible matching  $M$ , a set of dual weights  $y(\cdot)$ , and an augmenting path  $P$  as input. For any  $(b, a) \in P \setminus M$ , where  $a \in A$  and  $b \in B$ , set  $y(b) \leftarrow y(b) - 2c(a, b)$ . Then augment  $M$  along  $P$  by setting  $M \leftarrow M \oplus P$ . By doing so, every edge of  $M$  after augmentation satisfies the feasibility condition (2). This completes the description of our algorithm. The algorithm maintains the following invariants during its execution:

- (11) The matching  $M$  and the set of dual weights  $y(\cdot)$  are feasible. Let  $y_{\max} = \max_{v \in B} y(v)$ . The dual weight of every vertex  $v \in B_F$  is  $y_{\max}$  and the dual weight for every vertex  $v \in A_F$  is 0.
- (12) For every phase that is fully executed prior to obtaining a maximum matching, at least one augmenting path is found and the dual weight of every free vertex of  $B_F$  increases by at least 1.

**Comparison with the GT-Algorithm.** In the GT-Algorithm, the admissible graph does not have any alternating cycles. Also, every augmenting path edge can be shown to not participate in any future augmenting paths that are computed in the current phase. By using these facts, one can show that the edges visited unsuccessfully by a DFS will not lead to an augmenting path in the current phase. In our case, however, admissible cycles can exist. Also, some edges on the augmenting path that have zero weight remain admissible after augmentation and may participate in another augmenting path in the current phase. We show, however, that any admissible cycle must be completely inside a piece and cannot span multiple pieces (Lemma 2). Using this fact, we show that edges visited unsuccessfully by the DFS that do not lie in an affected piece will not participate in any more augmenting paths (Lemma 6 and Lemma 7) in the current phase. Therefore, we can safely delete them.

**Correctness.** From Invariant (I2), each phase of our algorithm will increase the cardinality of  $M$  by at least 1 and so, our algorithm terminates with a maximum matching.

**Efficiency.** We use the following notations to bound the efficiency of our algorithm. Let  $\{P_1, \dots, P_t\}$  be the  $t$  augmenting paths computed in the second step of the algorithm. Let  $\mathbb{K}_i$  be the set of affected pieces with respect to the augmenting path  $P_i$ . Let  $M_0$  be the matching at the end of the first step of the algorithm. Let, for  $1 \leq i \leq t$ ,  $M_i = M_{i-1} \oplus P_i$ , i.e.,  $M_i$  is the matching after the  $i$ th augmentation in the second step of the algorithm.

The first stage is an execution of Dijkstra's algorithm which takes  $\mathcal{O}(m + n \log n)$  time. Suppose there are  $\lambda$  phases; then the cumulative time taken across all phases for the first stage is  $\mathcal{O}(\lambda m + \lambda n \log n)$ . In the second stage, each edge visited by a DFS is discarded for the remainder of the phase, provided it is not in an affected piece. Since each affected piece has  $\mathcal{O}(mr/n)$  edges, the total time taken by all the DFS searches across all the  $\lambda$  phases is bounded by  $\mathcal{O}((m + n \log n)\lambda + (mr/n) \sum_{i=1}^t |\mathbb{K}_i|)$ . In Lemma 3, we bound  $\lambda$  by  $\sqrt{w}$  and  $\sum_{i=1}^t |\mathbb{K}_i|$  by  $\mathcal{O}(w \log w)$ . Therefore, the total time taken by the algorithm including the time taken by preprocessing step is  $\mathcal{O}(m\sqrt{r} + m\sqrt{w} + n\sqrt{w} \log n + \frac{mrv \log w}{n}) = \tilde{\mathcal{O}}(m(\sqrt{w} + \sqrt{r} + \frac{wr}{n}))$ .

► **Lemma 1.** For any feasible matching  $M, y(\cdot)$  maintained by the algorithm, let  $y_{\max}$  be the dual weight of every vertex of  $B_F$ . For any augmenting path  $P$  with respect to  $M$  from a free vertex  $u \in B_F$  to a free vertex  $v \in A_F$ ,

$$c(P) = y_{\max} + \sum_{(a,b) \in P} s(a,b).$$

**Proof.** The weight of  $P$  is

$$c(P) = \sum_{(a,b) \in P} c(a,b) = \sum_{(a,b) \in P \setminus M} (y(b) - y(a) + s(a,b)) + \sum_{(a,b) \in P \cap M} (y(a) - y(b)).$$

Since every vertex on  $P$  except for  $u$  and  $v$  participates in one edge of  $P \cap M$  and one edge of  $P \setminus M$ , we can write the above equation as

$$c(P) = y(u) - y(v) + \sum_{(a,b) \in P \setminus M} s(a,b) = y(u) - y(v) + \sum_{(a,b) \in P} s(a,b). \quad (4)$$

The last equality follows from the fact that edges of  $P \cap M$  satisfy (2) and have a slack of zero. From (I1), we get that  $y(u) = y_{\max}$  and  $y(v) = 0$ , which gives,

$$c(P) = y_{\max} + \sum_{(a,b) \in P} s(a,b). \quad \blacktriangleleft$$

► **Lemma 2.** *For any feasible matching  $M, y(\cdot)$  maintained by the algorithm, and for any alternating cycle  $C$  with respect to  $M$ , if  $c(C) > 0$ , then*

$$\sum_{(a,b) \in P} s(a,b) > 0,$$

i.e.,  $C$  is not a cycle in the admissible graph.

**Proof.** The claim follows from (4) and the fact that the first vertex  $u$  and the last vertex  $v$  in a cycle are the same. ◀

► **Lemma 3.** *The total number of phases is  $\mathcal{O}(\sqrt{w})$  and the total number of affected pieces is  $\mathcal{O}(w \log w)$ , i.e.,  $\sum_{i=1}^t |\mathbb{K}_i| = \mathcal{O}(w \log w)$ .*

**Proof.** Let  $M_{\text{OPT}}$  be a maximum matching, which has weight at most  $w$ . Consider any phase  $k$  of the algorithm. By (I2), the dual weight  $y_{\max}$  of every free vertex in  $B_F$  is at least  $k$ . The symmetric difference of  $M$  and  $M_{\text{OPT}}$  will contain  $j = |M_{\text{OPT}}| - |M|$  vertex-disjoint augmenting paths. Let  $\{\mathcal{P}_1, \dots, \mathcal{P}_j\}$  be these augmenting paths. These paths contain edges of  $M_{\text{OPT}}$  and  $M$ , both of which are of weight at most  $w$ . Therefore, the sum of weights of these paths is

$$\sum_{i=1}^j c(\mathcal{P}_i) \leq 2w.$$

Let  $y_{\max}$  be the dual weight of every vertex  $b$  of  $B$  that is free with respect to  $M$ . i.e.,  $b \in B_F$ . From (I2),  $y_{\max} \geq k$ . From Lemma 1 and the fact that the slack on every edge is non-negative, we immediately get,

$$2w \geq \sum_{i=1}^j c(\mathcal{P}_i) \geq jy_{\max} \geq jk. \tag{5}$$

When  $\sqrt{w} \leq k < \sqrt{w} + 1$ , it follows from the above equation that  $j = |M_{\text{OPT}}| - |M| \leq 2\sqrt{w}$ . From (I2), we will compute at least one augmenting path in each phase and so the remaining  $j$  unmatched vertices are matched in at most  $2\sqrt{w}$  phases. This bounds the total number of phases by  $3\sqrt{w}$ .

Recollect that  $\{P_1, \dots, P_t\}$  are the augmenting paths computed by the algorithm. The matching  $M_0$  has  $|M_{\text{OPT}}| - t$  edges. Let  $y_{\max}^l$  correspond to the dual weight of the free vertices of  $B_F$  when the augmenting path  $P_l$  is found by the algorithm. From Lemma 1, and the fact that  $P_l$  is an augmenting path consisting of zero slack edges, we have  $y_{\max}^l = c(P_l)$ . Before augmenting along  $P_l$ , there are  $|M_{\text{OPT}}| - t + l - 1$  edges in  $M_{l-1}$  and  $j = |M_{\text{OPT}}| - |M_{l-1}| = t - l + 1$ . Plugging this in to (5), we get  $c(P_l) = y_{\max}^l \leq \frac{2w}{t-l+1}$ . Summing over all  $1 \leq l \leq t$ , we get,

$$\sum_{l=1}^t c(P_l) \leq w \sum_{l=1}^t \frac{2}{t-l+1} = \mathcal{O}(w \log t) = \mathcal{O}(w \log w). \tag{6}$$

For any augmenting path  $P_l$ , the number of affected pieces is upper bounded by the number of non-zero weight edges on  $P_l$ , i.e.,  $|\mathbb{K}_l| \leq c(P_l)$ . Therefore,

$$\sum_{l=1}^t |\mathbb{K}_l| \leq \sum_{l=1}^t c(P_l) = \mathcal{O}(w \log w). \quad \blacktriangleleft$$

#### 4 Proof of invariants

We now prove (I1) and (I2). Consider any phase  $k$  in the algorithm. Assume inductively that at the end of phase  $k - 1$ , (I1) and (I2) hold. We will show that (I1) and (I2) also hold at the end of the phase  $k$ .

► **Lemma 4.** *Any matching  $M$  and dual weights  $y(\cdot)$  maintained during the execution of the algorithm are feasible.*

Next we show that the dual weights  $A_F$  are zero and the dual weights of all vertices of  $B_F$  are equal to  $y_{\max}$ . At the start of the second step, all dual weights are 0. During the first stage, the dual weight of any vertex  $v$  will increase by  $\ell - \ell_v$  only if  $\ell_v < \ell$ . By (3), for every free vertex  $a \in A_F$ ,  $\ell_a \geq \ell$ , and so the dual weight of every free vertex of  $A$  remains unchanged at 0. Similarly, for any free vertex  $b \in B_F$ ,  $\ell_b = 0$ , and the dual weight increases by  $\ell$ , which is the largest possible increase. This implies that every free vertex in  $B_F$  will have the same dual weight of  $y_{\max}$ . In the second stage, matched vertices of  $B$  undergo a decrease in their dual weights, which does not affect vertices in  $B_F$ . Therefore, the dual weights of vertices of  $B_F$  will still have a dual weight of  $y_{\max}$  after stage two. This completes the proof of (I1).

Before we prove (I2), we will first establish a property of the admissible graph after the dual weight modifications in the first stage of the algorithm.

► **Lemma 5.** *After the first stage of each phase, there is an augmenting path consisting of admissible edges.*

**Proof of (I2).** From Lemma 5, there is an augmenting path of admissible edges at the end of the first stage of any phase. Since we execute a DFS from every free vertex  $b \in B_F$  in the second stage, we are guaranteed to find an augmenting path. Next, we show in Corollary 8 that there is no augmenting path of admissible edges at the end of stage two of phase  $k$ , i.e., all augmenting paths in the residual network have a slack of at least 1. This will immediately imply that the first stage of phase  $k + 1$  will have to increase the dual weight of every free vertex by at least 1 completing the proof for (I2).

Edges that are deleted during a phase do not participate in any augmenting path for the rest of the phase. We show this in two steps. First, we show that at the time of deletion of an edge  $(u, v)$ , there is no path in the admissible graph that starts from the edge  $(u, v)$  and ends at a free vertex  $a \in A_F$  (Lemma 7). In Lemma 6, we show that any such edge  $(u, v)$  will not participate in any admissible alternating path to a free vertex of  $A_F$  for the rest of the phase.

► **Lemma 6.** *Consider some point during the second stage of phase  $k$  where there is an edge  $(u, v)$  that does not participate in any admissible alternating path to a vertex of  $A_F$ . Then, for the remainder of phase  $k$ ,  $(u, v)$  does not participate in any admissible alternating path to a vertex of  $A_F$ .*



► **Lemma 7.** *Consider a DFS initiated from some free vertex  $b \in B_F$  in phase  $k$ . Let  $M$  be the matching at the start of this DFS and  $M'$  be the matching when the DFS terminates. Suppose the edge  $(u, v)$  was deleted during this DFS. Then there is no admissible path starting with  $(u, v)$  and ending at a free vertex  $a \in A_F$  in  $G_{M'}$ .*

By combining Lemmas 6 and 7, we get the following corollary.

► **Corollary 8.** *At the end of any phase, there is no augmenting path of admissible edges.*

## 5 Minimum bottleneck matching

We are given two sets  $A$  and  $B$  of  $n$   $d$ -dimensional points. Consider a weighted and complete bipartite graph on points of  $A$  and  $B$ . The weight of any edge  $(a, b) \in A \times B$  is given by its Euclidean distance and denoted by  $\|a - b\|$ . For any matching  $M$  of  $A$  and  $B$  let its largest weight edge be its *bottleneck edge*. In the *minimum bottleneck matching* problem, we wish to compute a matching  $M_{\text{OPT}}$  of  $A$  and  $B$  with the smallest weight bottleneck edge. We refer to this weight as the *bottleneck distance* of  $A$  and  $B$  and denote it by  $\beta^*$ . An  $\varepsilon$ -approximate *bottleneck matching* of  $A$  and  $B$  is any matching  $M$  with a bottleneck edge weight of at most  $(1 + \varepsilon)\beta^*$ . We present an algorithm that takes as input  $A, B$ , and a value  $\delta$  such that  $\beta^* \leq \delta \leq (1 + \varepsilon/3)\beta^*$ , and produces an  $\varepsilon$ -approximate bottleneck matching. For simplicity in presentation, we describe our algorithm for the 2-dimensional case when all points of  $A$  and  $B$  are in a bounding square  $S$ . The algorithm easily extends to any arbitrary fixed dimension  $d$ . For 2-dimensional case, given a value  $\delta$ , our algorithm executes in  $\tilde{O}(n^{4/3}/\varepsilon^3)$  time.

Although, the value of  $\delta$  is not known to the algorithm, we can first find a value  $\alpha$  that is guaranteed to be an  $n$ -approximation of the bottleneck distance [1, Lemma 2.2] and then select  $\mathcal{O}(\log n/\varepsilon)$  values from the interval  $[\alpha/n, \alpha]$  of the form  $(1 + \varepsilon/3)^i \alpha/n$ , for  $0 \leq i \leq \mathcal{O}(\log n/\varepsilon)$ . We will then execute our algorithm for each of these  $\mathcal{O}(\log n/\varepsilon)$  selected values of  $\delta$ . Our algorithm returns a maximum matching whose edges are of length at most  $(1 + \varepsilon/3)\delta$  in  $\mathcal{O}(n^{4/3}/\varepsilon^3)$  time. At least one of the  $\delta$  values chosen will be a  $\beta^* \leq \delta \leq (1 + \varepsilon/3)\beta^*$ . The matching returned by the algorithm for this value of  $\delta$  will be perfect ( $|M| = n$ ) and have a bottleneck edge of weight at most  $(1 + \varepsilon/3)^2 \beta^* \leq (1 + \varepsilon)\beta^*$  as desired. Among all executions of our algorithm that return a perfect matching, we return a perfect matching with the smallest bottleneck edge weight. Therefore, the total time taken to compute the  $\varepsilon$ -approximate bottleneck matching is  $\tilde{O}(n^{4/3}/\varepsilon^4)$ .

Given the value of  $\delta$ , the algorithm will construct a graph as follows: Let  $\mathbb{G}$  be a grid on the bounding square  $S$ . The side-length of every square in this grid is  $\varepsilon\delta/(6\sqrt{2})$ . For any cell  $\xi$  in the grid  $\mathbb{G}$ , let  $N(\xi)$  denote the subset of all cells  $\xi'$  of  $\mathbb{G}$  such that the minimum distance between  $\xi$  and  $\xi'$  is at most  $\delta$ . By the use of a simple packing argument, it can be shown that  $|N(\xi)| = \mathcal{O}(1/\varepsilon^2)$ .

For any point  $v \in A \cup B$ , let  $\xi_v$  be the cell of grid  $\mathbb{G}$  that contains  $v$ . We say that a cell  $\xi$  is *active* if  $(A \cup B) \cap \xi \neq \emptyset$ . Let  $A_\xi$  and  $B_\xi$  denote the points of  $A$  and  $B$  in the cell  $\xi$ . We construct a bipartite graph  $\mathcal{G}(A \cup B, \mathcal{E})$  on the points in  $A \cup B$  as follows: For any pair of points  $(a, b) \in A \times B$ , we add an edge in the graph if  $\xi_b \in N(\xi_a)$ . Note that every edge  $(a, b)$  with  $\|a - b\| \leq \delta$  will be included in  $\mathcal{G}$ . Since  $\delta$  is at least the bottleneck distance,  $\mathcal{G}$  will have a perfect matching. The maximum distance between any cell  $\xi$  and a cell in  $N(\xi)$  is  $(1 + \varepsilon/3)\delta$ . Therefore, no edge in  $\mathcal{G}$  will have a length greater than  $(1 + \varepsilon/3)\delta$ . This implies that any perfect matching in  $\mathcal{G}$  will also be an  $\varepsilon$ -approximate bottleneck matching. We use our algorithm for maximum matching to compute this perfect matching in  $\mathcal{G}$ . Note, that  $\mathcal{G}$  can have  $\Omega(n^2)$  edges. For the sake of efficiency, our algorithm executes on a compact representation of  $\mathcal{G}$  that is described later. Next, we assign weights of 0 and 1 to the edges of  $\mathcal{G}$  so that the any maximum matching in  $\mathcal{G}$  has a small weight  $w$ .



## 48:10 Weighted Approach to Maximum Cardinality Bipartite Matching

For a parameter<sup>3</sup>  $r > 1$ , we will carefully select another grid  $\mathbb{G}'$  on the bounding square  $S$ , each cell of which has a side-length of  $\sqrt{r}(\varepsilon\delta/(6\sqrt{2}))$  and encloses  $\sqrt{r} \times \sqrt{r}$  cells of  $\mathbb{G}$ . For any cell  $\xi$  of the grid  $\mathbb{G}$ , let  $\square_\xi$  be the cell in  $\mathbb{G}'$  that contains  $\xi$ . Any cell  $\xi$  of  $\mathbb{G}$  is a *boundary cell* with respect to  $\mathbb{G}'$  if there is a cell  $\xi' \in N(\xi)$  such that  $\square_{\xi'} \neq \square_\xi$ . Equivalently, if the minimum distance from  $\xi$  to  $\square_\xi$  is at most  $\delta$ , then  $\xi$  is a boundary cell. For any boundary cell  $\xi$  of  $\mathbb{G}$  with respect to grid  $\mathbb{G}'$ , we refer to all points of  $A_\xi$  and  $B_\xi$  that lie in  $\xi$  as boundary points. All other points of  $A$  and  $B$  are referred to as internal points. We carefully construct this grid  $\mathbb{G}'$  such that the total number of boundary points is  $\mathcal{O}(n/\varepsilon\sqrt{r})$  as follows: First, we will generate the vertical lines for  $\mathbb{G}'$ , and then we will generate the horizontal lines using a similar construction. Consider the vertical line  $y_{ij}$  to be the line  $x = i(\varepsilon\delta)/(6\sqrt{2}) + j\sqrt{r}(\varepsilon\delta/(6\sqrt{2}))$ . For any fixed integer  $i$  in  $[1, \sqrt{r}]$ , consider the set of vertical lines  $\mathbb{Y}_i = \{y_{ij} \mid y_{ij} \text{ intersects the bounding square } S\}$ . We label all cells  $\xi$  of  $\mathbb{G}$  as boundary cells with respect to  $\mathbb{Y}_i$  if the distance from  $\xi$  to some vertical line in  $\mathbb{Y}_i$  is at most  $\delta$ . We designate the points inside the boundary cells as boundary vertices with respect to  $\mathbb{Y}_i$ . For any given  $i$ , let  $A_i$  and  $B_i$  be the boundary vertices of  $A$  and  $B$  with respect to the lines in  $\mathbb{Y}_i$ . We select an integer  $\kappa = \arg \min_{1 \leq i \leq \sqrt{r}} |A_i \cup B_i|$  and use  $\mathbb{Y}_\kappa$  as the vertical lines for our grid  $\mathbb{G}'$ . We use a symmetric construction for the horizontal lines.

► **Lemma 9.** *Let  $A_i$  and  $B_i$  be the boundary points with respect to the vertical lines  $\mathbb{Y}_i$ . Let  $\kappa = \arg \min_{1 \leq i \leq \sqrt{r}} |A_i \cup B_i|$ . Then,  $|A_\kappa \cup B_\kappa| = \mathcal{O}(n/(\varepsilon\sqrt{r}))$ .*

**Proof.** For any fixed cell  $\xi$  in  $\mathbb{G}$ , of the  $\sqrt{r}$  values of  $i$ , there are  $\mathcal{O}(1/\varepsilon)$  values for which  $\mathbb{Y}_i$  has a vertical line at a distance at most  $\delta$  from  $\xi$ . Therefore, each cell  $\xi$  will be a boundary cell in only  $\mathcal{O}(1/\varepsilon)$  shifts out of  $\sqrt{r}$  shifts. So,  $A_\xi$  and  $B_\xi$  will be counted in  $A_i \cup B_i$  for  $\mathcal{O}(1/\varepsilon)$  different values of  $i$ . Therefore, if we take the average over choices of  $i$ , we get

$$\min_{1 \leq i \leq \sqrt{r}} |A_i \cup B_i| \leq \frac{1}{\sqrt{r}} \sum_{i=1}^{\sqrt{r}} |A_i \cup B_i| \leq \mathcal{O}(n/(\varepsilon\sqrt{r})). \quad \blacktriangleleft$$

Using a similar construction, we guarantee that the boundary points with respect to the horizontal lines of  $\mathbb{G}'$  is also at most  $\mathcal{O}(n/(\varepsilon\sqrt{r}))$ .

► **Corollary 10.** *The grid  $\mathbb{G}'$  that we construct has  $\mathcal{O}(n/(\varepsilon\sqrt{r}))$  many boundary points.*

For any two cells  $\xi$  and  $\xi' \in N(\xi)$  of the grid  $\mathbb{G}$ , suppose  $\square_\xi \neq \square_{\xi'}$ . Then the weights of all edges of  $A_\xi \times B_{\xi'}$  and of  $B_\xi \times A_{\xi'}$  are set to 1. All other edges have a weight of 0. We do not make an explicit weight assignment as it is expensive to do so. Instead, we can always derive the weight of an edge when we access it. Only boundary points will have edges of weight 1 incident on them. From Corollary 10, it follows that any maximum matching will have a weight of  $w = \mathcal{O}(n/(\varepsilon\sqrt{r}))$ .

The edges of every piece in  $\mathcal{G}$  have endpoints that are completely inside a cell of  $\mathbb{G}'$ . Note, however, that there is no straight-forward bound on the number of points and edges of  $\mathcal{G}$  inside each piece. Moreover, the number of edges in  $\mathcal{G}$  can be  $\Theta(n^2)$ . Consider any feasible matching  $M, y(\cdot)$  in  $\mathcal{G}$ . Let  $\mathcal{G}_M$  be the residual network. In order to obtain a running time of  $\tilde{\mathcal{O}}(n^{4/3}/\varepsilon^3)$ , we use the grid  $\mathbb{G}$  to construct a compact residual network  $\mathcal{CG}_M$  for any feasible matching  $M, y(\cdot)$  and use this compact graph to implement our algorithm. The following lemma assists us in constructing the compressed residual network.

<sup>3</sup> Assume  $r$  to be a perfect square.

► **Lemma 11.** Consider any feasible matching  $M, y(\cdot)$  maintained by our algorithm on  $\mathcal{G}$  and any active cell  $\xi$  in the grid  $\mathbb{G}$ . The dual weight of any two points  $a, a' \in A_\xi$  can differ by at most 2. Similarly, the dual weights of any two points  $b, b' \in B_\xi$  can differ by at most 2.

**Proof.** We present our proof for two points  $b, b' \in B_\xi$ . A similar argument will extend for  $a, a' \in A_\xi$ . For the sake of contradiction, let  $y(b) \geq y(b') + 3$ .  $b'$  must be matched since  $y(b') < y(b) \leq y_{\max}$ . Let  $m(b') \in A$  be the match of  $b'$  in  $M$ . From (2),  $y(m(b')) - y(b') = c(b', m(b'))$ . Since both  $b$  and  $b'$  are in  $\xi$ , the distance  $c(b, m(b')) = c(b', m(b'))$ . So,  $y(b) - y(m(b')) \geq (y(b') + 3) - y(m(b')) = 3 - c(b, m(b'))$ . This violates (1) leading to a contradiction. ◀

For any feasible matching and any cell  $\xi$  of  $\mathbb{G}$ , we divide points of  $A_\xi$  and  $B_\xi$  based on their dual weight into at most three clusters. Let  $A_\xi^1, A_\xi^2$  and  $A_\xi^3$  be the three clusters of points in  $A_\xi$  and let  $B_\xi^1, B_\xi^2$  and  $B_\xi^3$  be the three clusters of points in  $B_\xi$ . We assume that points with the largest dual weights are in  $A_\xi^1$  (resp.  $B_\xi^1$ ), the points with the second largest dual weights are in  $A_\xi^2$  (resp.  $B_\xi^2$ ), and the points with the smallest dual weights are in  $A_\xi^3$  (resp.  $B_\xi^3$ ).

**Compact residual network.** Given a feasible matching  $M$ , we construct a compact residual network  $\mathcal{CG}_M$  to assist in the fast implementation of our algorithm. This vertex set  $\mathcal{A} \cup \mathcal{B}$  for the compact residual network is constructed as follows. First we describe the vertex set  $\mathcal{A}$ . For every active cell  $\xi$  in  $\mathbb{G}$ , we add a vertex  $a_\xi^1$  (resp.  $a_\xi^2, a_\xi^3$ ) to represent the set  $A_\xi^1$  (resp.  $A_\xi^2, A_\xi^3$ ) provided  $A_\xi^1 \neq \emptyset$  (resp.  $A_\xi^2 \neq \emptyset, A_\xi^3 \neq \emptyset$ ). We designate  $a_\xi^1$  (resp.  $a_\xi^2, a_\xi^3$ ) as a free vertex if  $A_\xi^1 \cap A_F \neq \emptyset$  (resp.  $A_\xi^2 \cap A_F \neq \emptyset, A_\xi^3 \cap A_F \neq \emptyset$ ). Similarly, we construct a vertex set  $\mathcal{B}$  by adding a vertex  $b_\xi^1$  (resp.  $b_\xi^2, b_\xi^3$ ) to represent the set  $B_\xi^1$  (resp.  $B_\xi^2, B_\xi^3$ ) provided  $B_\xi^1 \neq \emptyset$  (resp.  $B_\xi^2 \neq \emptyset, B_\xi^3 \neq \emptyset$ ). We designate  $b_\xi^1$  (resp.  $b_\xi^2, b_\xi^3$ ) as a free vertex if  $B_\xi^1 \cap B_F \neq \emptyset$  (resp.  $B_\xi^2 \cap B_F \neq \emptyset, B_\xi^3 \cap B_F \neq \emptyset$ ). Each active cell  $\xi$  of the grid  $\mathbb{G}$  therefore has at most six points. Each point in  $\mathcal{A} \cup \mathcal{B}$  will inherit the dual weights of the points in its cluster; for any vertex  $a_\xi^1 \in \mathcal{A}$  (resp.  $a_\xi^2 \in \mathcal{A}, a_\xi^3 \in \mathcal{A}$ ), let  $y(a_\xi^1)$  (resp.  $y(a_\xi^2), y(a_\xi^3)$ ) be the dual weight of all points in  $A_\xi^1$  (resp.  $A_\xi^2, A_\xi^3$ ). We define  $y(b_\xi^1), y(b_\xi^2)$ , and  $y(b_\xi^3)$  as dual weights of points in  $B_\xi^1, B_\xi^2$ , and  $B_\xi^3$  respectively. Since there are at most  $n$  active cells,  $|\mathcal{A} \cup \mathcal{B}| = \mathcal{O}(n)$ .

Next, we create the edge set for the compact residual network  $\mathcal{CG}$ . For any active cell  $\xi$  in the grid  $\mathbb{G}$  and for any cell  $\xi' \in N(\xi)$ ,

- We add a directed edge from  $a_\xi^i$  to  $b_{\xi'}^j$ , for  $i, j \in \{1, 2, 3\}$  if there is an edge  $(a, b) \in (A_\xi^i \times B_{\xi'}^j) \cap M$ . We define the weight of  $(a_\xi^i, b_{\xi'}^j)$  to be  $c(a, b)$ . We also define the slack  $s(a_\xi^i, b_{\xi'}^j)$  to be  $c(a_\xi^i, b_{\xi'}^j) - y(a_\xi^i) + y(b_{\xi'}^j)$  which is equal to  $s(a_\xi^i, b_{\xi'}^j) = c(a, b) - y(a) + y(b) = s(a, b) = 0$ .
- We add a directed edge from  $b_\xi^i$  to  $a_{\xi'}^j$ , for  $i, j \in \{1, 2, 3\}$  if  $(B_\xi^i \times A_{\xi'}^j) \setminus M \neq \emptyset$ . Note that the weight and slack of every directed edge in  $B_\xi^i \times A_{\xi'}^j$  are identical. We define the weight of  $(b_\xi^i, a_{\xi'}^j)$  to be  $c(a, b)$  for any  $(a, b) \in A_{\xi'}^j \times B_\xi^i$ . We also define the slack  $s(b_\xi^i, a_{\xi'}^j) = c(b_\xi^i, a_{\xi'}^j) - y(b_\xi^i) + y(a_{\xi'}^j)$  which is equal to the slack  $s(a, b)$ .

For each vertex in  $\mathcal{A} \cup \mathcal{B}$ , we added at most two edges to every cell  $\xi' \in N(\xi)$ . Since  $N(\xi) = \mathcal{O}(1/\varepsilon^2)$ , the total number of edges in  $\mathcal{E}$  is  $\mathcal{O}(n/\varepsilon^2)$ . For a cell  $\square$  in  $\mathbb{G}'$ , let  $\mathcal{A}_\square$  be the points of  $\mathcal{A}$  generated by cells of  $\mathbb{G}$  that are contained inside the cell  $\square$ . A piece  $K_\square$  has  $\mathcal{A}_\square \cup \mathcal{B}_\square$  as the vertex set and  $\mathcal{E}_\square = ((\mathcal{A}_\square \times \mathcal{B}_\square) \cup (\mathcal{B}_\square \times \mathcal{A}_\square) \cap \mathcal{E})$  as the edge set. Note that the number of vertices in any piece  $K_\square$  is  $\mathcal{O}(r)$  and the number of edges in  $K_\square$  is  $\mathcal{O}(r/\varepsilon^2)$ . Every edge  $(u, v)$  of any piece  $K_\square$  has a weight  $c(u, v) = 0$  and every edge  $(u, v)$  with a weight of zero belongs to some piece of  $\mathcal{CG}$ .

The following lemma shows that the compact graph  $\mathcal{CG}$  preserves all minimum slack paths in  $\mathcal{G}_M$ .

► **Lemma 12.** *For any directed path  $\mathcal{P}$  in the compact residual network  $\mathcal{CG}$ , there is a directed path  $P$  in the residual network such that  $\sum_{(u,v) \in P} s(u,v) = \sum_{(u,v) \in \mathcal{P}} s(u,v)$ . For any directed path  $P$  in  $\mathcal{G}_M$ , there is a directed path  $\mathcal{P}$  in the compact residual network such that  $\sum_{(u,v) \in P} s(u,v) \geq \sum_{(u,v) \in \mathcal{P}} s(u,v)$ .*

**Preprocessing step.** At the start,  $M = \emptyset$  and all dual weights are 0. Consider any cell  $\square$  of the grid  $\mathbb{G}'$  and any cell  $\xi$  of  $\mathbb{G}$  that is contained inside  $\square$ . Suppose we have a point  $a_\xi^1$ . We assign a demand  $d_{a_\xi^1} = |A_\xi^1| = |A_\xi|$  to  $a_\xi^1$ . Similarly, suppose we have a point  $b_\xi^1$ , we assign a supply  $s_{b_\xi^1} = |B_\xi^1| = |B_\xi|$ . The preprocessing step reduces to finding a maximum matching of supplies to demand. This is an instance of the unweighted transportation problem which can be solved using the algorithm of [13] in  $\tilde{O}(|\mathcal{E}_\square| \sqrt{|\mathcal{A}_\square \cup \mathcal{B}_\square|}) = \tilde{O}(|\mathcal{E}_\square| \sqrt{r})$ . Every edge of  $\mathcal{E}$  participates in at most one piece. Therefore, the total time taken for preprocessing across all pieces is  $\tilde{O}(|\mathcal{E}| \sqrt{r}) = \tilde{O}(n\sqrt{r}/\varepsilon^2)$ . We can trivially convert the matching of supplies to demand to a matching in  $\mathcal{G}$ .

**Efficient implementation of the second step.** Recollect that the second step of the algorithm consists of phases. Each phase has two stages. In the first stage, we execute Dijkstra's algorithm in  $\mathcal{O}(n \log n / \varepsilon^2)$  time by using the compact residual network  $\mathcal{CG}$ . After adjusting the dual weight of nodes in the compact graph, in the second stage, we iteratively compute augmenting paths of admissible edges by conducting a DFS from each vertex. Our implementation of DFS has the following differences from the one described in Section 3.

- Recollect that each free vertex  $v \in \mathcal{B}$  may represent a cluster that has  $t > 0$  free vertices. We will execute DFS from  $v$  exactly  $t$  times, once for each of the free vertices of  $\mathcal{B}$ .
- During the execution of any DFS, unlike the algorithm described in Section 3, the DFS will mark an edge as visited only when it backtracks from the edge. Due to this change, all edges on the path maintained by the DFS are marked as unvisited. Therefore, unlike the algorithm from Section 3, this algorithm will not discard weight 1 edges of an augmenting path after augmentation. From Lemma 3, the total number of these edges is  $\mathcal{O}(w \log w)$ .

**Efficiency.** The first stage is an execution of Dijkstra's algorithm which takes  $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}| \log |\mathcal{V}|) = \mathcal{O}(n \log n / \varepsilon^2)$  time. Suppose there are  $\lambda$  phases; then the cumulative time taken across all phases for the first stage is  $\tilde{O}(\lambda n / \varepsilon^2)$ . In the second stage of the algorithm, in each phase, every edge is discarded once it is visited by a DFS, unless it is in an affected piece or it is an edge of weight 1 on an augmenting path. Since each affected piece has  $\mathcal{O}(r / \varepsilon^2)$  edges, and since there are  $\mathcal{O}(w \log w)$  edges of weight 1 on the computed augmenting paths, the total time taken by all the DFS searches across all the  $\lambda$  phases is bounded by  $\tilde{O}(n\lambda / \varepsilon^2 + r / \varepsilon^2 \sum_{i=1}^t |\mathbb{K}_i| + w \log w)$ . In Lemma 3, we bound  $\lambda$  by  $\sqrt{w}$  and  $\sum_{i=1}^t |\mathbb{K}_i|$  by  $\mathcal{O}(w \log w)$ . Therefore, the total time taken by the algorithm including the time taken by preprocessing step is  $\tilde{O}((n / \varepsilon^2)(\sqrt{r} + \sqrt{w} + \frac{wr}{n}))$ . Setting  $r = n^{2/3}$ , we get  $w = \mathcal{O}(n / (\varepsilon \sqrt{r})) = \mathcal{O}(n^{2/3} / \varepsilon)$ , and the total running time of our algorithm is  $\tilde{O}(n^{4/3} / \varepsilon^3)$ . To obtain the bottleneck matching, we execute this algorithm on  $\mathcal{O}(\log(n / \varepsilon))$  guesses; therefore, the total time taken to compute an  $\varepsilon$ -approximate bottleneck matching is  $\tilde{O}(n^{4/3} / \varepsilon^4)$ . For  $d > 2$ , we choose  $r = n^{\frac{d}{2d-1}}$  and  $w = \mathcal{O}(n / (d\varepsilon r^{1/d}))$ . With these values, the execution time of our algorithm is  $\frac{1}{\varepsilon^{\frac{d}{2d-1}}} n^{1 + \frac{d-1}{2d-1}}$  poly log  $n$ .

## References

- 1 P. K. Agarwal and K. R. Varadarajan. A near-linear constant-factor approximation for euclidean bipartite matching? In *Proc. 12th Annual Sympos. Comput. Geom.*, pages 247–252, 2004.
- 2 Pankaj K. Agarwal, Kyle Fox, Debmalya Panigrahi, Kasturi R. Varadarajan, and Allen Xiao. Faster Algorithms for the Geometric Transportation Problem. In *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, pages 7:1–7:16, 2017.
- 3 Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 555–564, 2014.
- 4 Mudabir Kabir Asuthulla, Sanjeev Khanna, Nathaniel Lahn, and Sharath Raghvendra. A Faster Algorithm for Minimum-Cost Bipartite Perfect Matching in Planar Graphs. In *SODA*, pages 457–476, 2018.
- 5 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In *FOCS*, pages 170–179, 2011.
- 6 A. Efrat, A. Itai, and M. J. Katz. Geometry Helps in Bottleneck Matching and Related Problems. *Algorithmica*, 31(1):1–28, September 2001. doi:10.1007/s00453-001-0016-8.
- 7 L. R. Ford Jr and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *No. RAND/P-743*, 1955.
- 8 H. N. Gabow and R.E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18:1013–1036, October 1989. doi:10.1137/0218069.
- 9 Harold N Gabow. The weighted matching approach to maximum cardinality matching. *Fundamenta Informaticae*, 154(1-4):109–130, 2017.
- 10 J. Hopcroft and R. Karp. An  $n^{5/2}$  Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.
- 11 Harold Kuhn. Variants of the Hungarian method for assignment problems. *Naval Research Logistics*, 3(4):253–258, 1956.
- 12 Nathaniel Lahn and Sharath Raghvendra. A Faster Algorithm for Minimum-Cost Bipartite Matching in Minor Free Graphs. In *SODA*, pages 569–588, 2019.
- 13 Yin Tat Lee and Aaron Sidford. Following the Path of Least Resistance : An  $\tilde{O}(m \sqrt{n})$  Algorithm for the Minimum Cost Flow Problem. *CoRR*, abs/1312.6713, 2013. arXiv:1312.6713.
- 14 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Foundations of Computer Science (FOCS)*, pages 253–262. IEEE, 2013.
- 15 Marcin Mucha and Piotr Sankowski. Maximum matchings via Gaussian elimination. In *FOCS*, pages 248–255, 2004.
- 16 Jeff M. Phillips and Pankaj K. Agarwal. On Bipartite Matching under the RMS Distance. In *Proceedings of the 18th Annual Canadian Conference on Computational Geometry, CCCG 2006, August 14-16, 2006, Queen's University, Ontario, Canada*, 2006.
- 17 R. Sharathkumar. A sub-quadratic algorithm for bipartite matching of planar points with bounded integer coordinates. In *SOCG*, pages 9–16, 2013.
- 18 R. Sharathkumar and P. K. Agarwal. A Near-Linear time Approximation Algorithm for Geometric Bipartite Matching. In *Proc. 44th Annual ACM Annual Sympos. on Theory of Comput.*, pages 385–394, 2012.
- 19 R. Sharathkumar and P. K. Agarwal. Algorithms for Transportation Problem in Geometric Settings. In *Proc. 23rd Annual ACM/SIAM Sympos. on Discrete Algorithms*, pages 306–317, 2012.



# The Unbearable Hardness of Unknotting

**Arnaud de Mesmay**

Univ. Grenoble Alpes, CNRS, Grenoble INP<sup>1</sup>, GIPSA-lab, 38000 Grenoble, France  
arnaud.de-mesmay@gipsa-lab.fr

**Yo'av Rieck**

Department of Mathematical Sciences, University of Arkansas Fayetteville, AR 72701, USA  
yoav@uark.edu

**Eric Sedgwick**

School of Computing, DePaul University, 243 S. Wabash Ave, Chicago, IL 60604, USA  
ESedgwick@cdm.depaul.edu

**Martin Tancer**

Department of Applied Mathematics, Charles University,  
Malostranské nám. 25, 118 00 Praha 1, Czech Republic  
tancer@kam.mff.cuni.cz

---

## Abstract

We prove that deciding if a diagram of the unknot can be untangled using at most  $k$  Reidemeister moves (where  $k$  is part of the input) is **NP**-hard. We also prove that several natural questions regarding links in the 3-sphere are **NP**-hard, including detecting whether a link contains a trivial sublink with  $n$  components, computing the unlinking number of a link, and computing a variety of link invariants related to four-dimensional topology (such as the 4-ball Euler characteristic, the slicing number, and the 4-dimensional clasp number).

**2012 ACM Subject Classification** Mathematics of computing → Geometric topology; Theory of computation → Problems, reductions and completeness

**Keywords and phrases** Knot, Link, NP-hard, Reidemeister move, Unknot recognition, Unlinking number, intermediate invariants

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.49

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1810.03502>.

**Funding** *Arnaud de Mesmay*: Partially supported by the French ANR projects ANR-16-CE40-0009-01 (GATO), the CNRS PEPS project COMP3D and the Czech-French collaboration project EMBEDS II (CZ: 7AMB17FR029, FR: 38087RM).

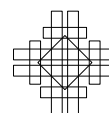
*Yo'av Rieck*: This work was partially supported by a grant from the Simons Foundation (grant number 283495 to Yo'av Rieck).

*Martin Tancer*: Partially supported by the Czech-French collaboration project EMBEDS II (CZ: 7AMB17FR029, FR: 38087RM), by the project CE-ITI (GAČR P202/12/G061) and by the Charles University projects PRIMUS/17/SCI/3 and UNCE/SCI/004.

**Acknowledgements** We thank Amey Kaloti and Jeremy Van Horn Morris for many helpful conversations. We are grateful to the anonymous referees for a careful reading of the paper and many helpful suggestions.

---

<sup>1</sup> Institute of Engineering Univ. Grenoble Alpes



## 1 Introduction

**Unknot recognition via Reidemeister moves.** The *unknot recognition problem* asks whether a given knot is the unknot. Decidability of this problem was established by Haken [10], and since then several other algorithms were constructed (see, e.g., the survey of Lackenby [20]).

One can ask, naively, if one can decide whether a given knot diagram represents the unknot simply by untangling the diagram: trying various Reidemeister moves until there are no more crossings. The problem is knowing when to stop: if we have not been able to untangle the diagram using so many moves, is the knot in question necessarily knotted or should we keep on trying? “Hard unknots” can be found, for example, in [14].

In [11], Hass and Lagarias gave an explicit (albeit rather large) bound on the number of Reidemeister moves needed to untangle a diagram of the unknot. Lackenby [18] improved the bound to polynomial thus showing that the unknot recognition problem is in **NP** (this was previously proved in [12]). The unknot recognition problem is also in **co-NP** [19] (assuming the Generalized Riemann Hypothesis, this was previously shown by Kuperberg [16]). Thus if the unknot recognition problem were **NP**-complete (or **co-NP**-complete) we would have that **NP** and **co-NP** coincide which is commonly believed not to be the case; see, for example, [9, Section 7.1]. This suggests that the unknot recognition problem is *not* **NP**-hard.

It is therefore natural to ask if there is a way to use Reidemeister moves leading to a better solution than a generic brute-force search. Our main result suggests that there may be serious difficulties in such an approach: given a 3-SAT instance  $\Phi$  we construct an unknot diagram and a number  $k$ , so that the diagram can be untangled using at most  $k$  Reidemeister moves if and only if  $\Phi$  is satisfiable. Hence any algorithm that can calculate the minimal number of Reidemeister moves needed to untangle unknot diagrams will be robust enough to tackle any problem in **NP**:

► **Theorem 1.** *Given an unknot diagram  $D$  and an integer  $k$ , deciding if  $D$  can be untangled using at most  $k$  Reidemeister moves is **NP**-complete.*

**NP**-membership follows from the result of Lackenby [18], so we only show **NP**-hardness. For the reduction in the proof of Theorem 1 we have to construct arbitrarily large diagrams of the unknot. The difficulty in the proof is to establish tools powerful enough to provide useful lower bounds on the minimal number of Reidemeister moves needed to untangle these diagrams. For instance, the algebraic methods of Hass and Nowik [13] do not appear strong enough for our reduction. It is also quite easy to modify the construction and give more easily lower bounds on the number of Reidemeister moves needed to untangle unlinks if one allows the use of arbitrarily many components of diagrams with constant size, but those techniques too cannot be used for Theorem 1. We develop the necessary tools in Section 4.

**Computational problems for links.** Our approach for proving Theorem 1 partially builds on techniques to encode satisfiability instances using Hopf links and Borromean rings, that we previously used in [7] (though the technical details are very different). With these techniques, we also show that a variety of link invariants are **NP**-hard to compute. Precisely, we prove:

► **Theorem 2.** *Given a link diagram  $L$  and an integer  $k$ , the following problems are **NP**-hard:*

- (a) *deciding whether  $L$  admits a trivial unlink with  $k$  components as a sublink.*
- (b) *deciding whether an intermediate invariant has value  $k$  on  $L$ ,*
- (c) *deciding whether  $\chi_4(L) = 0$ ,*
- (d) *deciding whether  $L$  admits a smoothly slice sublink with  $k$  components.*



We refer to Definition 13 for the definitions of  $\chi_4(L)$ , the 4-ball Euler characteristic, and intermediate invariants. These are broadly related to the topology of the 4-ball, and include the unlinking number, the ribbon number, the slicing number, the concordance unlinking number, the concordance ribbon number, the concordance slicing number, and the 4-dimensional clasp number. See, e.g., [25] for a discussion of many intermediate invariants.

**Related results.** The complexity of problems with knots and links is poorly understood. In particular, only very few computational lower bounds are known, and, as far as we know, almost none concern classical knots (i.e., knots embedded in  $S^3$ ): apart from our Theorem 1, the only other such hardness proof we know of [17, 24] concerns counting coloring invariants of knots. More lower bounds are known for classical links. Lackenby [21] showed that determining if a link is a sublink of another one is **NP**-hard. Our results strengthen this by showing that even finding a trivial sublink is already **NP**-hard. Agol, Hass and Thurston [1] showed that computing the genus of a knot in a 3-manifold is **NP**-hard, and Lackenby [21] showed that computing the Thurston complexity of a link in  $S^3$  is also **NP**-hard. Our results complement this by showing that the 4-dimensional version of this problem is also **NP**-hard.

Regarding upper bounds, the current state of knowledge is only slightly better. While, as we mentioned before, it is now known that the unknot recognition problem is in **NP**  $\cap$  **co-NP**, many natural link invariants are not even known to be decidable. In particular, this is the case for all the invariants for which we prove **NP**-hardness, except for the problem of finding the maximal number of components of a trivial sublink, which is in **NP** (see Theorem 5).

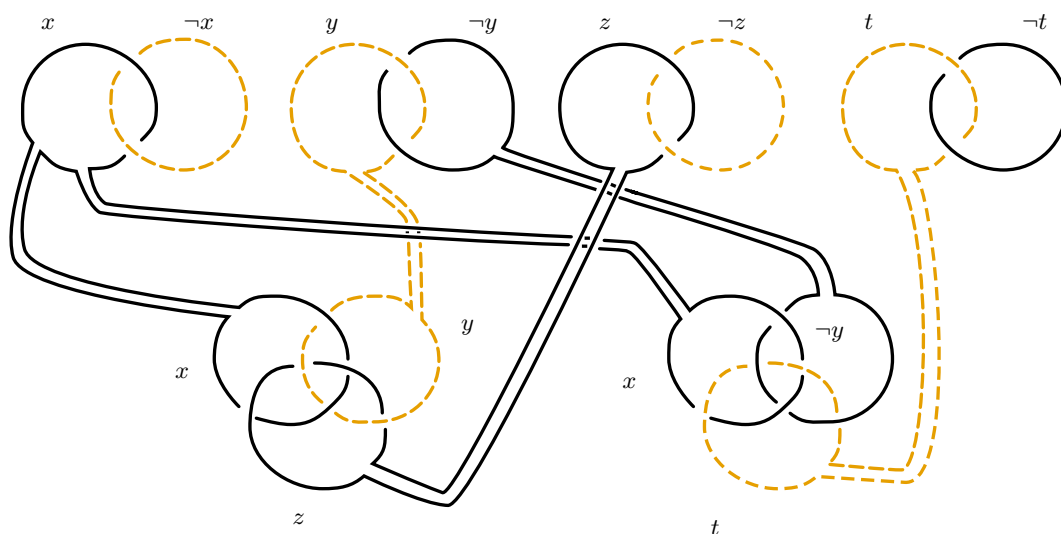
Shortly before we finished our manuscript, Koenig and Tsvietkova posted a preprint [15] that also shows that certain computational problems on links are **NP**-hard, with some overlap with the results obtained in this paper (the trivial sublink problem and the unlinking number). They also show **NP**-hardness of computing the number of Reidemeister moves for turning one unlink diagram into another, but their construction does not untangle the diagram and requires arbitrarily many components. Theorem 1 of the current paper is stronger and answers Question 17 of [15].

**Organization.** After some preliminaries in Section 2, we start by sketching the hardness of the trivial sublink problem in Section 3 because it is very simple and provides a good introduction for our other reductions. We then proceed to prove Theorem 1 in Sections 4 and 5 and the hardness of the unlinking number and the other invariants in Sections 6 and 7. These three proofs are mostly independent and the reader can read any of them separately.

## 2 Preliminaries

**Notation.** Most of the notation we use is standard. By *knot* we mean a tame piecewise linear embedding of the circle  $S^1$  into the 3-sphere  $S^3$ . By *link* we mean a tame, piecewise linear embedding of the disjoint union of any finite number of copies of  $S^1$ . We assume basic familiarity with computational complexity and knot theory, and refer to basic textbooks such as Arora and Barak [5] for the former and Rolfsen [23] for the latter.

**Diagram of a knot or a link.** All the computational problems that we study in this paper take as input the *diagram* of a knot or a link, which we define here. A diagram of a knot is a piecewise linear map  $D: S^1 \rightarrow \mathbb{R}^2$  in general position, obtained by composing the embedding  $S^1 \rightarrow \mathbb{R}^3$  with a projection  $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ . For such a map, every point in  $\mathbb{R}^2$  has at most two preimages, and there are finitely many points in  $\mathbb{R}^2$  with exactly two preimages (called *crossings*); at each crossing we indicate which arc crosses over and which crosses under. We similarly define a link diagram; for details see, for example [23].



■ **Figure 1**  $L_\Phi$  for  $\Phi = (x \vee y \vee z) \wedge (x \vee \neg y \vee t)$ , a satisfying assignment is dashed.

By an *arc* in the diagram  $D$  we mean a set  $D(\alpha)$  where  $\alpha$  is an arc in  $S^1$ , i.e., a subset of  $S^1$  homeomorphic to the closed interval (note that this definition is slightly non-standard).

The size of a knot or a link diagram is its number of crossings plus number of components of the link. Up to a constant factor, this complexity exactly describes the complexity of encoding the combinatorial information contained in a knot or link diagram.

**3-satisfiability.** In our reductions, we use the NP-hardness of the 3-SAT problem. An input is a formula in conjunctive normal form and we assume that each clause contains exactly three literals. For the proof of Theorem 1, we need a slightly restricted form of the 3-SAT problem given in the lemma below. The proof is simple (see Lemma 5 of the full version [8]).

► **Lemma 3** (Probably folklore). *Deciding whether a formula  $\Phi$  in conjunctive normal form is satisfiable is NP-hard even if we assume the following conditions on  $\Phi$ .*

- Each clause contains exactly three literals.
- No clause contains both  $x$  and  $\neg x$  for some variable  $x$ .
- Each pair of literals  $\{\ell_1, \ell_2\}$  occurs in at most one clause.

### 3 Trivial sublink

Informally, the trivial sublink problem asks, given a link  $L$  and a positive integer  $n$ , whether  $L$  admits the  $n$ -component unlink as a sublink. We define:

► **Definition 4** (The Trivial Sublink Problem). *An unlink, or a trivial link, is a link in  $S^3$  whose components bound disjointly embedded disks. A trivial sublink of a link  $L$  is an unlink formed by a subset of the components of  $L$ . The trivial sublink problem asks, given a link  $L$  and a positive integer  $n$ , whether  $L$  admits an  $n$  component trivial sublink.*

► **Theorem 5.** *The trivial sublink problem is NP-complete.*

For a complete proof of this theorem see Theorem 4 of the full version [8], yet it is simple and intuitive enough to be explained in a few words and a picture here.

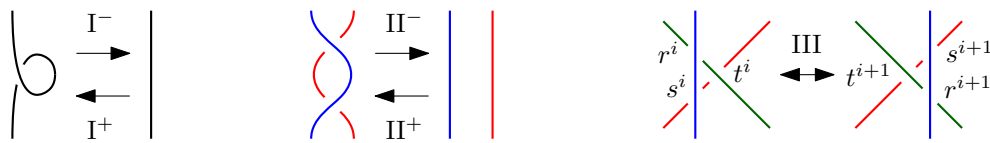


Figure 2 Reidemeister moves.

**Sketch.** NP membership follows from Hass, Lagarias and Pippenger [12] (also Lackenby [18]). For NP-hardness, starting with a 3-SAT instance  $\Phi$  with  $n$  variables, we construct a  $2n$ -component link  $L_\Phi$  (see Figure 1) consisting of a Hopf link for each variable and Borromean rings for each clause (each component corresponds to a literal as labeled). Each Borromean rings component is banded to the Hopf link component with the same label.

Given a satisfying assignment ( $x = z = \text{FALSE}$ ,  $y = t = \text{TRUE}$  in Figure 1), remove the components of satisfied literals (dashed yellow). Then from each set of Borromean rings at least one ring was removed; thus the sublink retracts into  $n$  separated unknots.

Conversely, let  $U_n$  be an  $n$ -component trivial sublink of  $L_\Phi$  (black in Figure 1). Since the Hopf link cannot be found in  $U_n$ , for each variable  $x$ , one of the components corresponding to  $x$  and  $\neg x$  is in  $U_n$ , and the other is not. Since the Borromean rings cannot be found in  $U_n$  from each set Borromean rings at least one ring is *not* in  $U_n$ . We conclude that the components *not* in  $U_n$  define a satisfying assignment for  $\Phi$ . ◀

#### 4 The defect

**Reidemeister moves.** Reidemeister moves are local modifications of a diagram depicted in Figure 2 (the labels at the crossings in a III move will be used only later on). We distinguish the I move (left), the II move (middle) and the III move (right). The first two moves affect the number of crossings, thus we further distinguish the  $I^-$  and the  $II^-$  moves which reduce the number of crossings from the  $I^+$  and the  $II^+$  moves which increase the number of crossings.

**The number of Reidemeister moves for untying a knot.** A diagram of the unknot is *untangled* if it does not contain any crossings. The untangled diagram is denoted by  $U$ . Given a diagram  $D$  of an unknot, an *untangling* of  $D$  is a sequence  $\mathbf{D} = (D^0, \dots, D^k)$  where  $D^0 = D$ ,  $D^k = U$  (recall that diagrams are only considered up to isotopy) and  $D^i$  is obtained from  $D^{i-1}$  by a single Reidemeister move. The number of Reidemeister moves in  $\mathbf{D}$  is denoted by  $\text{rm}(\mathbf{D})$ , that is,  $\text{rm}(\mathbf{D}) = k$ . We also define  $\text{rm}(D) := \min \text{rm}(\mathbf{D})$  where the minimum is taken over all untanglings  $\mathbf{D}$  of  $D$ .

**The defect.** Let us denote by  $\text{cross}(D)$  the number of crossings in  $D$ . We define the *defect* of an untangling  $\mathbf{D}$  by the formula

$$\text{def}(\mathbf{D}) := 2 \text{rm}(\mathbf{D}) - \text{cross}(D).$$

The *defect* of a diagram  $D$  is defined as  $\text{def}(D) := 2 \text{rm}(D) - \text{cross}(D)$ . Equivalently,  $\text{def}(D) = \min \text{def}(\mathbf{D})$  where the minimum is taken over all untanglings  $\mathbf{D}$  of  $D$ . The defect is a convenient way to reparametrize  $2 \text{rm}(\mathbf{D})$  due to the following observation.

► **Observation 6.** For any diagram  $D$  of the unknot and any untangling  $\mathbf{D}$  of  $D$  we have  $\text{def}(\mathbf{D}) \geq 0$ . Equality holds if and only if  $\mathbf{D}$  uses only  $II^-$  moves.

This notion of defect is different from the one that was introduced by Chang and Erickson [6] (following Arnold [3, 4] and Aicardi [2]) to study homotopy moves.

**Proof.** A Reidemeister move in  $\mathbf{D} = (D^0, \dots, D^k)$  removes at most two crossings and the  $\text{II}^-$  move is the only move that removes exactly two crossings. Thus, the number of crossings in  $D = D^0$  is at most  $2k$  and equality holds if and only if every move is a  $\text{II}^-$  move. ◀

**Crossings contributing to the defect.** Let  $\mathbf{D} = (D^0, \dots, D^k)$  be an untangling of a diagram  $D = D^0$  of an unknot. Given a crossing  $r^i$  in  $D^i$ , for  $0 \leq i \leq k - 1$ , it may vanish by the move transforming  $D^i$  into  $D^{i+1}$  if this is a  $\text{I}^-$  or a  $\text{II}^-$  move affecting the crossing. In all other cases it *survives* and we denote by  $r^{i+1}$  the corresponding crossing in  $D_{i+1}$ . Note that in the case of a  $\text{III}$  move there are three crossings affected by the move and three crossings afterwards. Both before and after, each crossing is the unique intersection between a pair of the three arcs of the knot that appear in this portion of the diagram. So we may say that these three crossings survive the move though they change their actual geometric positions (they swap the order in which they occur along each of the three arcs); see Figure 2.

With a slight abuse of terminology, by a *crossing in  $\mathbf{D}$*  we mean a maximal sequence  $\mathbf{r} = (r^a, r^{a+1}, \dots, r^b)$  such that  $r^{i+1}$  is the crossing in  $D^{i+1}$  corresponding to  $r^i$  in  $D^i$  for any  $a \leq i \leq b - 1$ . By maximality we mean that  $r^b$  vanishes after the  $(b + 1)$ st move and either  $a = 0$  or  $r^a$  is introduced by the  $a$ th Reidemeister move (which must be a  $\text{I}^+$  or  $\text{II}^+$  move).

An *initial crossing* is a crossing  $\mathbf{r} = (r^0, r^1, \dots, r^b)$  in  $\mathbf{D}$ . Initial crossings in  $\mathbf{D}$  are in one-to-one correspondence with crossings in  $D = D^0$ . For simplicity of notation,  $r^0$  is also denoted  $r$  (as a crossing in  $D$ ).

A Reidemeister  $\text{II}^-$  move in  $\mathbf{D}$  is *economical* if both crossings removed by this move are initial crossings; otherwise, it is *wasteful*. Let  $m_3(\mathbf{r})$  be the number of  $\text{III}$  moves affecting a crossing  $\mathbf{r}$ . The *weight* of an initial crossing  $\mathbf{r}$  is defined in the following way.

$$w(\mathbf{r}) = \frac{2}{3}m_3(\mathbf{r}) + \begin{cases} 0 & \text{if } \mathbf{r} \text{ vanishes by an economical } \text{II}^- \text{ move;} \\ 1 & \text{if } \mathbf{r} \text{ vanishes by a } \text{I}^- \text{ move;} \\ 2 & \text{if } \mathbf{r} \text{ vanishes by a wasteful } \text{II}^- \text{ move.} \end{cases}$$

For later purposes, we also define  $w(r) := w(\mathbf{r})$  and  $w(R) := \sum_{r \in R} w(r)$  for a subset  $R$  of the set of all crossings in  $D$ .

► **Lemma 7.** *Let  $\mathbf{D}$  be an untangling of a diagram  $D$ . Then*

$$\text{def}(\mathbf{D}) \geq \sum_{\mathbf{r}} w(\mathbf{r}),$$

where the sum is over all initial crossings  $\mathbf{r}$  of  $\mathbf{D}$ . If, in addition,  $\mathbf{D}$  uses only the  $\text{I}^-$  and  $\text{II}^-$  moves only, then (for the same sum) we get

$$\text{def}(\mathbf{D}) = \sum_{\mathbf{r}} w(\mathbf{r}) = \text{number of } \text{I}^- \text{ moves.}$$

The proof uses a discharging technique; see Lemmas 7 and 8 of the full version [8].

**Twins and the preimage of a bigon.** Let  $\mathbf{r}$  be an initial crossing in an untangling  $\mathbf{D} = (D^0, \dots, D^k)$  removed by an economical  $\text{II}^-$  move. The *twin* of  $\mathbf{r}$ , denoted by  $t(\mathbf{r})$  is the other crossing in  $\mathbf{D}$  removed by the same  $\text{II}^-$  move. Note that  $t(\mathbf{r})$  is also an initial crossing (because the move is economical). We also get  $t(t(\mathbf{r})) = \mathbf{r}$ . If  $\mathbf{r} = (r^0, \dots, r^b)$ , then we extend the definition of a twin to  $D^i$  in such a way that  $t(r^i)$  is uniquely defined by  $t(\mathbf{r}) = (t(r^0), \dots, t(r^b))$ . In particular,  $t(r)$  is a twin of a crossing  $r = r^0$  in  $D$  (if it exists).

Furthermore, the crossings  $r^b$  and  $t(r^b)$  in  $D^b$  form a bigon that is removed by the forthcoming  $\text{II}^-$  move. Let  $\alpha^b(r)$  and  $\beta^b(r)$  be the two arcs of the bigon (with endpoints  $r^b$  and  $t(r^b)$ ) so that  $\alpha^b(r)$  is the arc that, after extending slightly, overpasses the crossings  $r^b$  and  $t(r^b)$  whereas a slight extension of  $\beta^b(r)$  underpasses these crossings. (The reader may remember this as  $\alpha$  is “above” and  $\beta$  is “below”.) Now we can inductively define arcs  $\alpha^i(r)$  and  $\beta^i(r)$  for  $0 \leq i \leq b-1$  so that  $\alpha^i(r)$  and  $\beta^i(r)$  are the unique arcs between  $r^i$  and  $t(r^i)$  which are transformed to (already defined)  $\alpha^{i+1}(r)$  and  $\beta^{i+1}(r)$  by the  $(i+1)$ st Reidemeister move. We also set  $\alpha(r) = \alpha^0(r)$  and  $\beta(r) = \beta^0(r)$ . Intuitively,  $\alpha(r)$  and  $\beta(r)$  form a preimage of the bigon removed by the  $b$ th move and we call them the *preimage arcs* between  $r$  and  $t(r)$ .

**Close neighbors.** Let  $R$  be a subset of the set of crossings in  $D$ . Let  $r$  and  $s$  be any two crossings in  $D$  (not necessarily in  $R$ ) and let  $c \geq 0$  be an integer. We say that  $r$  and  $s$  are *c-close neighbors with respect to  $R$*  if  $r$  and  $s$  can be connected by two arcs  $\alpha$  and  $\beta$  such that

- $\alpha$  enters  $r$  and  $s$  as an overpass and  $\beta$  enters  $r$  and  $s$  as an underpass;
- $\alpha$  and  $\beta$  may have self-crossings; however, neither  $r$  nor  $s$  is in the interior of  $\alpha$  or  $\beta$ ; and
- $\alpha$  and  $\beta$  together contain at most  $c$  crossings from  $R$  in their interiors. (If there is a crossing in the interior of both  $\alpha$  and  $\beta$ , this crossing is counted only once.)

► **Lemma 8.** *Let  $R$  be a subset of the set of crossings in  $D$ , let  $c \in \{0, 1, 2, 3\}$ . Let  $r$  be the crossing in  $R$  which is the first of the crossings in  $R$  removed by an economical  $\text{II}^-$  move (we allow a draw). If  $w(R) \leq c$ , then  $r$  and its twin  $t(r)$  are  $c$ -close neighbors with respect to  $R$ .*

**Sketch.** See Lemma 9 of the full version [8]. Let  $\alpha(r)$  and  $\beta(r)$  be the preimage arcs between  $r$  and  $t(r)$ . We want to verify that they satisfy the properties of the arcs from the definition of the close neighbors. The first item follows immediately from the definition of preimage arcs. For the second item, if we had  $r$  or  $t(r)$  in the interior of  $\alpha$  or  $\beta$  then we cannot get a bigon between  $r^b$  and  $t(r^b)$  by subsequent moves. The third item is verified by an analysis of Reidemeister moves removing the crossings of  $\alpha$  and  $\beta$  before we get a bigon between  $r^b$  and  $t(r^b)$ , using  $c \leq 3$ : if  $r$  and  $t(r)$  are not  $c$ -close with respect to  $R$ , then the Reidemeister moves needed to “clean” the bigon for the  $\text{II}^-$  move contribute more than  $c$  to  $w(R)$ . ◀

## 5 The reduction

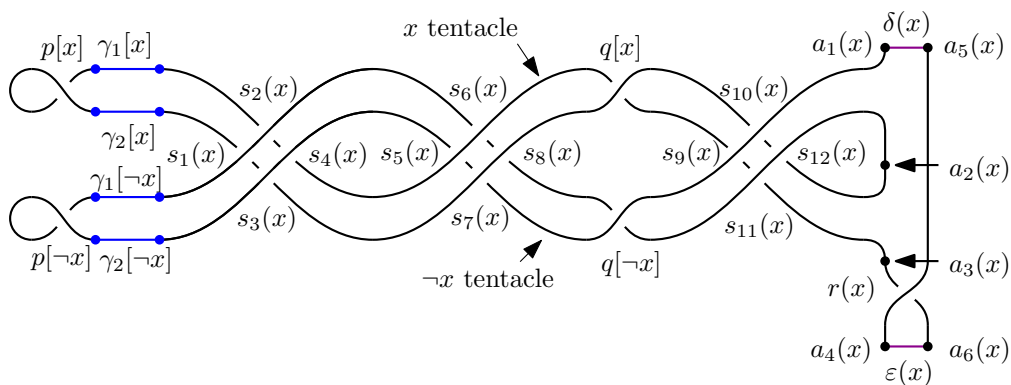
Let  $\Phi$  be a formula in conjunctive normal form satisfying the conditions stated in Lemma 3 and let  $n$  be the number of variables. Our aim is to build a diagram  $D(\Phi)$  by a polynomial-time algorithm such that  $\text{def}(D(\Phi)) \leq n$  if and only if  $\Phi$  is satisfiable.

**The variable gadget.** First we describe the variable gadget. For every variable  $x$  we consider the diagram depicted at Figure 3 and we denote it  $V(x)$ .

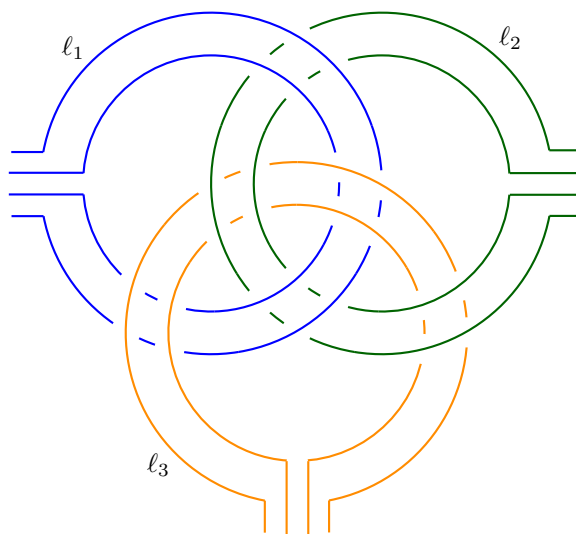
The gadget contains 17 crossings  $p[x], p[\neg x], q[x], q[\neg x], r(x)$  and  $s_i(x)$  for  $1 \leq i \leq 12$ .

The variable gadget also contains six distinguished arcs  $\gamma_i[x]$  and  $\gamma_i[\neg x]$  (for  $i = 1, 2$ ),  $\delta(x)$  and  $\varepsilon(x)$  and six distinguished auxiliary points  $a_1(x), \dots, a_6(x)$  which will be useful later on in order to describe how the variable gadget is used in the diagram  $D(\Phi)$ .

We also call the arc between  $a_1(x)$  and  $a_2(x)$  which contains  $\gamma_1[\neg x]$  and  $\gamma_2[\neg x]$  the  $\neg x$  *tentacle*, and similarly, we have removed  $\neg x$  tentacle between  $a_2(x)$  and  $a_3(x)$ . Informally, a satisfying assignment to  $\Phi$  will correspond to the choice whether we will decide to remove first the loop at  $p[x]$  by a  $\text{I}^-$  move and simplify the  $x$  tentacle or whether we remove first the loop at  $p[\neg x]$  and remove the  $\neg x$  tentacle in the final construction of  $D(\Phi)$ .



■ **Figure 3** The variable gadget  $V(x)$ .

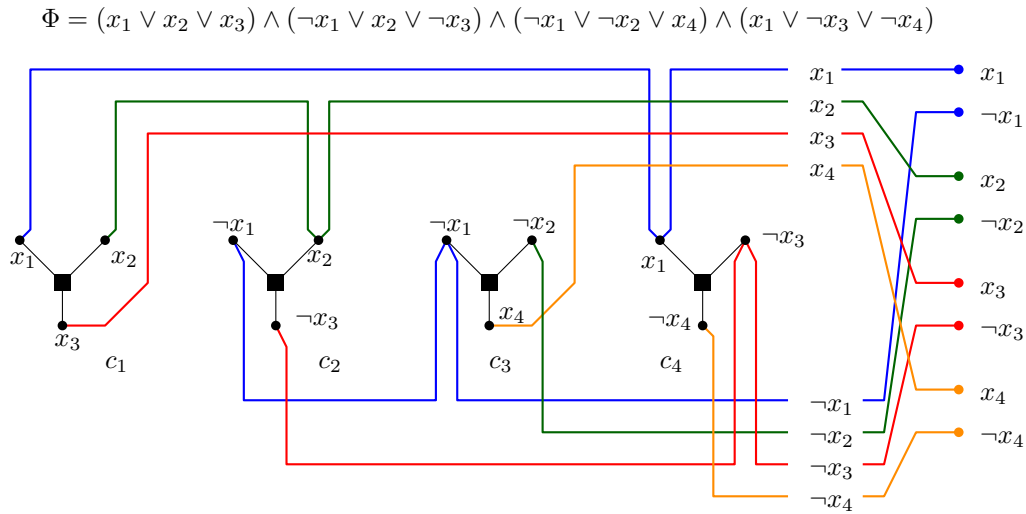


■ **Figure 4** The clause gadget for clause  $c = (\ell_1 \vee \ell_2 \vee \ell_3)$ .

We also remark that in the notation, we use square brackets for objects that come in pairs and will correspond to a choice of literal  $\ell \in \{x, \neg x\}$ . This regards  $p[\ell]$ ,  $q[\ell]$ ,  $\gamma_1[\ell]$  and  $\gamma_2[\ell]$  whereas we use parentheses for the remaining objects.

**The clause gadget.** Given a clause  $c = (\ell_1 \vee \ell_2 \vee \ell_3)$  in  $\Phi$ , the *clause gadget* is depicted at Figure 4. The construction is based on the Borromean rings. It contains three pairs of arcs (distinguished by color) and with a slight abuse of notation, we refer to each of the three pairs of arcs as a “ring”. Note that each ring has four pendent endpoints (or leaves) as in the picture. Each ring corresponds to one of the literals  $\ell_1$ ,  $\ell_2$ , and  $\ell_3$ .

**A blueprint for the construction.** Now we build a blueprint for the construction of  $D(\Phi)$ . Let  $x_1, \dots, x_n$  be the variables of  $\Phi$  and let  $c_1, \dots, c_m$  be the clauses of  $\Phi$ . For each clause  $c_j = (\ell_1 \vee \ell_2 \vee \ell_3)$  we take a copy of the graph  $K_{1,3}$  (also known as the star with three leaves). We label the vertices of degree 1 of such a  $K_{1,3}$  by the literals  $\ell_1$ ,  $\ell_2$ , and  $\ell_3$ . Now we draw these stars into the plane sorted along a horizontal line; see Figure 5. Next for each literal  $\ell \in \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$  we draw a piecewise linear segment containing all vertices labelled with that literal according to the following rules (follow Figure 5).



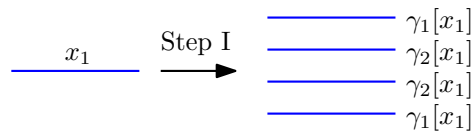
■ **Figure 5** A blueprint for the construction of  $D(\Phi)$ .

- The segments start on the right of  $K_{1,3}$ 's in the top down order  $x_1, \neg x_1, x_2, \dots, x_n, \neg x_n$ .
- They continue to the left while we permute them to the order  $x_1, \dots, x_n, \neg x_1, \dots, \neg x_n$ . We also require that  $x_1, \dots, x_n$  occur above the graphs  $K_{1,3}$  and  $\neg x_1, \dots, \neg x_n$  occur below these graphs (everything is still on the right of the graphs).
- For each literal  $\ell$  the segment for  $\ell$  continues to the left while it makes a “detour” to each vertex  $v$  labelled  $\ell$ . If  $v$  is not the leftmost vertex labelled  $\ell$ , then the detour is done by a “finger” of two parallel lines. Each finger avoids  $K_{1,3}$ 's except of  $v$ . If  $v$  is the leftmost, then we perform only a half of the finger so that  $v$  is the endpoint of the segment.

Note that the segments often intersect each other; however, for any  $i$  the segments for  $x_i$  and  $\neg x_i$  do not intersect (as we assume that no clause contains both  $x_i$  and  $\neg x_i$ ).

**The final diagram.** Finally, we explain how to build the diagram  $D(\Phi)$  from the blueprint.

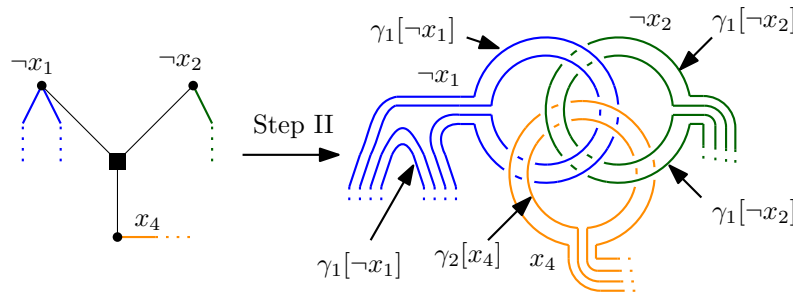
**Step I (four parallel segments):** We replace each segment for a literal  $\ell$  with four parallel segments; see Figure 6. The outer two will correspond to the arc  $\gamma_1[\ell]$  from the variable gadget and the inner two will correspond to  $\gamma_2[\ell]$ ; compare with Figure 3.



■ **Figure 6** Step I: Replacing segments.

**Step II (clause gadgets):** We replace each copy of  $K_{1,3}$  by a clause gadget for the corresponding clause  $c$ ; see Figure 7. Now we aim to describe how is the clause gadget connected to the quadruples of parallel segments obtained in Step I. Let  $v$  be a degree 1 vertex of the  $K_{1,3}$  we are just replacing. Let  $\ell$  be the literal which is the label of this vertex. Then  $c$  may or may not be the leftmost clause containing a vertex labelled  $\ell$ . If  $c$  is the leftmost clause containing a vertex labelled  $\ell$ , then there are four parallel segments for  $\ell$  with pendent endpoints (close to the original position of  $v$ ) obtained in



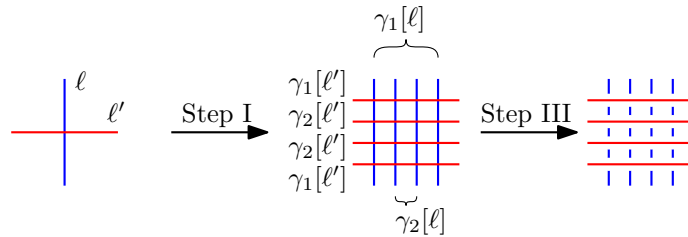


■ **Figure 7** Step II: Replacing the  $K_{1,3}$ .

Step I. We connect them to the pendent endpoints of the clause gadget (on the ring for  $\ell$ ); see  $\neg x_2$  and  $x_4$  in Figure 7. Note also that at this moment the two  $\gamma_1[\ell]$  arcs introduced in Step I merge as well as the two  $\gamma_2[\ell]$  arcs merge.

If  $c$  is not the leftmost clause labelled  $\ell$  then there are four parallel segments passing close to  $v$  (forming a tip of a finger from the blueprint). We disconnect the two segments closest to the tip of the finger and connect them to the pendent endpoints of the clause gadget (on the ring for  $\ell$ ); see  $\neg x_1$  in Figure 7.

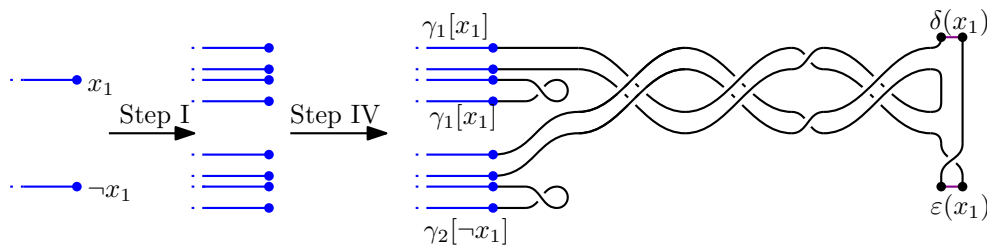
**Step III (resolving crossings):** If two segments in the blueprint, corresponding to literals  $\ell$  and  $\ell'$  cross, Step I blows up each such crossing into 16 crossings of corresponding quadruples. We resolve overpasses/underpasses at these crossings in the same way; see Figure 8.



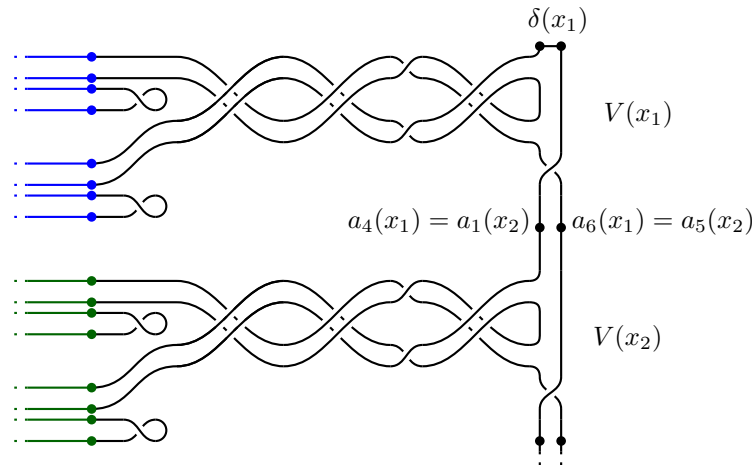
■ **Figure 8** Step III: Resolving crossings.

However, we require one additional condition on the choice of overpasses/underpasses. If  $\ell$  and  $\ell'$  appear simultaneously in some clause  $c$  we have 8 crossings on the rings for  $\ell$  and  $\ell'$  in the clause gadget for  $c$ . We can assume that the ring of  $\ell$  passes over the ring of  $\ell'$  at all these crossings (otherwise we swap  $\ell$  and  $\ell'$ ). Then for the 16 crossings on segments for  $\ell$  and  $\ell'$  we pick the other option, that is we want that the  $\gamma_1[\ell]$  and  $\gamma_2[\ell]$  arcs underpass the  $\gamma_1[\ell']$  and  $\gamma_2[\ell']$  arcs at these crossings. This is a globally consistent choice because we assume that there is at most one clause containing both  $\ell$  and  $\ell'$ , this is the third condition in the statement of Lemma 3.

**Step IV (the variable gadgets):** For each variable  $x_i$ , the segments  $\gamma_1[x_i], \gamma_2[x_i], \gamma_1[\neg x_i]$  and  $\gamma_2[\neg x_i]$  do not intersect each other. We extend them to a variable gadget as in Figure 9. Namely, to the bottom right endpoints of  $\gamma_1[x_i], \gamma_2[x_i], \gamma_1[\neg x_i]$  and  $\gamma_2[\neg x_i]$  we glue the parts of the variable gadget containing the crossings  $p[x_i]$  and  $p[\neg x_i]$  and to the top right endpoints of  $\gamma_1[x_i], \gamma_2[x_i], \gamma_1[\neg x_i]$  and  $\gamma_2[\neg x_i]$  we glue the remainder of the variable gadget. At this moment, we obtain a diagram of a link, where each link component has a diagram isotopic to the diagram on Figure 3.



■ **Figure 9** Step IV: Adding the variable gadgets.



■ **Figure 10** Step V: Interconnecting the variable gadgets.

**Step V (interconnecting the variable gadgets):** Finally, we form a connected sum of individual components. Namely, for every  $1 \leq i \leq n - 1$  we perform the knot sum along the arcs  $\delta(x_i)$  and  $\varepsilon(x_{i+1})$  by removing them and identifying  $a_4(x_i)$  with  $a_1(x_{i+1})$  and  $a_6(x_i)$  with  $a_5(x_{i+1})$  as on Figure 10. The arcs  $\delta(x_1)$  and  $\varepsilon(x_n)$  remain untouched. This way we obtain the desired unknot diagram  $D(\Phi)$ ; see Figure 11.

The core of the **NP**-hardness reduction is the following theorem.

► **Theorem 9.** *Let  $\Phi$  be a formula with  $n$  variables in conjunctive normal form satisfying the conditions in the statement of Lemma 3. Then  $\text{def}(D(\Phi)) \leq n$  if and only if  $\Phi$  is satisfiable.*

Theorem 1 immediately follows from Theorem 9 and Lemma 3:

**Proof of Theorem 1 modulo Theorem 9.** Due to the definition of the defect, the minimum number of Reidemeister moves required to untangle  $D$  equals  $\frac{1}{2}(\text{def}(D(\Phi)) + \text{cross}(D(\Phi)))$ . Therefore, with  $k = \frac{1}{2}(n + \text{cross}(D(\Phi)))$ , Theorem 9 gives that  $D(\Phi)$  can be untangled with at most  $k$  moves if and only if  $\Phi$  is satisfiable. This gives the required **NP**-hardness via Lemma 3. (Note that  $D(\Phi)$  and  $k$  can be obtained in polynomial time in the size of  $\Phi$ .) ◀

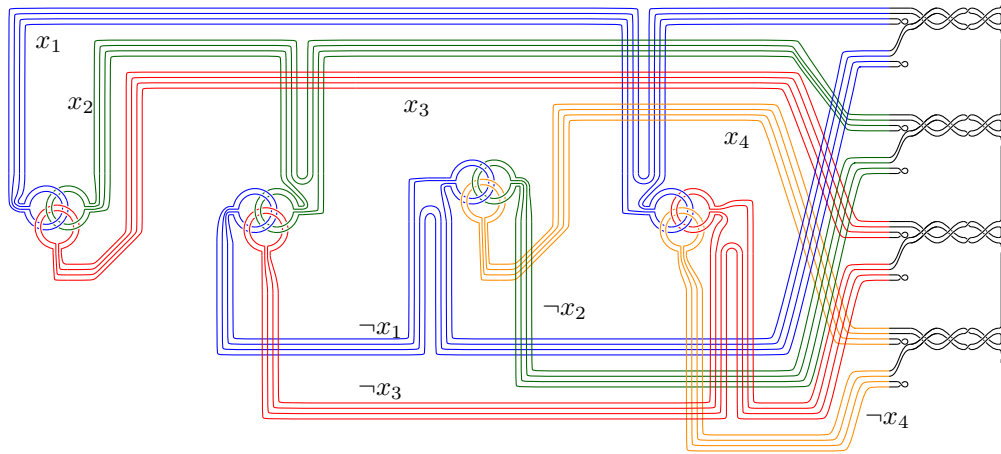
The remainder of this section is devoted to the proof of Theorem 9.

## 5.1 Satisfiable implies small defect

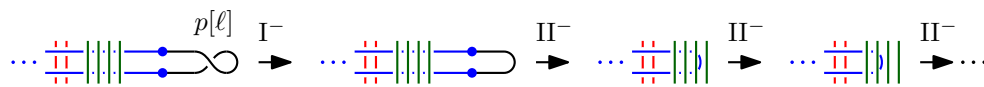
In this subsection we show that given a satisfying assignment for  $\Phi$ ,  $\text{def}(D(\Phi)) \leq n$ .

49:12 The Unbearable Hardness of Unknotting

$$\Phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$$



■ **Figure 11** The final construction for the formula  $\Phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee \neg x_4)$ . For simplicity of the picture, we do not visualize how the crossings are resolved in Step III. (Unfortunately, we cannot avoid tiny pictures of gadgets.)



■ **Figure 12** Initial simplifications.

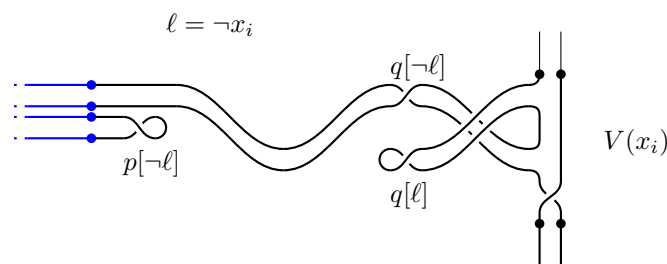
For any literal  $\ell = \text{TRUE}$ , we remove the crossing  $p[\ell]$  by a  $I^-$  move (see Figure 3). This uses exactly  $n$   $I^-$  moves. Next we finish the untangling of the diagram by  $II^-$  moves only. Once this is done, we get an untangling with defect  $n$  by Lemma 7, completing the proof.

Thus it remains to finish the untangling with  $II^-$  moves only. For any  $\ell = \text{TRUE}$ , we shrink the  $\ell$  tentacle by  $II^-$  moves (see Figure 12). By construction of  $D(\Phi)$  we can continue shrinking the  $\ell$  tentacle until we get a loop next to the  $q[\ell]$  vertex; see Figure 13.

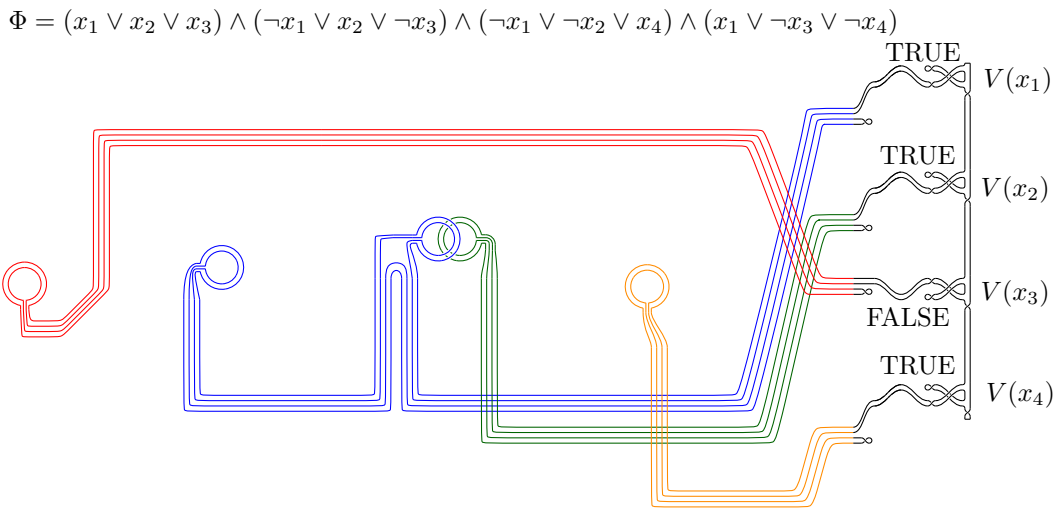
We continue the same process for every literal  $\ell = \text{TRUE}$ . Along the way, some of the arcs meeting  $\gamma_1[\ell]$  and  $\gamma_2[\ell]$  might have already been removed but it is still possible to simplify the  $\ell$  tentacle as before. See Figure 14 for the result after shrinking all satisfied tentacles.

Because the assignment was satisfying, in each clause gadget at least one ring of the Borromean rings disappears. Consequently, if there are two remaining rings in some clause gadget, then they can be pulled apart from each other by  $II^-$  moves as in Figure 15.

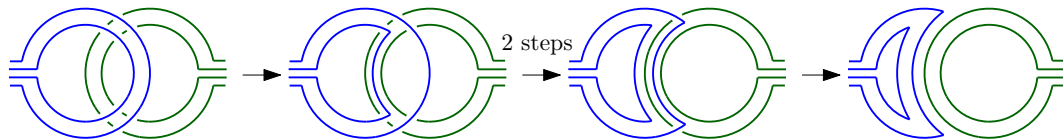
After this step, for each  $\ell = \text{TRUE}$ , the  $\gamma_1[\neg\ell]$  and  $\gamma_2[\neg\ell]$  form “fingers” of four parallel curves that can be further simplified by  $II^-$  moves so that any crossings among different



■ **Figure 13** The  $\ell$  tentacle was shrunk to a loop next to  $q[\ell]$ . In this example we have  $\ell = \neg x_i$ .



■ **Figure 14** Simplified  $D(\Phi)$  following a satisfying assignment  $x_1 = x_2 = x_4 = \text{TRUE}$  and  $x_3 = \text{FALSE}$ .



■ **Figure 15** Untangling two rings in the clause gadget via  $\text{II}^-$  moves.

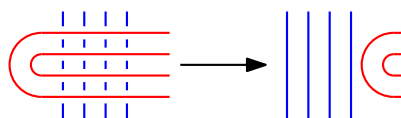
fingers are removed; see Figure 16. For each variable gadget we get one of the two pictures at Figure 17 left, and can be simplified to the picture on the right using  $\text{II}^-$  moves.

Finally, we recall how the variable gadgets are interconnected (compare the right picture at Figure 17 with Figure 10). Then it is easy to remove all remaining  $2n$  crossings by  $\text{II}^-$  moves gradually from top to bottom. This finishes the proof of the “if” part of Theorem 9.

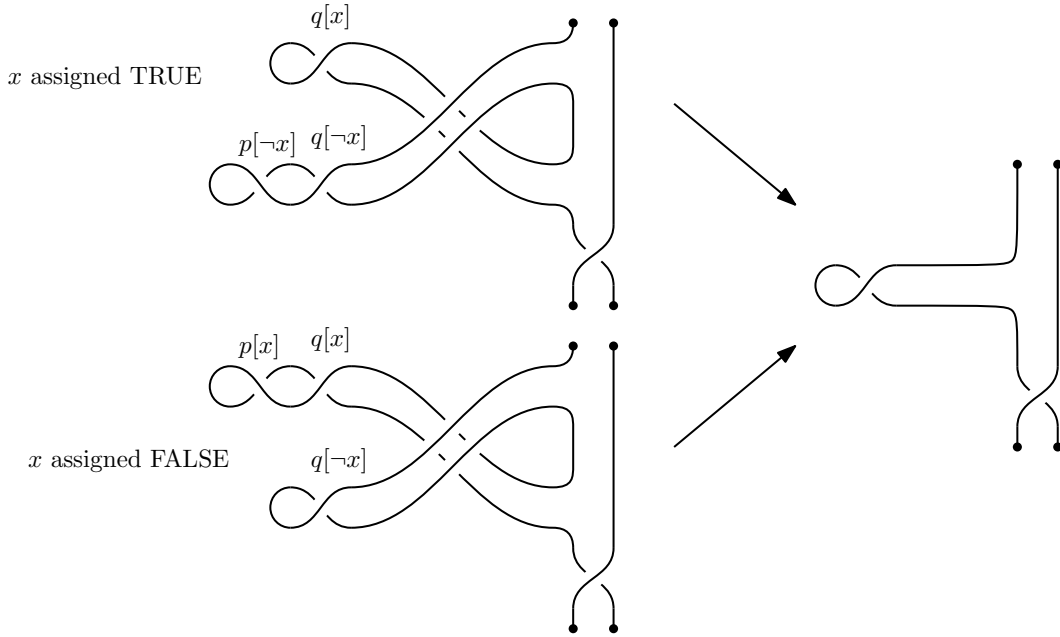
### 5.2 Small defect implies satisfiable

The purpose of this subsection is to sketch a proof of the “only if” part of the statement of Theorem 9. Recall that this means that we assume  $\text{def}(D(\Phi)) \leq n$  and we want to deduce that  $\Phi$  is satisfiable. (Along the way we will actually also deduce that  $\text{def}(D(\Phi)) = n$ .) In this subsection, we heavily use the terminology introduced in Section 4.

Let  $\mathbf{D} = (D^0, \dots, D^k)$  be an untangling of  $D$  with  $\text{def}(\mathbf{D}) \leq n$ . For a variable  $x$  let  $R(x) = \{p[x], p[\neg x], q[x], q[\neg x], s_1(x), \dots, s_{12}(x)\}$  be the set of 16 out of the 17 self-crossings in the variable gadget  $V(x)$  (we leave out  $r(x)$ ) and let the *weight* of  $x$ , denoted by  $w(x)$ , be the sum of weights of the crossings in  $R(x)$ .



■ **Figure 16** Simplifying  $\gamma_1[-\ell]$  and  $\gamma_2[-\ell]$  via  $\text{II}^-$  moves. First, we untangle the inner (horizontal) “finger” and then we untangle the outer (horizontal) “finger”.



■ **Figure 17** Results of the simplifications on the previous picture on the level of variable gadgets.

Now we provide two claims on the order of removals of the crossings. Both claims are proved using Lemma 8 via a careful case analysis, see Claims 10.1 and 10.2 of the full version [8]. The first claim analyzes the first economical  $\text{II}^-$  move that removes some of the crossings in  $R(x)$ . For these claims, recall Figure 3.

▷ **Claim 10.** Let  $x$  be a variable with  $w(x) \leq 1$ . Let  $r$  be the first crossing in  $R(x)$  which is removed by an economical  $\text{II}^-$  move (we allow a draw). Then one of the following cases holds (note that in both cases  $w(x) = 1$ ):

- (i)  $\{r, t(r)\} = \{s_1(x), s_2(x)\}$ ,  $w(p[x]) = w(x) = 1$  and  $p[x]$  is removed by a  $\text{I}^-$  move prior to removing  $r$  and  $t(r)$ .
- (ii)  $\{r, t(r)\} = \{s_1(x), s_3(x)\}$ ,  $w(p[\neg x]) = w(x) = 1$  and  $p[\neg x]$  is removed by a  $\text{I}^-$  move prior to removing  $r$  and  $t(r)$ .

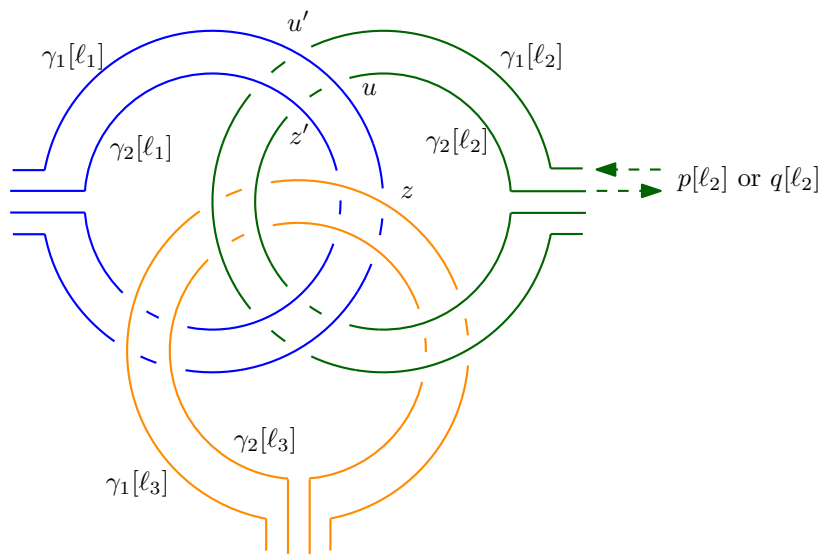
Now, let us set  $\ell := \ell(x) := x$  if the conclusion (i) of Claim 10 holds and  $\ell := \ell(x) := \neg x$  if (ii) holds (assuming  $w(x) \leq 1$ ). (We identify  $\neg\neg x$  with  $x$ , that is, if  $\ell = \neg x$ , then  $\neg\ell = x$ .)

▷ **Claim 11.** If  $w(x) \leq 1$ , then  $p[\neg\ell]$  and  $q[\neg\ell]$  are twins. In addition, the preimage arcs  $\alpha$  and  $\beta$  between  $p[\neg\ell]$  and  $q[\neg\ell]$  contain  $\gamma_1[\neg\ell]$  and  $\gamma_2[\neg\ell]$ .

Now, we have acquired enough tools to finish the proof of the theorem. By Claim 10, we have  $w(x) \geq 1$  for any variable  $x$ . By Lemma 7, we deduce

$$\text{def}(\mathbf{D}) \geq \sum_x w(x) \geq n,$$

where the sum is over all variables. On the other hand, we assume  $\text{def}(\mathbf{D}) \leq n$ . Therefore both inequalities above have to be equalities and in particular  $w(x) = 1$  for any variable  $x$ . In particular, the assumptions of Claims 10 and 11 are satisfied for any variable  $x$ .



■ **Figure 18** The clause gadget and some of the crossings of  $R''(c)$ .

Given a variable  $x$ , we assign  $x$  with TRUE if the conclusion (i) of Claim 10 holds (that is, if  $x = \ell(x)$ ). Otherwise, if the conclusion (ii) of Claim 10 holds (i.e.  $\neg x = \ell(x)$ ), we set  $x$  to FALSE. It remains to prove that we get a satisfying assignment this way.

For contradiction, suppose there is a clause  $c = (\ell_1 \vee \ell_2 \vee \ell_3)$  which is not satisfied with this assignment. Let  $x_i$  be the variable of  $\ell_i$ , that is,  $\ell_i = x_i$  or  $\ell_i = \neg x_i$ . The fact that  $c$  is not satisfied with the assignment above translates as  $\ell(x_i) = \neg \ell_i$  for any  $i \in \{1, 2, 3\}$ .

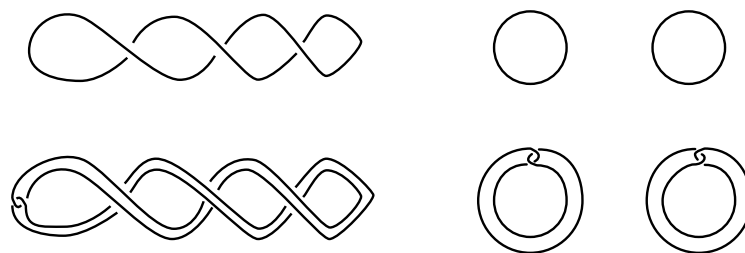
By Claim 11, we get that  $p[\ell_i]$  and  $q[\ell_i]$  are twins for any  $i \in \{1, 2, 3\}$ . Let  $R''(c)$  be the set of crossings in the clause gadget of  $c$  union the sets  $\{p[\ell_i], q[\ell_i]\}$  for  $i \in \{1, 2, 3\}$ . All the crossings in  $R''(c)$  have weight 0 and they have to be removed by economical  $\Pi^-$  moves as all defect is realized on points  $p[\ell(x)]$  (but  $p[\ell_i] = p[\neg \ell(x_i)]$  are not among these points).

Let  $r$  be the first removed crossing among the crossings in  $R''(c)$ . Our aim is to rule out all options in which a crossing of  $R''(c)$  can be  $r$ . This way, we will obtain the required contradiction. First, we observe that  $r$  cannot be any of  $p[\ell_i]$  or  $q[\ell_i]$  for  $i \in \{1, 2, 3\}$ . This follows from Claim 11 as the arcs  $\gamma_1[\ell_i]$  and  $\gamma_2[\ell_i]$  contain some crossings in  $R''(c)$ .

Next we apply Lemma 8 (for 0-close neighbors) with  $R = R''(c)$ . For sake of example, in this sketch we rule out the (interesting) option  $r = u$ , where  $u$  is the vertex on Figure 18. Let  $\alpha$  and  $\beta$  be the arcs between  $r$  and  $t(r)$  from the definition of 0-close neighbors ( $\alpha$  and  $\beta$  exist by Lemma 8). In particular, neither  $\alpha$  nor  $\beta$  has a crossing from  $R''(c)$  in its interior. The only option for  $\alpha$  is to emanate to the left reaching the crossing  $u'$  as emanating to the right reaches a crossing  $z \in R''(c)$  as an underpass. In particular,  $t(u) = u'$ . Consequently,  $\beta$  has to emanate to the right since emanating to the left would reach  $z'$ . However, before  $\beta$  reaches  $u'$ , it has to pass through  $p[\ell_2]$  or  $q[\ell_2]$  in  $R''(c)$  which rules out this option.

## 6 Intermediate invariants

In this section we describe the family of link invariants from the statement of Theorem 2. The material presented here is standard and details can be found in various textbooks; since every piecewise linear knot can be smoothed in a unique way, we assume that the knots discussed are smooth. In Sections 6 and 7 we work in the smooth category. We first define:



■ **Figure 19** Some Whitehead doubles.

► **Definition 12.** Let  $L$  be a link in the 3-sphere. We now give a list of the invariants that we will be using; for a detailed discussion see, for example, [23].

1. A smooth slice surface for  $L$  is an orientable surface with no closed components, properly and smoothly embedded in the 4-ball, whose boundary is  $L$ .
2. The 4-ball Euler characteristic of  $L$ , denoted  $\chi_4(L)$ , is the largest integer so that  $L$  bounds a smooth slice surface of Euler characteristic  $\chi_4(L)$ .
3. A link is called smoothly slice if it bounds a slice surface that consists entirely of disks; equivalently, the  $\chi_4(L)$  equals the number of components of  $L$ .
4. The unlinking number, denoted  $u(L)$ , is the smallest nonnegative integer so that  $L$  admits some diagram  $D$  so that after  $u(L)$  crossing changes on  $D$  a trivial link is obtained.
5. By transversality, every link  $L$  bounds smoothly immersed disks in  $B^4$  with finitely many double points. The 4-dimensional clasp number (sometimes called the 4-ball crossing number) of  $L$ , denoted  $c_s(L)$ , is the minimal number of double points for such disks.

Finally, we define intermediate invariants, whose existence follows from Theorem 2 of [25].

► **Definition 13** (intermediate invariant). A real valued link invariant  $i(L)$  is called an intermediate invariant if  $u(L) \geq i(L) \geq c_s(L)$ .

Many invariants are known to be intermediate (see, for example, [25]). We list a few (and prove that they are intermediate) in Lemma 13 of the full version [8]. Shibuya [25] proved:

► **Lemma 14.** Let  $L$  be a link with  $\mu$  components. Then  $\chi_4(L) \geq \mu - 2c_s(L)$ .

## 7 Unlinking, 4-ball Euler characteristic, and intermediate invariants

In this section we will show that the link invariants defined in the previous section are NP-hard. Our reduction relies on the use of Whitehead doubles, which we now define:

► **Definition 15** (Whitehead Double). Let  $L$  be a link. A Whitehead double of  $L$  is a link obtained by taking two parallel copies of each component of  $L$  and joining them together with a clasp (see Figure 19). A Whitehead double is called positive if the crossings at the clasp are positive. If the linking number of the two copies of each component is zero the Whitehead double is called untwisted. It is easy to see that the untwisted positive Whitehead double is uniquely determined by  $L$ .

One last piece of background we will need is a result of Levine [22, Theorem 1.1]:

► **Lemma 16.** The untwisted positive Whitehead double of the Hopf link, and that of the Borromean rings, are not smoothly slice.

We are now ready to describe our construction:



**The construction of  $L_{\Phi}^{\text{WH}}$ .** Given a 3-SAT instance  $\Phi$ , recall the link  $L_{\Phi}$  from Section 3, and let  $L_{\Phi}^{\text{WH}}$  be its positive untwisted Whitehead double. There is a natural bijection between components before and after taking a Whitehead double; let  $\kappa_{x_i}^{\text{WH}}$  denote the component corresponding to  $\kappa_{x_i}$  and let  $\kappa_{\neg x_i}^{\text{WH}}$  denote the component corresponding to  $\kappa_{\neg x_i}$ .

The goal of this section is to prove:

► **Theorem 17.** *Given a 3-SAT instance  $\Phi$  with  $n$  variables, let  $L_{\Phi}^{\text{WH}}$  be the link constructed above. Then the following are equivalent, where here  $i$  is any intermediate invariant:*

1.  $\Phi$  is satisfiable.
2.  $u(L_{\Phi}^{\text{WH}}) = n$ .
3.  $i(L_{\Phi}^{\text{WH}}) = n$ .
4.  $c_s(L_{\Phi}^{\text{WH}}) = n$ .
5.  $\chi_4(L_{\Phi}^{\text{WH}}) = 0$ .
6.  $L_{\Phi}^{\text{WH}}$  admits a smoothly slice sublink with  $n$  components.

Theorem 2(b) – (d) directly follows from Theorem 17.

**Proof.** The proof of this Theorem is split in the following steps:

- (a)  $\Phi$  is satisfiable implies that  $u(L_{\Phi}^{\text{WH}}) \leq n$ .
- (b)  $c_s(L_{\Phi}^{\text{WH}}) \leq i(L_{\Phi}^{\text{WH}}) \leq u(L_{\Phi}^{\text{WH}})$ .
- (c)  $\chi_4(L_{\Phi}^{\text{WH}}) \geq 2n - 2c_s(L_{\Phi}^{\text{WH}})$ .
- (d) If  $\chi_4(L_{\Phi}^{\text{WH}}) \geq 0$  then the following two conditions hold:
  - (d.I)  $\chi_4(L_{\Phi}^{\text{WH}}) = 0$
  - (d.II)  $L_{\Phi}^{\text{WH}}$  admits a smoothly slice sublink with  $n$  components.
- (e) If  $L_{\Phi}^{\text{WH}}$  admits a smoothly slice sublink with  $n$  components, then  $\Phi$  is satisfiable.

We first show how (a) – (e) prove Theorem 17. Assume first that  $\Phi$  is satisfiable. Then by (a) and (b) we have that  $c_s(L_{\Phi}^{\text{WH}}) \leq n$ , and by (c) we have that  $\chi_4(L_{\Phi}^{\text{WH}}) \geq 0$ . Then (d.I) shows that  $\chi_4(L_{\Phi}^{\text{WH}}) = 0$ . Working our way back, we see that  $c_s(L_{\Phi}^{\text{WH}}) = i(L_{\Phi}^{\text{WH}}) = u(L_{\Phi}^{\text{WH}}) = n$ , establishing (1)  $\Rightarrow$  (2)  $\Rightarrow$  (3)  $\Rightarrow$  (4)  $\Rightarrow$  (5). In addition, (d.II) shows directly that (5)  $\Rightarrow$  (6). Finally, (e) establishes (6)  $\Rightarrow$  (1).

We complete the proof of Theorem 17 by establishing (a) – (e):

- (a) Suppose we have a satisfying assignment for  $\Phi$  (for this implication, cf. the proof of Theorem 5). By a single crossing change we resolve the clasp of every component that correspond to a satisfied literal, that is, if  $x_i = \text{TRUE}$  we change one of the crossings of the clasp of  $\kappa_{x_i}^{\text{WH}}$  and if  $x_i = \text{FALSE}$  we do it for  $\kappa_{\neg x_i}^{\text{WH}}$ ; as a result, the components corresponding to satisfied literals now form unlinks that are not linked with the remaining components, and we can isotope them away. Since the assignment is satisfying, from each copy of the Borromean rings at least one ring is removed, and the remaining components retract into the first  $n$  disks that contained the Hopf links. In each disk we have an untwisted Whitehead double of the unknot which is itself an unknot. Thus we see that the unlink on  $2n$  component is obtained, showing that  $u(L_{\Phi}^{\text{WH}}) \leq n$ .
- (b) By definition of intermediate invariant we have that  $c_s \leq i \leq u$ .
- (c) This is Lemma 14.
- (d) This step is the crux of the proof and relies on the computation of the *signature* of our link; see step (d) in the proof of Theorem 20 of the full version [8].

- (e) Finally, assume that  $L_{\Phi}^{\text{WH}}$  admits an  $n$  component smoothly slice sublink  $L_{\text{slice}}$ . By Lemma 16 we have that the positive untwisted Whitehead double of the Hopf link is not a sublink of  $L_{\text{slice}}$  and therefore for each  $i$  exactly one of  $\kappa_{x_i}^{\text{WH}}$  and  $\kappa_{\neg x_i}^{\text{WH}}$  is in  $L_{\text{slice}}$ . If  $\kappa_{x_i}^{\text{WH}}$  is in  $L_{\text{slice}}$  we set  $x_i = \text{FALSE}$  and if  $\kappa_{\neg x_i}^{\text{WH}}$  is in  $L_{\text{slice}}$  we set  $x_i = \text{TRUE}$ . Using Lemma 16 again we see that  $L_{\text{slice}}$  does not admit the positive untwisted Whitehead double of the Borromean rings as a sublink. Therefore, from every set of Borromean rings, at least one component does *not* belong to  $L_{\text{slice}}$ ; therefore the assignment is satisfying. ◀

---

## References

- 1 Ian Agol, Joel Hass, and William Thurston. The computational complexity of knot genus and spanning area. *Transactions of the American Mathematical Society*, 358:3821–3850, 2006. doi:10.1090/S0002-9947-05-03919-X.
- 2 Francesca Aicardi. Tree-like curves. *Singularities and bifurcations*, 21:1–31, 1994.
- 3 Vladimir I Arnold. Plane curves, their invariants, perestroikas and classifications. *Advances in Soviet Mathematics*, 21:33–91, 1994.
- 4 Vladimir Igorevich Arnold. *Topological invariants of plane curves and caustics*, volume 5. American Mathematical Soc., 1994.
- 5 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, Cambridge, 2009. doi:10.1017/CB09780511804090.
- 6 Hsien-Chih Chang and Jeff Erickson. Untangling planar curves. *Discrete & Computational Geometry*, 58(4):889–920, 2017. doi:10.1007/s00454-017-9907-6.
- 7 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. Embeddability in  $\mathbb{R}^3$  is NP-hard. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1316–1329. Society for Industrial and Applied Mathematics, 2018. Full version on arXiv:1604.00290. doi:10.1137/1.9781611975031.86.
- 8 Arnaud de Mesmay, Yo'av Rieck, Eric Sedgwick, and Martin Tancer. The unbearable hardness of unknotting. arXiv:1810.03502 (for numbering of claims, we refer to the version v1), 2018.
- 9 Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.
- 10 Wolfgang Haken. Theorie der Normalflächen. *Acta Mathematica*, 105(3-4):245–375, 1961.
- 11 Joel Hass and Jeffrey Lagarias. The number of Reidemeister moves needed for unknotting. *Journal of the American Mathematical Society*, 14(2):399–428, 2001. doi:10.1090/S0894-0347-01-00358-7.
- 12 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM (JACM)*, 46(2):185–211, 1999. doi:10.1145/301970.301971.
- 13 Joel Hass and Tahl Nowik. Unknot diagrams requiring a quadratic number of Reidemeister moves to untangle. *Discrete & Computational Geometry*, 44(1):91–95, 2010. doi:10.1007/s00454-009-9156-4.
- 14 Louis H Kauffman and Sofia Lambropoulou. Hard unknots and collapsing tangles. In *Introductory Lectures on Knot Theory*, 2014. doi:10.1142/9789814313001\_0009.
- 15 Dale Koenig and Anastasiia Tsvietkova. NP-hard problems naturally arising in knot theory, 2018. arXiv:1809.10334.
- 16 Greg Kuperberg. Knottedness is in NP, modulo GRH. *Adv. Math.*, 256:493–506, 2014. doi:10.1016/j.aim.2014.01.007.
- 17 Greg Kuperberg and Eric Samperton. Coloring invariants of knots and links are often intractable. Manuscript.
- 18 Marc Lackenby. A polynomial upper bound on Reidemeister moves. *Ann. Math. (2)*, 182(2):491–564, 2015. doi:10.4007/annals.2015.182.2.3.

- 19 Marc Lackenby. The efficient certification of knottedness and Thurston norm, 2016. [arXiv:1604.00290](#).
- 20 Marc Lackenby. Elementary knot theory. In *Lectures on Geometry (Clay Lecture Notes)*. Oxford University Press, 2017.
- 21 Marc Lackenby. Some conditionally hard problems on links and 3-manifolds. *Discrete & Computational Geometry*, 58(3):580–595, 2017. doi:10.1007/s00454-017-9905-8.
- 22 Adam Simon Levine. Slicing mixed Bing–Whitehead doubles. *Journal of Topology*, 5(3):713–726, 2012. doi:10.1112/jtopol/jts019.
- 23 Dale Rolfsen. *Knots and links*, volume 7 of *Mathematics Lecture Series*. Publish or Perish, Inc., Houston, TX, 1990. Corrected reprint of the 1976 original.
- 24 Eric Samperton. Computational Complexity of Enumerative 3-Manifold Invariants. *arXiv preprint*, 2018. [arXiv:1805.09275](#).
- 25 Tetsuo Shibuya. Some relations among various numerical invariants for links. *Osaka J. Math.*, 11:313–322, 1974. URL: <http://0-projecteuclid.org.library.uark.edu/euclid.ojm/1200757391>.



# On Grids in Point-Line Arrangements in the Plane

Mozhgan Mirzaei

Department of Mathematics, University of California at San Diego, La Jolla, CA, 92093 USA  
momirzae@ucsd.edu

Andrew Suk

Department of Mathematics, University of California at San Diego, La Jolla, CA, 92093 USA  
asuk@ucsd.edu

---

## Abstract

The famous Szemerédi-Trotter theorem states that any arrangement of  $n$  points and  $n$  lines in the plane determines  $O(n^{4/3})$  incidences, and this bound is tight. In this paper, we prove the following Turán-type result for point-line incidence. Let  $\mathcal{L}_a$  and  $\mathcal{L}_b$  be two sets of  $t$  lines in the plane and let  $P = \{\ell_a \cap \ell_b : \ell_a \in \mathcal{L}_a, \ell_b \in \mathcal{L}_b\}$  be the set of intersection points between  $\mathcal{L}_a$  and  $\mathcal{L}_b$ . We say that  $(P, \mathcal{L}_a \cup \mathcal{L}_b)$  forms a *natural  $t \times t$  grid* if  $|P| = t^2$ , and  $\text{conv}(P)$  does not contain the intersection point of some two lines in  $\mathcal{L}_a$  and does not contain the intersection point of some two lines in  $\mathcal{L}_b$ . For fixed  $t > 1$ , we show that any arrangement of  $n$  points and  $n$  lines in the plane that does not contain a natural  $t \times t$  grid determines  $O(n^{4/3 - \varepsilon})$  incidences, where  $\varepsilon = \varepsilon(t) > 0$ . We also provide a construction of  $n$  points and  $n$  lines in the plane that does not contain a natural  $2 \times 2$  grid and determines at least  $\Omega(n^{1 + \frac{1}{14}})$  incidences.

**2012 ACM Subject Classification** Mathematics of computing → Combinatoric problems

**Keywords and phrases** Szemerédi-Trotter Theorem, Grids, Sidon sets

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.50

**Funding** *Mozhgan Mirzaei*: Supported by NSF grant DMS-1800746.

*Andrew Suk*: Supported by an NSF CAREER award and an Alfred Sloan Fellowship.

## 1 Introduction

Given a finite set  $P$  of points in the plane and a finite set  $\mathcal{L}$  of lines in the plane, let  $I(P, \mathcal{L}) = \{(p, \ell) \in P \times \mathcal{L} : p \in \ell\}$  be the set of incidences between  $P$  and  $\mathcal{L}$ . The *incidence graph* of  $(P, \mathcal{L})$  is the bipartite graph  $G = (P \cup \mathcal{L}, I)$ , with vertex parts  $P$  and  $\mathcal{L}$ , and  $E(G) = I(P, \mathcal{L})$ . If  $|P| = m$  and  $|\mathcal{L}| = n$ , then the celebrated theorem of Szemerédi and Trotter [16] states that

$$|I(P, \mathcal{L})| \leq O(m^{2/3}n^{2/3} + m + n). \quad (1.1)$$

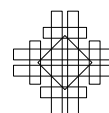
Moreover, this bound is tight which can be seen by taking the  $\sqrt{m} \times \sqrt{m}$  integer lattice and bundles of parallel “rich” lines (see [13]). It is widely believed that the extremal configurations maximizing the number of incidences between  $m$  points and  $n$  lines in the plane exhibit some kind of lattice structure. The main goal of this paper is to show that such extremal configurations must contain large *natural grids*.

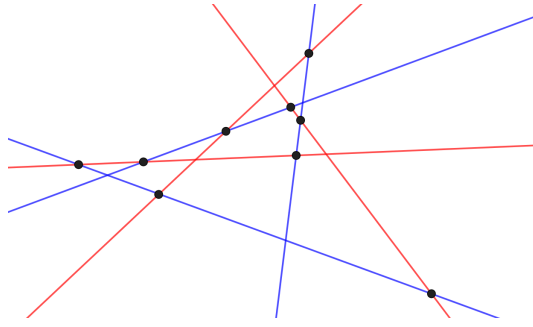
Let  $P$  and  $P_0$  (respectively,  $\mathcal{L}$  and  $\mathcal{L}_0$ ) be two sets of points (respectively, lines) in the plane. We say that the pairs  $(P, \mathcal{L})$  and  $(P_0, \mathcal{L}_0)$  are *isomorphic* if their incidence graphs are isomorphic. Solymosi made the following conjecture (see page 291 in [2]).

► **Conjecture 1.1.** *For any set of points  $P_0$  and for any set of lines  $\mathcal{L}_0$  in the plane, the maximum number of incidences between  $n$  points and  $n$  lines in the plane containing no subconfiguration isomorphic to  $(P_0, \mathcal{L}_0)$  is  $o(n^{4/3})$ .*



© Mozhgan Mirzaei and Andrew Suk;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 50; pp. 50:1–50:11  
Leibniz International Proceedings in Informatics  
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** An example with  $|\mathcal{L}_a| = |\mathcal{L}_b| = 3$  and  $|P| = 9$  in Theorem 1.3.

In [15], Solymosi proved this conjecture in the special case that  $P_0$  is a fixed set of points in the plane, no three of which are on a line, and  $\mathcal{L}_0$  consists of all of their connecting lines. However, it is not known if such configurations satisfy the following stronger conjecture.

► **Conjecture 1.2.** *For any set of points  $P_0$  and for any set of lines  $\mathcal{L}_0$  in the plane, there is a constant  $\varepsilon = \varepsilon(P_0, \mathcal{L}_0)$ , such that the maximum number of incidences between  $n$  points and  $n$  lines in the plane containing no subconfiguration isomorphic to  $(P_0, \mathcal{L}_0)$  is  $O(n^{4/3-\varepsilon})$ .*

Our first theorem is the following.

► **Theorem 1.3.** *For fixed  $t > 1$ , let  $\mathcal{L}_a$  and  $\mathcal{L}_b$  be two sets of  $t$  lines in the plane, and let  $P_0 = \{\ell_a \cap \ell_b : \ell_a \in \mathcal{L}_a, \ell_b \in \mathcal{L}_b\}$  such that  $|P_0| = t^2$ . Then there is a constant  $c = c(t)$  such that any arrangement of  $m$  points and  $n$  lines in the plane that does not contain a subconfiguration isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$  determines at most  $c(m^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}} + m^{1+\frac{1}{6t-3}} + n)$  incidences.*

See the Figure 1. As an immediate corollary, we prove Conjecture 1.2 in the following special case.

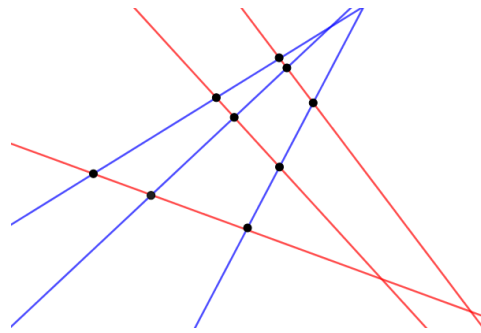
► **Corollary 1.4.** *For fixed  $t > 1$ , let  $\mathcal{L}_a$  and  $\mathcal{L}_b$  be two sets of  $t$  lines in the plane, and let  $P_0 = \{\ell_a \cap \ell_b : \ell_a \in \mathcal{L}_a, \ell_b \in \mathcal{L}_b\}$ . If  $|P_0| = t^2$ , then any arrangement of  $n$  points and  $n$  lines in the plane that does not contain a subconfiguration isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$  determines at most  $O(n^{\frac{4}{3}-\frac{1}{9t-6}})$  incidences.*

In the other direction, we prove the following.

► **Theorem 1.5.** *Let  $\mathcal{L}_a$  and  $\mathcal{L}_b$  be two sets of 2 lines in the plane, and let  $P_0 = \{\ell_a \cap \ell_b : \ell_1 \in \mathcal{L}_a, \ell_b \in \mathcal{L}_b\}$  such that  $|P_0| = 4$ . For  $n > 1$ , there exists an arrangement of  $n$  points and  $n$  lines in the plane that does not contain a subconfiguration isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$ , and determines at least  $\Omega(n^{1+\frac{1}{14}})$  incidences.*

Given two sets  $\mathcal{L}_a$  and  $\mathcal{L}_b$  of  $t$  lines in the plane, and the point set  $P_0 = \{\ell_a \cap \ell_b : \ell_a \in \mathcal{L}_a, \ell_b \in \mathcal{L}_b\}$ , we say that  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$  forms a *natural  $t \times t$  grid* if  $|P_0| = t^2$ , and the convex hull of  $P_0$ ,  $\text{conv}(P_0)$ , does not contain the intersection point of any two lines in  $\mathcal{L}_a$  and does not contain the intersection point of any two lines in  $\mathcal{L}_b$ . See Figure 2.

► **Theorem 1.6.** *For fixed  $t > 1$ , there is a constant  $\varepsilon = \varepsilon(t)$ , such that any arrangement of  $n$  points and  $n$  lines in the plane that does not contain a natural  $t \times t$  grid determines at most  $O(n^{\frac{4}{3}-\varepsilon})$  incidences.*



■ **Figure 2** An example of a natural  $3 \times 3$  grid.

Let us remark that  $\varepsilon = \Omega(1/t^2)$  in Theorem 1.6, and can be easily generalized to the off-balanced setting of  $m$  points and  $n$  lines.

We systemically omit floor and ceiling signs whenever they are not crucial for the sake of clarity of our presentation. All logarithms are assumed to be base 2. For  $N > 0$ , we let  $[N] = \{1, \dots, N\}$ .

## 2 Proof of Theorem 1.3

In this section we will prove Theorem 1.3. We first list several results that we will use. The first lemma is a classic result in graph theory.

► **Lemma 2.1** (Kővari-Sós-Turán [10]). *Let  $G = (V, E)$  be a graph that does not contain a complete bipartite graph  $K_{r,s}$  ( $1 \leq r \leq s$ ) as a subgraph. Then  $|E| \leq c_s |V|^{2-\frac{1}{r}}$ , where  $c_s > 0$  is constant which only depends on  $s$ .*

The next lemma we will use is a partitioning tool in discrete geometry known as *simplicial partitions*. We will use the dual version which requires the following definition. Let  $\mathcal{L}$  be a set of lines in the plane. We say that a point  $p$  *crosses*  $\mathcal{L}$  if it is incident to at least one member of  $\mathcal{L}$ , but not incident to all members in  $\mathcal{L}$ .

► **Lemma 2.2** (Matousek [12]). *Let  $\mathcal{L}$  be a set of  $n$  lines in the plane and let  $r$  be a parameter such that  $1 < r < n$ . Then there is a partition on  $\mathcal{L} = \mathcal{L}_1 \cup \dots \cup \mathcal{L}_r$  into  $r$  parts, where  $\frac{n}{2r} \leq |\mathcal{L}_i| \leq \frac{2n}{r}$ , such that any point  $p \in \mathbb{R}^2$  crosses at most  $O(\sqrt{r})$  parts  $\mathcal{L}_i$ .*

**Proof of Theorem 1.3.** Set  $t \geq 2$ . Let  $P$  be a set of  $m$  points in the plane and let  $\mathcal{L}$  be a set of  $n$  lines in the plane such that  $(P, \mathcal{L})$  does not contain a subconfiguration isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$ .

If  $n \geq m^2/100$ , then (1.1) implies that  $|I(P, \mathcal{L})| = O(n)$  and we are done. Likewise, if  $n \leq m^{\frac{t}{2t-1}}$ , then (1.1) implies that  $|I(P, \mathcal{L})| = O(m^{1+\frac{1}{6t-3}})$  and we are done. Therefore, let us assume  $m^{\frac{t}{2t-1}} < n < m^2/100$ . In what follows, we will show that  $|I(P, \mathcal{L})| = O(m^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}})$ . For sake of contradiction, suppose that  $I(P, \mathcal{L}) \geq cm^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}}$ , where  $c$  is a large constant depending on  $t$  that will be determined later.

Set  $r = \lceil 10n^{\frac{4t-2}{3t-2}}/m^{\frac{2t}{3t-2}} \rceil$ . Let us remark that  $1 < r < n/10$  since we are assuming  $m^{\frac{t}{2t-1}} < n < m^2/100$ . We apply Lemma 2.2 with parameter  $r$  to  $\mathcal{L}$ , and obtain the partition  $\mathcal{L} = \mathcal{L}_1 \cup \dots \cup \mathcal{L}_r$  with the properties described above. Note that  $|\mathcal{L}_i| > 1$ . Let  $G$  be the incidence graph of  $(P, \mathcal{L})$ . For  $p \in P$ , consider the set of lines in  $\mathcal{L}_i$ . If  $p$  is incident to exactly one line in  $\mathcal{L}_i$ , then delete the corresponding edge in the incidence graph  $G$ . After performing



## 50:4 On Grids in Point-Line Arrangements in the Plane

this operation between each point  $p \in P$  and each part  $\mathcal{L}_i$ , by Lemma 2.2, we have deleted at most  $c_1 m \sqrt{r}$  edges in  $G$ , where  $c_1$  is an absolute constant. By setting  $c$  sufficiently large, we have

$$c_1 m \sqrt{r} = \sqrt{10} c_1 m^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}} < (c/2) m^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}}.$$

Therefore, there are at least  $(c/2) m^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}}$  edges remaining in  $G$ . By the pigeonhole principle, there is a part  $\mathcal{L}_i$  such that the number of edges between  $P$  and  $\mathcal{L}_i$  in  $G$  is at least

$$\frac{cm^{\frac{2t-2}{3t-2}} n^{\frac{2t-1}{3t-2}}}{2r} = \frac{cm^{\frac{4t-2}{3t-2}}}{20n^{\frac{2t-1}{3t-2}}}.$$

Hence, every point  $p \in P$  has either 0 or at least 2 neighbors in  $\mathcal{L}_i$  in  $G$ . We claim that  $(P, \mathcal{L}_i)$  contains a subconfiguration isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$ . To see this, let us construct a graph  $H = (\mathcal{L}_i, E)$  as follows. Set  $V(H) = \mathcal{L}_i$ . Let  $Q = \{q_1, \dots, q_w\} \subset P$  be the set of points in  $P$  that have at least two neighbors in  $\mathcal{L}_i$  in the graph  $G$ . For  $q_j \in Q$ , consider the set of lines  $\{\ell_1, \dots, \ell_s\}$  from  $\mathcal{L}_i$  incident to  $q_j$ , such that  $\{\ell_1, \dots, \ell_s\}$  appears in clockwise order. Then we define  $E_j \subset \binom{\mathcal{L}_i}{2}$  to be a matching on  $\{\ell_1, \dots, \ell_s\}$ , where

$$E_j = \begin{cases} \{(\ell_1, \ell_2), (\ell_3, \ell_4), \dots, (\ell_{s-1}, \ell_s)\} & \text{if } s \text{ is even.} \\ \{(\ell_1, \ell_2), (\ell_3, \ell_4), \dots, (\ell_{s-2}, \ell_{s-1})\} & \text{if } s \text{ is odd.} \end{cases}$$

Set  $E(H) = E_1 \cup E_2 \cup \dots \cup E_w$ . Note that  $E_j$  and  $E_k$  are disjoint, since no two points are contained in two lines. Since  $|E_j| \geq 1$ , we have

$$|E(H)| \geq \frac{cm^{\frac{4t-2}{3t-2}}}{60n^{\frac{2t-1}{3t-2}}}.$$

Since

$$|V(H)| = |\mathcal{L}_i| \leq \frac{m^{\frac{2t}{3t-2}}}{5n^{\frac{t}{3t-2}}},$$

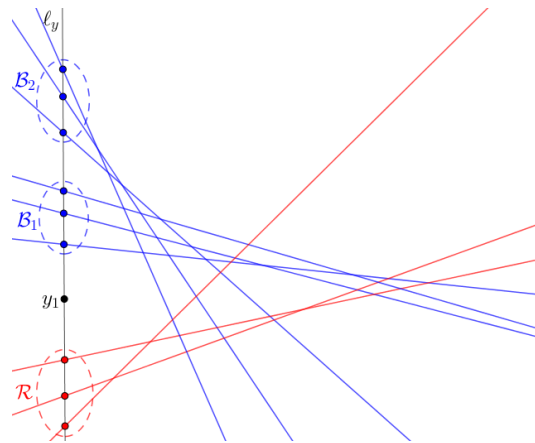
this implies

$$|E(H)| \geq \frac{c}{60 \cdot 25} (V(H))^{2-\frac{1}{t}}.$$

By setting  $c = c(t)$  to be sufficiently large, Lemma 2.1 implies that  $H$  contains a copy of  $K_{t,t}$ . Let  $\mathcal{L}'_1, \mathcal{L}'_2 \subset \mathcal{L}_i$  correspond to the vertices of this  $K_{t,t}$  in  $H$ , and let  $P' = \{\ell_1 \cap \ell_2 \in P : \ell_1 \in \mathcal{L}'_1, \ell_2 \in \mathcal{L}'_2\}$ . We claim that  $(P', \mathcal{L}'_1 \cup \mathcal{L}'_2)$  is isomorphic to  $(P_0, \mathcal{L}_a \cup \mathcal{L}_b)$ . It suffices to show that  $|P'| = t^2$ . For the sake of contradiction, suppose  $p \in \ell_1 \cap \ell_2 \cap \ell_3$ , where  $\ell_1, \ell_2 \in \mathcal{L}'_1$  and  $\ell_3 \in \mathcal{L}'_2$ . This would imply  $(\ell_1, \ell_3), (\ell_2, \ell_3) \in E_j$  for some  $j$  which contradicts the fact that  $E_j \subset \binom{\mathcal{L}_i}{2}$  is a matching. Same argument follows if  $\ell_1 \in \mathcal{L}'_1$  and  $\ell_2, \ell_3 \in \mathcal{L}'_2$ . This completes the proof of Theorem 1.3.  $\blacktriangleleft$

### 3 Natural Grids

Given a set of  $n$  points  $P$  and a set of  $n$  lines  $\mathcal{L}$  in the plane, if  $|I(P, \mathcal{L})| \geq cn^{\frac{4}{3} - \frac{1}{9k-6}}$ , where  $c$  is a sufficiently large constant depending on  $k$ , then Corollary 1.4 implies that there are two sets of  $k$  lines such that each pair of them from different sets intersects at a unique point in  $P$ . Therefore, Theorem 1.6 follows by combining Theorem 1.3 with the following lemma.



■ **Figure 3** Sets  $\mathcal{R}, \mathcal{B}_1, \mathcal{B}_2$  in the proof of Lemma 3.1.

► **Lemma 3.1.** *There is a natural number  $c$  such that the following holds. Let  $\mathcal{B}$  be a set of  $ct^2$  blue lines in the plane, and let  $\mathcal{R}$  be a set of  $ct^2$  red lines in the plane such that for  $P = \{\ell_1 \cap \ell_2 : \ell_1 \in \mathcal{B}, \ell_2 \in \mathcal{R}\}$  we have  $|P| = c^2t^4$ . Then  $(P, \mathcal{B} \cup \mathcal{R})$  contains a natural  $t \times t$  grid.*

To prove Lemma 3.1, we will need the following lemma which is an immediate consequence of Dilworth’s Theorem.

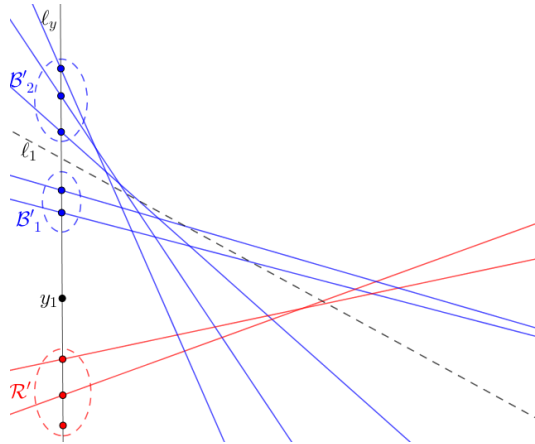
► **Lemma 3.2.** *For  $n > 0$ , let  $\mathcal{L}$  be a set of  $n^2$  lines in the plane, such that no two members intersect the same point on the  $y$ -axis. Then there is a subset  $\mathcal{L}' \subset \mathcal{L}$  of size  $n$  such that the intersection point of any two members in  $\mathcal{L}'$  lies to the left of the  $y$ -axis, or the intersection point of any two members in  $\mathcal{L}'$  lies to the right of the  $y$ -axis.*

**Proof.** Let us order the elements in  $\mathcal{L} = \{\ell_1, \dots, \ell_{n^2}\}$  from bottom to top according to their  $y$ -intercept. By Dilworth’s Theorem [5],  $\mathcal{L}$  contains a subsequence of  $n$  lines whose slopes are either increasing or decreasing. In the first case, all intersection points are to the left of the  $y$ -axis, and in the latter case, all intersection points are to the right of the  $y$ -axis. ◀

**Proof of Lemma 3.1.** Let  $(P, \mathcal{B} \cup \mathcal{R})$  be as described above, and let  $\ell_y$  be the  $y$ -axis. Without loss of generality, we can assume that all lines in  $\mathcal{B} \cup \mathcal{R}$  are not vertical, and the intersection point of any two lines in  $\mathcal{B} \cup \mathcal{R}$  lies to the right of  $\ell_y$ . Moreover, we can assume that no two lines intersect at the same point on  $\ell_y$ .

We start by finding a point  $y_1 \in \ell_y$  such that at least  $|\mathcal{B}|/2$  blue lines in  $\mathcal{B}$  intersect  $\ell_y$  on one side of the point  $y_1$  (along  $\ell_y$ ) and at least  $|\mathcal{R}|/2$  red lines in  $\mathcal{R}$  intersect  $\ell_y$  on the other side. This can be done by sweeping the point  $y_1$  along  $\ell_y$  from bottom to top until  $ct^2/2$  lines of the first color, say red, intersect  $\ell_y$  below  $y_1$ . We then have at least  $ct^2/2$  blue lines intersecting  $\ell_y$  above  $y_1$ . Discard all red lines in  $\mathcal{R}$  that intersect  $\ell_y$  above  $y_1$ , and discard all blue lines in  $\mathcal{B}$  that intersect  $\ell_y$  below  $y_1$ . Hence,  $|\mathcal{B}| \geq ct^2/2$ .

Set  $s = \lfloor ct^2/4 \rfloor$ . For the remaining lines in  $\mathcal{B}$ , let  $\mathcal{B} = \{b_1, \dots, b_{2s}\}$ , where the elements of  $\mathcal{B}$  are ordered in the order they cross  $\ell_y$ , from bottom to top. We partition  $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$  into two parts, where  $\mathcal{B}_1 = \{b_1, \dots, b_s\}$  and  $\mathcal{B}_2 = \{b_{s+1}, \dots, b_{2s}\}$ . By applying an affine transformation, we can assume all lines in  $\mathcal{R}$  have positive slope and all lines in  $\mathcal{B}_1 \cup \mathcal{B}_2$  have negative slope. See Figure 3.



■ **Figure 4** An example for the line  $\ell_1$ .

Let us define a 3-partite 3-uniform hypergraph  $H = (\mathcal{R} \cup \mathcal{B}_1 \cup \mathcal{B}_2, E)$ , whose vertex parts are  $\mathcal{R}, \mathcal{B}_1, \mathcal{B}_2$ , and  $(r, b_i, b_j) \in \mathcal{R} \times \mathcal{B}_1 \times \mathcal{B}_2$  is an edge in  $H$  if and only if the intersection point  $p = b_i \cap b_j$  lies above the line  $r$ . Note, if  $b_i$  and  $b_j$  are parallel, then  $(r, b_i, b_j) \notin E$ . Then a result of Fox et al. on semi-algebraic hypergraphs implies the following (see also [3] and [9]).

► **Lemma 3.3** (Fox et al. [8], Theorem 8.1). *There exists a positive constant  $\alpha$  such that the following holds. In the hypergraph above, there are subsets  $\mathcal{R}' \subseteq \mathcal{R}, \mathcal{B}'_1 \subseteq \mathcal{B}_1, \mathcal{B}'_2 \subseteq \mathcal{B}_2$ , where  $|\mathcal{R}'| \geq \alpha|\mathcal{R}|, |\mathcal{B}'_1| \geq \alpha|\mathcal{B}_1|, |\mathcal{B}'_2| \geq \alpha|\mathcal{B}_2|$ , such that either  $\mathcal{R}' \times \mathcal{B}'_1 \times \mathcal{B}'_2 \subseteq E$ , or  $(\mathcal{R}' \times \mathcal{B}'_1 \times \mathcal{B}'_2) \cap E = \emptyset$ .*

We apply Lemma 3.3 to  $H$  and obtain subsets  $\mathcal{R}', \mathcal{B}'_1, \mathcal{B}'_2$  with the properties described above. Without loss of generality, we can assume that  $\mathcal{R}' \times \mathcal{B}'_1 \times \mathcal{B}'_2 \subseteq E$ , since a symmetric argument would follow otherwise. Let  $\ell_1$  be a line in the plane such that the following holds.

1. The slope of  $\ell_1$  is negative.
2. All intersection points between  $\mathcal{R}'$  and  $\mathcal{B}'_1$  lie above  $\ell_1$ .
3. All intersection points between  $\mathcal{R}'$  and  $\mathcal{B}'_2$  lie below  $\ell_1$ .

See Figure 4.

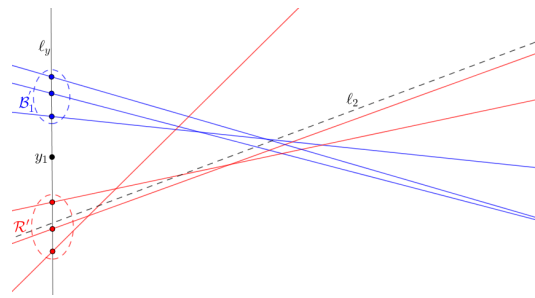
► **Observation 3.4.** *Line  $\ell_1$  defined above exists.*

**Proof.** Let  $U$  be the upper envelope of the arrangement  $\bigcup_{\ell \in \mathcal{R}'} \ell$ , that is,  $U$  is the closure of all points that lie on exactly one line of  $\mathcal{R}'$  and strictly above exactly the  $|\mathcal{R}'| - 1$  lines in  $\mathcal{R}'$ .

Let  $P_1$  be the set of intersection points between the lines in  $\mathcal{B}'_1$  with  $U$ . Likewise, we define  $P_2$  to be the set of intersection points between the lines in  $\mathcal{B}'_2$  with  $U$ . Since  $U$  is  $x$ -monotone and convex the set  $P_2$  lies to the left of the set  $P_1$ . Then the line  $\ell_1$  that intersects  $U$  between  $P_1$  and  $P_2$  and intersects  $\ell_y$  between  $\mathcal{B}'_1$  and  $\mathcal{B}'_2$  satisfies the conditions above. ◀

Now we apply Lemma 3.2 to  $\mathcal{R}'$  with respect to the line  $\ell_1$ , to obtain  $\sqrt{ac/2} \cdot t$  members in  $\mathcal{R}'$  such that every pair of them intersects on one side of  $\ell_1$ . Discard all other members in  $\mathcal{R}'$ . Without loss of generality, we can assume that all intersection points between any two members in  $\mathcal{R}'$  lie below  $\ell_1$ , since a symmetric argument would follow otherwise. We now discard the set  $\mathcal{B}'_2$ .

Notice that the order in which the lines in  $\mathcal{R}'$  cross  $b \in \mathcal{B}'_1$  will be the same for any line  $b \in \mathcal{B}'_1$ . Therefore, we order the elements in  $\mathcal{R}' = \{r_1, \dots, r_m\}$  with respect to this ordering, from left to right, where  $m = \lceil \sqrt{ac/2} \cdot t \rceil$ . We define  $\ell_2$  to be the line obtained by slightly perturbing the line  $r_{\lfloor m/2 \rfloor}$  such that:



■ **Figure 5** An example for the line  $\ell_2$ .

1. The slope of  $\ell_2$  is positive.
2. All intersection points between  $\mathcal{B}'_1$  and  $\{r_1, \dots, r_{\lfloor m/2 \rfloor}\}$  lie above  $\ell_2$ .
3. All intersection points between  $\mathcal{B}'_1$  and  $\{r_{\lfloor m/2 \rfloor + 1}, \dots, r_m\}$  lie below  $\ell_2$ .

See the Figure 5.

Finally, we apply Lemma 3.2 to  $\mathcal{B}'_1$  with respect to the line  $\ell_2$ , to obtain at least  $\sqrt{\alpha c} \cdot t/2$  members in  $\mathcal{B}'_1$  with the property that any two of them intersect on one side of  $\ell_2$ . Without loss of generality, we can assume that any two such lines intersect below  $\ell_2$  since a symmetric argument would follow. Set  $\mathcal{B}^* \subset \mathcal{B}'_1$  to be these set of lines. Then  $\mathcal{B}^* \cup \{r_1, \dots, r_{\lfloor m/2 \rfloor}\}$  and their intersection points form a natural grid. By setting  $c = c(t)$  to be sufficiently large, we obtain a natural  $t \times t$  grid. ◀

#### 4 Lower Bound Construction

In this section, we will prove Theorem 1.5. First, let us recall the definitions of Sidon and  $k$ -fold Sidon sets.

Let  $A$  be a finite set of positive integers. Then  $A$  is a *Sidon set* if the sum of all pairs are distinct, that is, the equation  $x + y = u + v$  has no solutions with  $x, y, u, v \in A$ , except for trivial solutions given by  $u = x, y = v$  and  $x = v, y = u$ . We define  $s(N)$  to be the size of the largest Sidon set  $A \subset \{1, \dots, N\}$ . Erdős and Turán proved the following.

▶ **Lemma 4.1** (See [7] and [14]). *For  $N > 1$ , we have  $s(N) = \Theta(\sqrt{N})$ .*

Let us now consider a more general equation. Let  $u_1, \dots, u_4$  be integers such that  $u_1 + u_2 + u_3 + u_4 = 0$ , and consider the equation

$$u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 = 0. \tag{4.1}$$

We are interested in solutions to (4.1) with  $x_1, x_2, x_3, x_4 \in \mathbb{Z}$ . Suppose  $(x_1, x_2, x_3, x_4) = (a_1, a_2, a_3, a_4)$  is an integer solution to (4.1). Let  $d \leq 4$  be the number of distinct integers in the set  $\{a_1, a_2, a_3, a_4\}$ . Then we have a partition on the indices

$$\{1, 2, 3, 4\} = T_1 \cup \dots \cup T_d,$$

where  $i$  and  $j$  lie in the same part  $T_\nu$  if and only if  $x_i = x_j$ . We call  $(a_1, a_2, a_3, a_4)$  a *trivial* solution to (4.1) if

$$\sum_{i \in T_\nu} u_i = 0, \quad \nu = 1, \dots, d.$$

Otherwise, we will call  $(a_1, a_2, a_3, a_4)$  a *nontrivial* solution to (4.1).

50:8 On Grids in Point-Line Arrangements in the Plane

In [11], Lazebnik and Verstraëte introduced  $k$ -fold Sidon sets which are defined as follows. Let  $k$  be a positive integer. A set  $A \subset \mathbb{N}$  is a  $k$ -fold Sidon set if each equation of the form

$$u_1x_1 + u_2x_2 + u_3x_3 + u_4x_4 = 0, \tag{4.2}$$

where  $|u_i| \leq k$  and  $u_1 + \dots + u_4 = 0$ , has no nontrivial solutions with  $x_1, x_2, x_3, x_4 \in A$ . Let  $r(k, N)$  be the size of the largest  $k$ -fold Sidon set  $A \subset \{1, \dots, N\}$ .

► **Lemma 4.2.** *There is an infinite sequence  $1 = a_1 < a_2 < \dots$  of integers such that*

$$a_m \leq 2^8 k^4 m^3,$$

and the system of equations (4.2) has no nontrivial solutions in the set  $A = \{a_1, a_2, \dots\}$ . In particular, for integers  $N > k^4 \geq 1$ , we have  $r(k, N) \geq ck^{-4/3}N^{1/3}$ , where  $c$  is a positive constant.

The proof of Lemma 4.2 is a slight modification of the proof of Theorem 2.1 in [14]. For the sake of completeness, we include the proof here.

**Proof.** We put  $a_1 = 1$  and define  $a_m$  recursively. Given  $a_1, \dots, a_{m-1}$ , let  $a_m$  be the smallest positive integer satisfying

$$a_m \neq -\left(\sum_{i \in S} u_i\right)^{-1} \sum_{1 \leq i \leq 4, i \notin S} u_i x_i, \tag{4.3}$$

for every choice  $u_i$  such that  $|u_i| \leq k$ , for every set  $S \subset \{1, \dots, 4\}$  of subscripts such that  $\left(\sum_{i \in S} u_i\right) \neq 0$ , and for every choice of  $x_i \in \{a_1, \dots, a_{m-1}\}$ , where  $i \notin S$ . For a fixed  $S$  with  $|S| = j$ , this excludes  $(m-1)^{4-j}$  numbers. Since  $|u_i| \leq k$ , the total number of excluded integers is at most

$$(2k+1)^4 \sum_{j=1}^3 \binom{4}{j} (m-1)^{4-j} = (2k+1)^4 (m^4 - (m-1)^4 - 1) < 2^8 k^4 m^3.$$

Consequently, we can extend our set by an integer  $a_m \leq 2^8 k^4 m^3$ . This will automatically be different from  $a_1, \dots, a_{m-1}$ , since putting  $x_i = a_j$  for all  $i \notin S$  in (4.3) we get  $a_m \neq a_j$ . It will also satisfy  $a_m > a_{m-1}$  by minimal choice of  $a_{m-1}$ .

We show that the system of equations (4.2) has no nontrivial solutions in the set  $\{a_1, \dots, a_m\}$ . We use induction on  $m$ . The statement is obviously true for  $m = 1$ . We establish it for  $m$  assuming for  $m - 1$ . Suppose that there is a nontrivial solution  $(x_1, x_2, x_3, x_4)$  to (4.2) for some  $u_1, u_2, u_3, u_4$  with the properties described above. Let  $S$  denote the set of those subscripts for which  $x_i = a_m$ . If  $\sum_{i \in S} u_i \neq 0$ , then this contradicts (4.3). If  $\sum_{i \in S} u_i = 0$ , then by replacing each occurrence of  $a_m$  by  $a_1$ , we get another nontrivial solution, which contradicts the induction hypothesis. ◀

For more problems and results on Sidon sets and  $k$ -fold Sidon sets, we refer the interested reader to [11, 14, 4].

We are now ready to prove Theorem 1.5.

**Proof of Theorem 1.5.** We start by applying Lemma 4.1 to obtain a Sidon set  $M \subset [n^{1/7}]$ , such that  $|M| = \Theta(n^{1/14})$ . We then apply Lemma 4.2 with  $k = n^{1/7}$  and  $N = \frac{1}{4}n^{11/14}$ , to obtain a  $k$ -fold Sidon set  $A \subset [N]$  such that

$$|A| \geq cn^{1/14},$$

where  $c$  is defined in Lemma 4.2. Without loss of generality, let us assume  $|A| = cn^{1/14}$ .

Let  $P = \{(i, j) \in \mathbb{Z}^2 : i \in A, 1 \leq j \leq n^{13/14}\}$ , and let  $\mathcal{L}$  be the family of lines in the plane of the form  $y = mx + b$ , where  $m \in M$  and  $b$  is an integer such that  $1 \leq b \leq n^{13/14}/2$ .

Hence, we have

$$|P| = |A| \cdot n^{13/14} = \Theta(n),$$

$$|\mathcal{L}| = |M| \cdot \frac{n^{13/14}}{2} = \Theta(n).$$

Notice that each line in  $\mathcal{L}$  has exactly  $|A| = cn^{1/14}$  points from  $P$  since  $1 \leq b \leq n^{13/14}/2$ . Therefore,

$$|I(P, \mathcal{L})| = |\mathcal{L}||A| = \Theta(n^{1+1/14}).$$

▷ **Claim 4.3.** There are no four distinct lines  $\ell_1, \ell_2, \ell_3, \ell_4 \in \mathcal{L}$  and four distinct points  $p_1, p_2, p_3, p_4 \in P$  such that  $\ell_1 \cap \ell_2 = p_1, \ell_2 \cap \ell_3 = p_2, \ell_3 \cap \ell_4 = p_3, \ell_4 \cap \ell_1 = p_4$ .

*Proof.* For the sake of contradiction, suppose there are four lines  $\ell_1, \ell_2, \ell_3, \ell_4$  and four points  $p_1, p_2, p_3, p_4$  with the properties described above. Let  $\ell_i = m_i x + b_i$  and let  $p_i = (x_i, y_i)$ . Therefore,

$$\begin{aligned} \ell_1 \cap \ell_2 &= p_1 = (x_1, y_1), \\ \ell_2 \cap \ell_3 &= p_2 = (x_2, y_2), \\ \ell_3 \cap \ell_4 &= p_3 = (x_3, y_3), \\ \ell_4 \cap \ell_1 &= p_4 = (x_4, y_4). \end{aligned}$$

Hence,

$$\begin{aligned} p_1 \in \ell_1, \ell_2 &\implies (m_1 - m_2)x_1 + b_1 - b_2 = 0, \\ p_2 \in \ell_2, \ell_3 &\implies (m_2 - m_3)x_2 + b_2 - b_3 = 0, \\ p_3 \in \ell_3, \ell_4 &\implies (m_3 - m_4)x_3 + b_3 - b_4 = 0, \\ p_4 \in \ell_4, \ell_1 &\implies (m_4 - m_1)x_4 + b_4 - b_1 = 0. \end{aligned}$$

By summing up the four equations above, we get

$$(m_1 - m_2)x_1 + (m_2 - m_3)x_2 + (m_3 - m_4)x_3 + (m_4 - m_1)x_4 = 0.$$

By setting  $u_1 = m_1 - m_2, u_2 = m_2 - m_3, u_3 = m_3 - m_4, u_4 = m_4 - m_1$ , we get

$$u_1 x_1 + u_1 x_2 + u_3 x_3 + u_4 x_4 = 0, \tag{4.4}$$

where  $u_1 + u_2 + u_3 + u_4 = 0$  and  $|u_i| \leq n^{1/7}$ . Since  $x_1, \dots, x_4 \in A$ ,  $(x_1, x_2, x_3, x_4)$  must be a trivial solution to (4.4). The proof now falls into the following cases, and let us note that no line in  $\mathcal{L}$  is vertical.

**Case 1.** Suppose  $x_1 = x_2 = x_3 = x_4$ . Then  $\ell_i$  is vertical and we have a contradiction.

**Case 2.** Suppose  $x_1 = x_2 = x_3 \neq x_4$  and  $u_1 + u_2 + u_3 = 0$  and  $u_4 = 0$ . Then  $\ell_1$  and  $\ell_4$  have the same slope which is a contradiction. The same argument follows if  $x_1 = x_2 = x_4 \neq x_3, x_1 = x_3 = x_4 \neq x_2$ , or  $x_2 = x_3 = x_4 \neq x_1$ .

**Case 3.** Suppose  $x_1 = x_2 \neq x_3 = x_4$ ,  $u_1 + u_2 = 0$ , and  $u_3 + u_4 = 0$ . Since  $p_1, p_2 \in \ell_2$  and  $x_1 = x_2$ , this implies that  $\ell_2$  is vertical which is a contradiction. A similar argument follows if  $x_1 = x_4 \neq x_2 = x_3$ ,  $u_1 + u_4 = 0$ , and  $u_2 + u_3 = 0$ .

**Case 4.** Suppose  $x_1 = x_3 \neq x_2 = x_4$ ,  $u_1 + u_3 = 0$ , and  $u_2 + u_4 = 0$ . Then  $u_1 + u_3 = 0$  implies that  $m_1 + m_3 = m_2 + m_4$ . Since  $M$  is a Sidon set, we have either  $m_1 = m_2$  and  $m_3 = m_4$  or  $m_1 = m_4$  and  $m_2 = m_3$ . The first case implies that  $\ell_1$  and  $\ell_2$  are parallel which is a contradiction, and the second case implies that  $\ell_2$  and  $\ell_3$  are parallel, which is again a contradiction.  $\triangleleft$

This completes the proof of Theorem 1.5.  $\blacktriangleleft$

## 5 Concluding Remarks

- An old result of Erdős states that every  $n$ -vertex graph that does not contain a cycle of length  $2k$ , has  $O_k(n^{1+1/k})$  edges. It is known that this bound is tight when  $k = 2, 3$ , and  $5$ , but it is a long standing open problem in extremal graph theory to decide whether or not this upper bound can be improved for other values of  $k$ . Hence, Erdős's upper bound of  $O(n^{5/4})$  when  $k = 4$  implies Theorem 1.3 when  $t = 2$  and  $m = n$ . It would be interesting to see if one can improve the upper bound in Theorem 1.3 when  $t = 2$ . For more problems on cycles in graphs, see [17].
- The proof of Lemma 3.1 is similar to the proof of the main result in [1]. The main difference is that we use the result of Fox et al. [8] instead of the Ham-Sandwich Theorem. We also note that a similar result was established by Dujmović and Langerman (see Theorem 6 in [6]).

---

## References

- 1 Boris Aronov, Paul Erdős, Wayne Goddard, Daniel Kleitman, Michael Klugerman, János Pach, and Leonard J. Schulman. Crossing families. *Combinatorica*, 14(2):127–134, 1994.
- 2 Peter Brass, William O.J. Moser, and János Pach. *Research problems in discrete geometry*. Springer Science & Business Media, 2006.
- 3 Boris Bukh and Alfredo Hubard. Space crossing numbers. *Combin. Probab. Comput.*, 21(3):358–373, 2012.
- 4 Javier Cilleruelo and Craig Timmons.  $k$ -Fold Sidon Sets. *Electron. J. Combin.*, 21(4):P4–12, 2014.
- 5 Robert P Dilworth. A decomposition theorem for partially ordered sets. *Ann. of Math.*, pages 161–166, 1950.
- 6 Vida Dujmović and Stefan Langerman. A Center Transversal Theorem for Hyperplanes and Applications to Graph Drawing. *Discrete Comput. Geom.*, 49(1):74–88, January 2013. doi:10.1007/s00454-012-9464-y.
- 7 Paul Erdős and Pál Turán. On a problem of Sidon in additive number theory, and on some related problems. *J. Lond. Math. Soc. (2)*, 1(4):212–215, 1941.
- 8 Jacob Fox, Mikhail Gromov, Vincent Lafforgue, Assaf Naor, and János Pach. Overlap properties of geometric expanders. *J. Reine Angew. Math.*, 2012(671):49–83, 2012.
- 9 Jacob Fox, János Pach, and Andrew Suk. A polynomial regularity lemma for semialgebraic hypergraphs and its applications in geometry and property testing. *SIAM J. Comput.*, 45(6):2199–2223, 2016.
- 10 Tamás Kovári, Vera Sós, and Pál Turán. On a problem of K. Zarankiewicz. 3(1):50–57, 1954.
- 11 Felix Lazebnik and Jacques Verstraëte. On hypergraphs of girth five. *Electron. J. Combin.*, 10:1–25, 2003.
- 12 Jiří Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8(3):315–334, 1992.



- 13 János Pach and Pankaj K Agarwal. *Combinatorial geometry*, volume 37. John Wiley & Sons, 2011.
- 14 Imre Z Ruzsa. Solving a linear equation in a set of integers I. *Acta Arith.*, 65(3):259–282, 1993.
- 15 József Solymosi. Dense arrangements are locally very dense I. *SIAM J. Discrete Math.*, 20(3):623–627, 2006.
- 16 Endre Szemerédi and William T. Trotter. Extremal problems in discrete geometry. *Combinatorica*, 3(3-4):381–392, 1983.
- 17 Jacques Verstraëte. Extremal problems for cycles in graphs. In *Recent Trends in Combinatorics*, pages 83–116. Springer, 2016.



# On Weak $\epsilon$ -Nets and the Radon Number

Shay Moran

Department of Computer Science, Princeton University, Princeton, USA  
shaym@cs.princeton.edu

Amir Yehudayoff

Department of Mathematics, Techion-IIT, Haifa, Israel  
amir.yehudayoff@gmail.com

---

## Abstract

We show that the Radon number characterizes the existence of weak nets in separable convexity spaces (an abstraction of the Euclidean notion of convexity). The construction of weak nets when the Radon number is finite is based on Helly's property and on metric properties of VC classes. The lower bound on the size of weak nets when the Radon number is large relies on the chromatic number of the Kneser graph. As an application, we prove an amplification result for weak  $\epsilon$ -nets.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Mathematics of computing  $\rightarrow$  Combinatoric problems

**Keywords and phrases** abstract convexity, weak epsilon nets, Radon number, VC dimension, Haussler packing lemma, Kneser graphs

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.51

**Funding** *Shay Moran*: supported by the Simons Foundation and the NSF; part of this project was carried while the author was at the Institute for Advanced Study and was supported by the National Science Foundation under agreement No. CCF-1412958.

*Amir Yehudayoff*: Research supported by ISF grant 1162/15. Part of this work was done while the author was visiting the Simons Institute for the Theory of Computing.

**Acknowledgements** We thank Noga Alon, Yuval Dagan, and Gil Kalai for helpful conversations. We also thank the anonymous reviewers assigned by SoCG '19 for their helpful comments which improved the presentation of this work.

## 1 Introduction

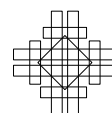
Weak and strong  $\epsilon$ -nets were defined by Haussler and Welzl as a tool for fast processing of geometric range queries [20]. They have been consequently studied in many areas, including computational geometry, combinatorics, and machine learning, and they were used in many algorithmic applications, including range searching and geometric optimization.

An  $\epsilon$ -net is a set that pierces all large sets in a given family of sets. Formally, let  $\mu$  be a probability distribution over a domain  $X$  and let<sup>1</sup>  $C \subseteq 2^X$  be a family of sets. A subset  $S$  of  $X$  is called a *weak  $\epsilon$ -net* for  $C$  over  $\mu$  if  $S \cap c \neq \emptyset$  for every  $c \in C$  with  $\mu(c) \geq \epsilon$ . A subset  $S$  is called a *strong  $\epsilon$ -net* if in addition  $S$  is contained in the support of  $\mu$ . We say that  $C$  has weak/strong  $\epsilon$ -nets of size  $\beta = \beta(C, \epsilon)$  if for every distribution  $\mu$  there is a weak/strong  $\epsilon$ -net for  $C$  over  $\mu$  of size at most  $\beta$  (we stress that  $\beta$  may depend on  $\epsilon$ , but not on  $\mu$ ).

To illustrate the difference between weak and strong nets, consider the uniform distribution on  $n$  points on the unit circle in the plane  $X = \mathbb{R}^2$ , and the family  $C$  to be all convex hulls of subsets of these  $n$  points. Any strong  $\epsilon$ -net must contain at least  $(1 - \epsilon)n$  points, but there

---

<sup>1</sup> Here and below we assume that all sets considered are measurable.



are weak  $\epsilon$ -nets of size  $O(\alpha(\epsilon)/\epsilon)$ , where  $\alpha(\cdot)$  is the inverse Ackermann function [5]; using points inside the unit disc allows to use significantly less points. In general, weak nets may be much smaller than strong nets.

The main question we address is under what conditions do weak nets exist. This question for strong  $\epsilon$ -nets is fairly well understood; the fundamental theorem of statistical learning, which shows that the VC dimension characterizes PAC learnability, also shows that the VC dimension characterizes the existence of strong nets (see [20, 29] and references within). We show that the Radon number characterizes the existence of weak nets in a pretty general setting (viz. separable convexity spaces).

## Weak nets

Weak  $\epsilon$ -nets were mostly studied in the context of discrete and convex geometry, where they are related to several deep phenomena. For example, Alon and Kleitman [6] used weak nets in their famous solution of the  $(p, q)$ -conjecture by Hadwiger and Debrunner [16].

Barany, Furedi, and Lovasz [8] showed that convex sets in the plane admit weak nets, and Alon, Barany, Furedi, and Kleitman [3] established a bound of  $O(1/\epsilon^2)$  on their size. Recently Rubin improved this bound to  $O(1/\epsilon^{3/2+\gamma})$ , where  $\gamma > 0$  is arbitrarily small [28]. Alon, Barany, Furedi, and Kleitman [3] extended the existence of weak nets for convex sets to all dimensions  $d$ ; there are weak  $\epsilon$ -nets of size at most roughly  $(1/\epsilon)^{d+1}$ . Their proof relies on several results from convex geometry, like Tverberg's theorem and the colorful Caratheodory theorem. Chazelle, Edelsbrunner, Grigni, Guibas, Sharir, and Welzl [11] later improved the bound to at most roughly  $(1/\epsilon)^d$ . Overall, there are at least three different constructions of weak nets for convex sets. Matousek [26] showed that any weak  $\epsilon$ -net for convex sets in  $\mathbb{R}^d$  must contain at least  $\Omega(\exp(\sqrt{d/2}))$  points for  $\epsilon \leq 1/50$ . Later, Alon [2] proved the first lower bound that is superlinear in  $1/\epsilon$ ; this was later improved to  $\Omega((1/\epsilon) \log^d(1/\epsilon))$  by Bukh, Matousek, and Nivasch [10]. Ezra [14] constructed weak  $\epsilon$ -nets for the more restricted class of axis-parallel boxes in  $\mathbb{R}^d$ . Alon, Kalai, Matousek, and Meshulam [4] defined weak nets in an abstract setting, and asked about combinatorial conditions that yield their existence.

We identify that the existence of weak nets follows from a basic combinatorial property of convex sets, Radon's theorem:

► **Theorem 1 (Radon).** *Any set of  $d + 2$  points in  $\mathbb{R}^d$  can be partitioned into two disjoint subsets whose convex hulls intersect.*

We show that this property alone is sufficient and necessary for the existence of weak nets in a general setting, which we describe next.

It is worth mentioning that Radon's theorem also plays a central role in the context of strong nets. Indeed, it implies that the VC dimension of half-spaces in  $\mathbb{R}^d$  is at most  $d + 1$ , which consequently bounds the VC dimension of many geometrically defined classes.

## Convexity spaces

We consider an abstraction of Euclidean convexity that originated in a paper by Levi [24], and defined in the form presented here by Kay and Womble [21]. For a thorough introduction to this subject see the survey by Danzer, Grunbaum, and Klee [13] or the more recent book by van de Vel [30].

A *convexity space* is a pair  $(X, C)$  where  $C \subseteq 2^X$  is a family of subsets that satisfies<sup>2</sup>:

- $\emptyset, X \in C$ .
- $C$  is closed under intersections<sup>3</sup>:  $\cap C' \in C$  for every  $C' \subseteq C$ .

The members of  $C$  are called *convex sets*. The *convex-hull* of a set  $Y \subseteq X$ , denoted by  $\text{conv}(Y) = \text{conv}_C(Y)$ , is the intersection of all convex sets  $c \in C$  that contain  $Y$ . A convex set  $b \in C$  is called a *half-space* if its complement is also convex.

We next define the notion of separability, which is an abstraction of the hyperplane separation theorem (and the more general Hahn-Banach theorem). The convexity space  $(X, C)$  is *separable*, if for every  $c \in C$  and  $x \in X \setminus c$  there exists a half-space  $b \in C$  so that  $c \subseteq b$  and  $x \notin b$ . It can be verified that  $(X, C)$  is separable if and only if every convex set  $c \in C$  is the intersection of all half-spaces containing it. This form of separability, as well as other forms, have been extensively studied (e.g. [15, 18, 17, 22, 12]).

Convexity spaces appear in many contexts in mathematics. For instance, the family of closed subsets in a topological space, the subgroups of a given groups, and the subrings of a ring are all examples<sup>4</sup> of convexity spaces. They are also closely related to the notion of  $\pi$ -systems in probability theory. In Section 1.1 below we discuss a few examples of convexity spaces that arise in algebra and combinatorics.

## Main results

The combinatorial property that characterizes the existence of weak nets is the Radon number [24, 21], which is an abstraction of Radon's theorem: we say that  $C$  *Radon-shatters* a set  $Y \subseteq X$  if for every partition of  $Y$  into two parts  $Y_1, Y_2$  it holds that  $\text{conv}(Y_1) \cap \text{conv}(Y_2) = \emptyset$ . The *Radon number* of  $(X, C)$  is the minimum number  $r$  such that  $C$  does not Radon-shatter any set of size  $r$ . Radon's theorem states that the Radon number of the space of convex sets in  $\mathbb{R}^d$  is at most  $d + 2$ .

► **Theorem 2.** *Let  $(X, C)$  be a finite separable convexity space.*

1. *If the Radon number of  $(X, C)$  is at most  $r$  then it has weak  $\epsilon$ -nets of size at most  $(120r^2/\epsilon)^{4r^2 \ln(1/\epsilon)}$  for every  $\epsilon > 0$ .*
2. *If the Radon number of  $(X, C)$  is more than  $r$  then there is a distribution  $\mu$  over  $X$  such that every  $\frac{1}{4}$ -net for  $C$  over  $\mu$  has size at least  $r/2$ .*

We often refer to a construction of small weak  $\epsilon$ -nets as an *upper bound*, and to a proof that no small weak  $\epsilon$ -nets exists as a *lower bound*. The upper bound in 1 above is quantitatively worse than the one for Euclidean convex sets [11], but holds in a more general setting. We do not know what is the optimal bound in this generality.

A possible interpretation of the upper bound is that the existence of weak nets is not directly related to “geometric” properties of the underlying space (as in the various constructions surveyed above). This is somewhat surprising: consider a family  $C$  and a distribution  $\mu$  such that  $C$  has no strong  $\epsilon$ -net with respect to  $\mu$ ; in order to construct a weak  $\epsilon$ -net we should have a mechanism that suggests “good points” outside the support of  $\mu$ . In Euclidean geometry there are such natural choices, like “center of mass”. We notice that the Radon number provides such a mechanism (see Section 1.2).

<sup>2</sup> Note that van de Vel [30] also requires that the union of an ascending chain of convex sets is convex.

<sup>3</sup> We use the standard notation  $\cap C' = \bigcap_{c \in C'} c$ .

<sup>4</sup> One should sometimes add the empty set in order to satisfy all axioms of a convexity space.

## 51:4 On Weak $\epsilon$ -Nets and the Radon Number

We next discuss conditions that are equivalent to the existence of small  $\epsilon$ -nets. It is convenient to present these equivalences for infinite spaces. Quantitative variants of these statements apply to finite spaces as well.

We use the following standard notion of compactness; a family  $C$  is *compact* if for every  $C' \subseteq C$  so that  $\bigcap C' = \emptyset$  there is a finite  $C'' \subseteq C'$  so that  $\bigcap C'' = \emptyset$ . This condition is satisfied e.g. by closed sets in a compact topological space.

► **Corollary 3 (Equivalences).** *The following are equivalent for a compact separable convexity space  $(X, C)$ :*

1.  $(X, C)$  has a finite Radon number.
2.  $(X, C)$  has weak  $\epsilon$ -nets of finite size for some  $0 < \epsilon < 1/2$ .
3.  $(X, C)$  has weak  $\epsilon$ -nets of finite size for every  $\epsilon > 0$ .

The proof of Corollary 3 appears in Section 4.1. It shows that the role of the Radon number in the existence of weak nets is similar to the role of the VC dimension in the existence of strong nets, at least for separable compact convexity spaces.

In Section 1.4, we provide an example showing that the compactness assumption in Corollary 3 is necessary. We do not know if the separability assumption is necessary.

The implication  $2 \Rightarrow 3$  in Corollary 3 is an amplification statement for the parameter  $\epsilon$  in weak nets. In Section 4.2 we give an example showing that the threshold  $1/2$  in item 2 is sharp for amplification of weak nets:

► **Example 4.** There is a compact separable convexity space that has weak  $\epsilon$ -nets of finite size for every  $\epsilon > 1/2$  but has no weak  $\epsilon$ -nets of finite size for  $\epsilon < \frac{1}{2}$ .

This demonstrates an interesting difference with strong  $\epsilon$ -nets, for which there is no such threshold: if  $C$  has strong  $\epsilon$ -nets for some  $\epsilon < 1$  then it has finite VC dimension, which implies the existence of strong  $\epsilon$ -net for all  $\epsilon > 0$ .

### Organization

In Section 1.1 we provide some examples of convexity spaces. In Section 1.2 we outline the construction that leads to the upper bound in Theorem 2, in Section 1.3 we outline the lower bound in Theorem 2, and in Section 1.4 we provide some examples that demonstrate the necessity of some of our assumptions.

In Section 2 we prove the upper bound, and in Section 3 we prove the lower bound. In Section 4 we prove the characterizations (Corollary 3), and in Section 5 we discuss an extension to convexity spaces that are not necessarily separable or compact (like bounded convex sets in  $\mathbb{R}^d$ ). Finally, in Section 6 we conclude the paper and offer some directions for future research.

### 1.1 Some convexity spaces

We now present a few examples of “non Euclidean” convexity spaces. These examples will be used later on to show that some of our theorems/lemmas are tight in the sense that each premise is necessary. More examples can be found in the book [30].

**Example 1 (power set).** Let  $X$  be a set. Perhaps the simplest convexity space is  $(X, 2^X)$ . Here every convex set is a half-space and therefore this space is separable. When  $X$  is finite, the Radon number of this space is  $|X| + 1$ . When  $X$  is infinite, the Radon number is  $\infty$ .

**Example 2 (subgroups).** Let  $G$  be a group with identity  $e$ . The space  $(G \setminus \{e\}, \{H \setminus \{e\} : H \leq G\})$  of all subgroups of  $G$  (with the identity removed) is a convexity space. Here,  $Y \subseteq G$  is Radon-shattered, if every two disjoint subsets of  $Y$  generate groups whose intersection is  $\{e\}$ .

**Example 3 (cylinders).** Let  $X = \{0, 1\}^n$ , and let the  $C$  be the family of *cylinders*: a set  $c \subseteq \{0, 1\}^n$  is called a cylinder if there exists  $Y \subseteq [n]$ , and  $v \in \{0, 1\}^Y$  such that

$$c = \{u \in \{0, 1\}^n : u|_Y = v\}.$$

The size of  $Y$  is called the co-dimension of the cylinder. This is a separable convexity space. Its half-spaces are the cylinders with co-dimension 1, and its Radon number is  $\Theta(\log n)$ .

**Example 4 (subtrees).** Let  $T = (V, E)$  be a finite tree. Consider the convexity space  $(V, C)$ , where

$$C = \{U \subseteq V : \text{the induced subgraph on } U \text{ is connected}\}.$$

It is a separable convexity space and its Radon number is at most 4. Theorem 2 hence implies the existence of weak nets of size depending only on  $\epsilon$  (in this case there are elementary constructions of  $\epsilon$ -nets of size  $O(1/\epsilon)$ ). This example is a special case of *geodesic convexity* in metric spaces (see [30]).

**Example 5 (convex lattice sets).** Consider the space  $(\mathbb{Z}^d, C)$ , where  $C$  is the family of *convex lattice sets* in  $\mathbb{R}^d$ ; these are sets of the form  $K \cap \mathbb{Z}^d$  for some convex  $K \subseteq \mathbb{R}^d$ .

This is a separable convexity space. Indeed, let  $c = K \cap \mathbb{Z}^d \in C$  and  $x \in \mathbb{Z}^d \setminus c$ . Since  $x \notin c$  it follows that  $x \notin K$ , and therefore there is a half-space in  $\mathbb{R}^d$  separating  $x$  and  $K$ . This half-space induces a half-space in  $(\mathbb{Z}^d, C)$ .

Onn [27] proved that the Radon number of this space is at most  $O(d2^d)$  and at least  $2^d$ . Our results imply that weak  $\epsilon$ -nets exist in this case as well.

Note that the family of half-spaces has VC dimension  $d + 1$ , which is much smaller than the Radon number. Thus, Theorem 6 (which we state in the next subsection) gives better bounds on the size of the  $\epsilon$ -net than Theorem 2.

**Example 6 (linear extensions of posets).** Let  $\Omega$  be a set. For a partial order  $P$  on  $\Omega$ , let  $c(P) \subseteq X$  denote the set of all linear orders that extend  $P$ . Fix a partial order  $P_0$ , and consider the family  $C$  of all sets of the form  $c(P)$ , where  $P$  is a partial order that extends  $P_0$ . The space  $(c(P_0), C)$  is a separable convexity space whose half-spaces correspond to partial orders defined by taking two  $P_0$ -incomparable elements  $x, y \in \Omega$  and extending  $P_0$  by setting  $x < y$ .

## 1.2 Upper bound

### From Radon to Helly and VC

Let  $(X, C)$  be a separable convexity space and let  $B$  denote the family of half-spaces of  $C$  (the notation  $B$  is chosen to reflect that  $B$  generates all convex sets in  $C$  by taking intersection, and hence can be seen as a *basis*).



We first observe that the Radon number is an upper bound on the Helly number and the VC dimension of  $B$ . The *Helly number* of a family  $B$  is the minimum number  $h$  such that every finite<sup>5</sup>  $B' \subseteq B$  with  $\bigcap B' = \emptyset$  contains a subfamily  $B''$  with at most  $h$  sets such that already the intersection of the sets in  $B''$  is empty. Helly theorem states that the Helly number of half-spaces in  $\mathbb{R}^d$  is at most  $d + 1$ . The *VC dimension* of  $B \subseteq 2^X$  is the supremum over  $v$  for which there exists  $Y \subseteq X$  of size  $|Y| = v$  so that for every  $Z \subseteq Y$  there is a member of the family that contains  $Z$  and is disjoint from  $Y \setminus Z$ .

► **Lemma 5.** *Let  $B$  be the family of half-spaces of a separable convexity space  $C$ . If the Radon number of  $C$  is  $r$  then the VC dimension and the Helly number of  $B$  are less than  $r$ .*

The bound on the Helly number follows from a result by Levi [24], and the bound on the VC dimension is straightforward. The proof appears in Appendix A.

Lemma 5 reduces the construction of weak nets for separable convexity spaces to the following, more general construction.

► **Theorem 6.** *Let  $X$  be a set, let  $B \subseteq 2^X$  be a compact family with VC dimension  $v$  and Helly number  $h$ , and let  $B^\cap$  denote the family generated by taking arbitrary intersections of members of  $B$ . Then  $B^\cap$  has weak  $\epsilon$ -nets of size at most  $\beta < (120h^2/\epsilon)^{4hv \ln(1/\epsilon)}$  for every  $\epsilon > 0$ .*

We next give an overview of the proof of Theorem 6; the complete proof appears in Section 2.

## Outline of construction

The construction of weak nets underlying Theorem 6 is short and simple. We want to pierce all convex sets  $c \in C$  such that  $\mu(c) \geq \epsilon$ . We distinguish between two cases. The simpler case is when  $c$  can be written as the intersection of half-spaces, each of which has  $\mu$ -measure more than  $1 - 1/h$ . By Helly's property, it follows that there is a single point that pierces all such  $c$ 's. In the complementary case, when  $c$  can not be written in this way, we use Haussler's packing lemma [19] and show that there is a small collection  $A \subseteq B$  such that conditioning  $\mu$  on a single  $a \in A$  increases the measure of  $c$  by a factor of at least  $1 + 1/(2h)$ . The size of  $A$  can be bounded from above in terms of the VC dimension of  $B$ . So, constructing a set that pierces all  $c$ 's with measure at least  $\epsilon$  is reduced to constructing a bounded number of nets for larger density  $\epsilon' \geq (1 + 1/(2h))\epsilon$ .

To summarize, the Helly number yields the theorem for  $\epsilon$  close to 1, and the VC dimension allows to keep increasing the density until the Helly number becomes relevant.

Going back to the discussion after Theorem 2 concerning the mechanism that suggests "good points" we see that this mechanism is based on the Helly property (roughly speaking, the Helly property is a mechanism that given a collection of sets outputs a point).

## 1.3 Lower bound

The following lemma is a slight generalization of the lower bound in Theorem 2 (we do not assume separability or compactness, and replace  $1/4$  by any  $\epsilon > 0$ ).

► **Lemma 7.** *Let  $(X, C)$  be a convexity space. If the Radon number of  $C$  is greater than  $r > 0$  then there is a distribution  $\mu$  on  $X$  so that every weak  $\epsilon$ -net for  $C$  over  $\mu$  has size at least  $(1 - 2\epsilon)r$ .*

---

<sup>5</sup> The finiteness assumption can be removed when  $B$  is compact.

This gives a non-trivial lower bound as long as  $\epsilon < 1/2$ . Example 4 shows that this is sharp in the sense that when  $\epsilon > 1/2$  there is no lower bound that tends to infinity with the Radon number.

The proof of Lemma 7, as well as a finer distribution dependent lower bound, appear in Section 3. The proof is essentially by reduction to the chromatic number of Kneser graphs.

## 1.4 The necessity of assumptions

### Assumptions in Theorem 6

We first show that if either of the assumptions of having bounded VC dimension or of having bounded Helly number is removed then Theorem 6 ceases to hold.

To see why bounded Helly number is necessary, let  $X$  be any finite set and set  $B = \{X\} \cup \{X \setminus \{x\} : x \in X\}$ . The VC dimension of  $B$  is 1, but any subset of  $X$  can be represented as an intersection of members of  $B$ . Thus,  $B^\cap = 2^X$ , which does not have weak  $\epsilon$ -nets of size which is independent of  $|X|$ .

To see why bounded VC dimension is necessary, consider the convexity space  $(X, C)$  of cylinders (Example 3 in Section 1.1). Here,  $X = \{0, 1\}^n$ ,  $C$  is the family of cylinders, and the family of half-spaces  $B$  consists of cylinders with co-dimension 1:  $B = B_0 \cup B_1$  with

$$B_t = \{x \in X : x_i = t\} : i \in [n]\}.$$

The Helly number of  $B$  is 2, since an intersection of half-spaces is empty if and only if two complementing cylinders with co-dimension 1 participate in it.

The following claim gives a lower bound on weak  $\frac{1}{4}$ -nets for  $C$  over the uniform distribution  $\mu$  over  $X$ .

▷ **Claim 8.** Every weak  $(1/4)$ -net for  $C$  over  $\mu$  has size at least  $\log n$ .

*Proof.*  $S \subseteq X$  pierces every cylinder with measure  $1/4$  only if for every  $i \neq j$  in  $[n]$  there is  $x \in S$  with  $x_i = 0$  and  $x_j = 1$ . Now, consider the mapping from  $[n]$  to  $\{0, 1\}^S$ , which maps  $i \in [n]$  to  $(x_i)_{x \in S}$ . By the above, this mapping is one-to-one, and in particular  $2^{|S|} = |\{0, 1\}^S| \geq n$ . ◁

### Assumptions in Corollary 3

We now describe an example showing that the compactness assumption in Corollary 3 is necessary (a related example appears in [9, 31] in the context of strong  $\epsilon$ -nets). Let  $X = \omega_1$  be the first uncountable ordinal, and  $C$  be the family of all intervals in the well-ordering of  $X$  (a set  $I \subseteq X$  is an interval if whenever  $a, b \in I$  and  $a \leq x \leq b$  then also  $x \in I$ ). The space  $(X, C)$  is separable with Radon number 3, but is not compact.

We claim that it does not have finite weak  $\epsilon$ -nets, even for  $\epsilon = 1$ . Indeed, let  $\mu$  be the probability distribution defined over the  $\sigma$ -algebra generated by countable subsets of  $X$  and assigns every countable subset of  $X$  measure 0. Every interval is either countable or has a countable complement, and is therefore measurable.

We now claim that there is no finite  $S \subseteq X$  that pierces all intervals of measure 1. Indeed, let  $S$  be finite, and let  $m$  be the maximum element in  $S$ . The interval  $\{x \in X : x > m\}$  has measure 1 but is not pierced by  $S$ .

## 2 Proof of upper bound

Here we construct weak  $\epsilon$ -nets when the Helly number and the VC dimension of the half-spaces are bounded (Theorem 6). The property of VC classes that we use is the following packing lemma due to Haussler [19].

► **Theorem 9 (Haussler).** *Let  $B \subseteq 2^X$  be a class of VC dimension  $v$ . For every distribution  $\mu$  on  $X$  and for every  $\delta > 0$ , there is  $A \subseteq B$  of size  $|A| \leq (4e^2/\delta)^v$  such that for every  $b \in B$  there is  $a \in A$  with  $\mu(a\Delta b) \leq \delta$ .*

Haussler's stated the lemma in a dual way; the number of disjoint balls of a given radius in a VC class is small. Haussler's proof is elaborate, but a weaker bound can be proved fairly easily. Indeed, consider a finite set  $A$  so that  $\mu(a\Delta a') > \delta$  for all  $a \neq a'$  in  $A$ . Let  $x_1, \dots, x_m$  be  $m$  independent samples from  $\mu$  for  $m \geq 2 \log(|A|)/\delta$ . Let  $Y = \{x_1, \dots, x_m\}$ , and let  $A|_Y = \{a \cap Y : a \in A\}$ . On one hand, the Sauer-Shelah-Perles lemma implies that  $A|_Y$  is small:  $|A|_Y| \leq (em/v)^v$ . On the other hand, by the union bound,  $|A|_Y| = |A|$  with positive probability. This implies that  $A$  is small.

**Proof of Theorem 6.** We start by focusing on the set  $B_0 = \{b \in B : \mu(b) > 1 - 1/h\}$ . By the union bound, every  $h$  members of  $B_0$  intersect. Since  $B_0 \subseteq B$  is compact with Helly number  $h$ , there is a single point  $x_0 \in X$  that pierces all sets in  $B_0$ .

Let  $0 < \epsilon < 1$ . We construct the  $\epsilon$ -net by induction on  $N(\epsilon)$ , which is defined to be the minimum integer  $n$  such that  $\epsilon(1 + 1/(2h))^n > 1 - 1/h$ .

### Induction base

If  $N(\epsilon) = 0$ , define the piercing set  $S = S(\mu, \epsilon)$  as

$$S = \{x_0\},$$

where  $x_0$  is the point that pierces all half-spaces in  $B_0$ . Indeed,  $S$  is an  $\epsilon$ -net as every  $c \in C$  with  $\mu(c) \geq \epsilon > 1 - 1/h$  is the intersection of half-spaces from  $B_0$ , so  $x_0$  pierces  $c$  as well.

### Induction step

Let  $1 > \epsilon > 0$  such that  $N(\epsilon) > 0$ . We construct the piercing set  $S = S(\mu, \epsilon)$  as follows: Set  $\delta = \epsilon/(2h)^2$  and pick some  $A \subseteq B$  as in Theorem 9. Also set  $\epsilon' = (1 + 1/(2h))\epsilon$ . Note that  $N(\epsilon') = N(\epsilon) - 1$ . By induction, for each  $a \in A$  with  $\mu(a) > 0$ , pick a piercing set  $S_a = S(\mu|_a, \epsilon')$ , where  $\mu|_a$  denotes the distribution  $\mu$  conditioned on  $a$ . Finally, let

$$S = \{x_0\} \cup \bigcup_{a \in A: \mu(a) > 0} S_a.$$

It remains to prove that  $S$  satisfies the required properties.

**$S$  is piercing.** Let  $c \in C$  with  $\mu(c) \geq \epsilon$ . If  $c$  is generated<sup>6</sup> by  $B_0$  then  $x_0$  pierces  $c$ . Otherwise, there is some  $b \in B$  with  $\mu(b) \leq 1 - 1/h$  that contains  $c$ . Pick  $a \in A$  such that  $\mu(a\Delta b) \leq \delta$ . Since  $\mu(b) \geq \epsilon$  and  $\mu(a\Delta b) \leq \delta$ ,

$$\mu(a) \geq \mu(b) - \mu(b \setminus a) \geq \epsilon - \delta > 0,$$

<sup>6</sup> I.e.  $c$  can be presented as an intersection of sets from  $B_0$ .

which means that  $\mu|_a$  is well-defined. We claim that  $S(\mu|_a, \epsilon')$  pierces  $c$ . To this end, it suffices to show that  $\mu|_a(c) \geq \epsilon'$ :

$$\begin{aligned}
 \mu|_a(c) &= \frac{\mu(c \cap a)}{\mu(a)} \\
 &= \frac{\mu(c) - \mu(c \cap (b \setminus a))}{\mu(a)} && (c \subseteq b) \\
 &\geq \frac{\mu(c) - \mu(b \setminus a)}{\mu(a)} \\
 &\geq \frac{\epsilon - \delta}{1 - 1/h + \delta} && (\mu(c) \geq \epsilon, \mu(b) \leq 1 - 1/h, \mu(a \Delta b) \leq \delta) \\
 &\geq \frac{\epsilon(1 - 1/(2h)^2)}{1 - 1/(2h)} && (\delta = \epsilon/(2h)^2) \\
 &= \epsilon'.
 \end{aligned}$$

**S is small.** Let  $\beta(n)$  denote the maximum possible size of  $S$  for  $\epsilon$  with  $N(\epsilon) = n$ . The argument above yields

$$\beta(n) \leq 1 + (4e^2/\delta)^v \cdot \beta(n-1) = 1 + (16e^2h^2/\epsilon)^v \cdot \beta(n-1).$$

Since  $\beta(0) = 1$ , we get

$$\beta(n) \leq (120h^2/\epsilon)^{vn}.$$

Finally, since  $N(\epsilon) \leq 4h \ln(1/\epsilon)$ , we get the bound

$$|S(\mu, \epsilon)| \leq (120h^2/\epsilon)^{4hv \ln(1/\epsilon)}. \quad \blacktriangleleft$$

### 3 Proof of lower bound

The proof of Lemma 7 is based on the following distribution-dependent lower bound on the size of weak nets:

► **Lemma 10.** *Let  $C$  be a family of subsets over a domain  $X$  and let  $\mu$  be a distribution on  $X$ . For  $\epsilon > 0$  define a graph  $G = G(\mu, \epsilon)$  whose vertices are the sets  $c \in C$  such that  $\mu(c) \geq \epsilon$ , and two sets are connected by an edge if and only if they are disjoint. Then, every weak  $\epsilon$ -net for  $C$  over  $\mu$  has size at least the chromatic number of  $G$ , which is denoted by  $\chi(G)$ .*

**Proof.** Let  $S$  be a set that pierces all  $c \in C$  with  $\mu(c) \geq \epsilon$ . Define a coloring of  $G$  by assigning to every vertex  $c$  an element  $x \in S \cap c$ . This is a proper coloring of  $G$ , since if  $\{c, c'\}$  is an edge of  $G$  then  $c$  and  $c'$  are disjoint and therefore can not be pierced by the same element of  $S$ . ◀

Lemma 10 is tight whenever the class  $C$  has Helly number 2 (like in examples 3 and 4 in Section 1.1). Indeed, consider an optimal coloring of  $G(\mu, \epsilon)$ . Every color class is an independent set in  $G$  (which means that every two sets in it have a non-empty intersection). So, when the Helly number is 2, each color class can be pierced by a single element, and we get a piercing set of size  $\chi(G)$ .

The proof Lemma 7 thus reduces to a lower bound on the chromatic number of the relevant graph, which in our case contains a copy of the Kneser graph. The *Kneser graph*  $KG_{n,k}$  is the graph whose vertices correspond to the  $k$ -element subsets of a set of  $n$  elements, and where two vertices are adjacent if and only if the two corresponding sets are disjoint.

## 51:10 On Weak $\epsilon$ -Nets and the Radon Number

Lovasz [25] proved Kneser's conjecture on the chromatic number of this graph (this proof is considered seminal in the topological method in combinatorics):

► **Theorem 11** (Lovasz). *The chromatic number of  $KG_{n,k}$  is  $n - 2k + 2$ .*

We actually do not need the full strength of Lovasz's result. A lower bound of the form  $\chi(KG_{n,n/4}) \geq n/10$  suffices for deducing the equivalence between the existence of weak nets and finite Radon number (in fact, even much weaker bounds suffice). Noga Alon informed us that for this range of the parameters there is a short and elementary proof [1]. Since this argument does not appear in the literature and may be useful elsewhere, we include Alon's proof in Section B.

**Proof of Lemma 7.** Since the Radon number is greater than  $r$ , it follows that there is a set  $Y \subseteq X$  of size  $r$  that is Radon-shattered by  $C$ . Pick  $\mu$  to be the uniform distribution over  $Y$ . By Lemma 10 it suffices to show that  $\chi(G(\mu, \epsilon)) \geq (1 - 2\epsilon)r$ . This follows by noticing that since  $Y$  is Radon-shattered, it follows that the subgraph of  $G(\mu, \epsilon)$  induced by the vertices  $\text{conv}(Z)$ , for  $Z \subseteq Y$  is of size  $\lceil \epsilon r \rceil$ , is isomorphic to  $KG_{r, \lceil \epsilon r \rceil}$ . ◀

### 4 Proof of equivalences

#### 4.1 The existence of weak nets

**Proof of Corollary 3.** Let  $(X, C)$  be a compact separable convexity space.

1  $\Rightarrow$  3. By Lemma 5, the family  $B$  of half-spaces of  $C$  has finite VC dimension and Helly number. Theorem 6 now implies that  $B^\cap = C$  has finite weak  $\epsilon$ -nets for every  $\epsilon > 0$ .

3  $\Rightarrow$  2. Obvious.

2  $\Rightarrow$  1. Assume that  $C$  has weak  $\epsilon$ -nets of size  $\beta = \beta(\epsilon) < \infty$  for some  $\epsilon < 1/2$ . By Lemma 7, the Radon number of  $(X, C)$  is at most  $\frac{\beta}{1-2\epsilon}$ . ◀

#### 4.2 The threshold $1/2$ is sharp

Here we describe Example 4. Let  $X = \{0, 1\}^{\mathbb{N}}$  be the Cantor space, and let  $C$  be the family of all cylinders; recall that  $c \subseteq X$  is a cylinder if there exist  $Y \subseteq \mathbb{N}$  and  $u \in \{0, 1\}^Y$  such that  $c = \{v \in X : v|_Y = u\}$ . The space  $(X, C)$  is a separable convexity space. It is also compact (this follows e.g. from Tychonoff's theorem).

We claim that  $(X, C)$  has  $\epsilon$ -nets of size 1 for every  $\epsilon > 1/2$ . This follows since for every distribution  $\mu$  the family  $\{c \in C : \mu(c) > 1/2\}$  is intersecting<sup>7</sup>, and since  $C$  has Helly number 2. Hence,  $\cap\{c \in C : \mu(c) > 1/2\} \neq \emptyset$  for all  $\mu$ , as claimed.

It remains to show that  $(X, C)$  has no finite weak  $\epsilon$ -nets for  $\epsilon < 1/2$ . By Corollary 3, it suffices to consider the case  $\epsilon = 1/4$ . Let  $\mu$  be the Bernoulli measure on the Cantor space; namely the infinite product of uniform measure on  $\{0, 1\}$ . Now,  $S \subseteq X$  is a weak  $\epsilon$ -net over  $\mu$  if and only if it intersects every cylinder with co-dimension 2. In particular for every  $i \neq j$  in  $\mathbb{N}$  there must be  $x \in S$  with  $x_i = 0$  and  $x_j = 1$ . By the proof of Claim 8 it follows that such an  $S$  must be infinite.

---

<sup>7</sup> A family of sets is *intersecting* if every two members of it intersect.

## 5 An extension

Consider the space of bounded closed convex sets in  $\mathbb{R}^d$ . The corresponding convexity space is not separable nor compact. Nevertheless, our results extend to this space as well.

The following variants of separability and compactness suffice. A convexity space is called *locally-separable* if there is a set  $B \subseteq C$  so that  $C = B^\cap$  and for every finite  $Y \subseteq X$  and for every  $b \in B$  there is  $\bar{b} \in B$  so that  $b \cap Y$  and  $\bar{b} \cap Y$  form a partition of  $Y$ . Every separable space is locally-separable, but there are convexity spaces which are locally separable but not separable (like the space of bounded convex sets in  $\mathbb{R}^d$ ). A convex set  $c \in C$  is called *compact* if the restricted convexity space  $(c, \{c' \cap c : c' \in C\})$  is compact. The *Radon number* of  $c$  is the Radon number of the space  $(c, \{c' \cap c : c' \in C\})$ .

► **Theorem 12.** *Let  $(X, C)$  be a locally separable convexity space such that there exists a chain  $c_1 \subseteq c_2 \subseteq \dots$  of compact convex sets each of which has Radon number at most  $r$  such that  $\bigcup_i c_i = X$ . Then  $(X, C)$  has finite weak  $\epsilon$ -nets for every  $\epsilon > 0$ .*

Theorem 2 and its proof apply for locally-separable and compact convexity spaces. Theorem 12 therefore follows by applying it to a  $c_i$  in the chain such that  $\mu(c_i) \geq 1 - \epsilon/2$ .

## 6 Future research

We showed that for compact separable convexity spaces, the existence of weak nets is equivalent to having a finite Radon number. We now suggest several directions for future research.

One interesting direction is to find a characterization that is valid even more generally (i.e. for families that are not necessarily separable convexity spaces). It is worth noting in this context that the definition of the Radon number can be extended to arbitrary families. One may extend the definition of Radon-shattering as follows: A family  $C \subseteq 2^X$  *Radon-shatters* the set  $Y \subseteq X$  if for every partition of  $Y$  into two parts  $Y_1, Y_2$  there are two disjoint sets  $c_1, c_2 \in C$  such that  $c_1 \cap Y = Y_1$  and  $c_2 \cap Y = Y_2$ .

This extension of the Radon number does not characterize the existence of weak nets for arbitrary families. For instance, let  $C = \{Y \subseteq [n] : |Y| > n/2\}$ . The Radon number is 2 since every two sets in  $C$  intersect, but every weak  $\frac{1}{2}$ -net over the uniform distribution has size at least  $n/2 - 1$ . However, this family  $C$  is not convex (closed under intersection), which is a crucial property in our work. It may also be interesting to fully understand the role of convexity in the context of weak nets.

Alon et al. [4] studied weak nets and the  $(p, q)$ -property in an abstract setting and described connections to fractional Helly properties. It may further be interesting to investigate which other combinatorial properties of convex sets apply in more general settings.

An additional question that comes to mind is the dependence of the size of weak  $\epsilon$ -nets on  $\epsilon$ . In this direction, Bukh, Matousek and Nivasch proved an  $\Omega\left(\frac{1}{\epsilon} \log^{d-1} \frac{1}{\epsilon}\right)$  lower bound on the size of weak  $\epsilon$ -nets for convex sets in  $\mathbb{R}^d$  [10], and recently Rubinfeld [28] proved an upper bound of roughly  $O(1/\epsilon^{3/2})$  in  $\mathbb{R}^2$  (which improves upon the general bound in  $\mathbb{R}^d$  of roughly  $O(1/\epsilon^d)$  by [11]). Proving tight bounds on the size of weak  $\epsilon$ -nets is a central open problem in this area. The general framework developed here may be useful in proving stronger lower bounds.

## References

- 1 Noga Alon. Private communication.
- 2 Noga Alon. A Non-linear Lower Bound for Planar Epsilon-nets. *Discrete & Computational Geometry*, 47(2):235–244, 2012. doi:10.1007/s00454-010-9323-7.
- 3 Noga Alon, Imre Bárány, Zoltán Füredi, and Daniel J Kleitman. Point selections and weak  $\epsilon$ -nets for convex hulls. *Combinatorics, Probability and Computing*, 1(03):189–200, 1992.
- 4 Noga Alon, Gil Kalai, Jiri Matousek, and Roy Meshulam. Transversal numbers for hypergraphs arising in geometry. *Advances in Applied Mathematics*, 29(1):79–101, 2002.
- 5 Noga Alon, Haim Kaplan, Gabriel Nivasch, Micha Sharir, and Shakhar Smorodinsky. Weak  $\epsilon$ -nets and interval chains. *J. ACM*, 55(6):28:1–28:32, 2008. doi:10.1145/1455248.1455252.
- 6 Noga Alon and Daniel J Kleitman. Piercing convex sets and the Hadwiger-Debrunner  $(p, q)$ -problem. *Advances in Mathematics*, 96(1):103–112, 1992.
- 7 Noga Alon and Joel H. Spencer. *The probabilistic method*. John Wiley & Sons, Inc., 2000.
- 8 Imre Bárány, Zoltán Füredi, and László Lovász. On the number of halving planes. *Combinatorica*, 10(2):175–183, 1990.
- 9 A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. Assoc. Comput. Mach.*, 36(4):929–965, 1989. doi:10.1145/76359.76371.
- 10 Boris Bukh, Jiří Matoušek, and Gabriel Nivasch. Lower bounds for weak epsilon-nets and stair-convexity. *Israel Journal of Mathematics*, 182(1):199–228, 2011.
- 11 Bernard Chazelle, Herbert Edelsbrunner, Michelangelo Grigni, Leonidas Guibas, Micha Sharir, and Emo Welzl. Improved bounds on weak  $\epsilon$ -nets for convex sets. In *STOC*, pages 495–504, 1993.
- 12 Victor Chepoi. Separation of two convex sets in convexity structures. *Journal of Geometry*, 50(1):30–51, 1994. doi:10.1007/BF01222661.
- 13 L. Danzer, B. Grünbaum, and V. Klee. *Helly's Theorem and Its Relatives*. Proceedings of symposia in pure mathematics: Convexity. American Mathematical Society, 1963. URL: <https://books.google.com/books?id=I115HAAACAAJ>.
- 14 Esther Ezra. A note about weak epsilon-nets for axis-parallel boxes in d-space. *Inf. Process. Lett.*, 110(18-19):835–840, 2010. doi:10.1016/j.ipl.2010.06.005.
- 15 Branko Grünbaum and Theodore S. Motzkin. On Components in Some Families of Sets. *Proceedings of the American Mathematical Society*, 12(4):607–613, 1961. URL: <http://www.jstor.org/stable/2034254>.
- 16 Hugo Hadwiger and H Debrunner. Über eine variante zum hellyschen satz. *Archiv der Mathematik*, 8(4):309–313, 1957.
- 17 P.C. Hammer. Semispaces and the Topology of Convexity, 1961. URL: [https://books.google.com/books?id=0Vkn\\_gAACAAJ](https://books.google.com/books?id=0Vkn_gAACAAJ).
- 18 Preston C. Hammer. Maximal convex sets. *Duke Math. J.*, 22(1):103–106, March 1955. doi:10.1215/S0012-7094-55-02209-2.
- 19 David Haussler. Sphere packing numbers for subsets of the Boolean n-cube with bounded Vapnik-Chervonenkis dimension. *Journal of Combinatorial Theory, Series A*, 69(2):217–232, 1995.
- 20 David Haussler and Emo Welzl. Epsilon-Nets and Simplex Range Queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
- 21 David Kay and Eugene W Womble. Axiomatic convexity theory and relationships between the Carathéodory, Helly, and Radon numbers. *Pacific Journal of Mathematics*, 38(2):471–485, 1971.
- 22 V. L. Klee. The Structure Of Semispaces. *Mathematica Scandinavica*, 4(1):54–64, 1956. URL: <http://www.jstor.org/stable/24490011>.
- 23 D. J. Kleitman. Families of non-disjoint subsets. *J. Combinatorial Theory*, pages 153–155, 1966.



- 24 Friedrich W Levi. On Helly's theorem and the axioms of convexity. *J. Indian Math. Soc.*, 15:65–76, 1951.
- 25 László Lovász. Kneser's conjecture, chromatic number, and homotopy. *Journal of Combinatorial Theory, Series A*, 25(3):319–324, 1978.
- 26 Jiří Matoušek. A Lower Bound for Weak epsilon-Nets in High Dimension. *Discrete & Computational Geometry*, 28(1):45–48, 2002. doi:10.1007/s00454-001-0090-3.
- 27 Shmuel Onn. On the Geometry and Computational Complexity of Radon Partitions in the Integer Lattice. *SIAM J. Discrete Math.*, 4(3):436–447, 1991. doi:10.1137/0404039.
- 28 Natan Rubin. An Improved Bound for Weak Epsilon-Nets in the Plane. In *FOCS*, pages 224–235. IEEE Computer Society, 2018.
- 29 Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- 30 M.L.J. van de Vel. *Theory of Convex Structures*, volume 50 of *North-Holland mathematical library*. North-Holland, 1993. URL: <https://books.google.com/books?id=xt9-1AEACAAJ>.
- 31 Roberta S Wenocur and Richard M Dudley. Some special vapnik-chervonenkis classes. *Discrete Mathematics*, 33(3):313–318, 1981.

## A Radon, Helly and VC

Here we prove that the Radon number bounds from above both the Helly number and the VC dimension (Lemma 5). The proof follows from the following two claims. Levi [24] proved that

▷ **Claim 13.** Let  $C$  be a convexity space. If the Radon number of  $C$  is  $r$  then its Helly number is smaller than  $r$ .

Proof (for completeness). Let  $C' \subseteq C$  be a finite family so that  $\bigcap_{c \in C'} c = \emptyset$ . Let  $K \subseteq C'$  be a minimal subfamily so that  $\bigcap_{c \in K} c = \emptyset$ . Assume towards a contradiction that  $|K| \geq r$ . Minimality implies that for each  $k \in K$  we have  $C_k := \bigcap_{c \in K \setminus \{k\}} c \neq \emptyset$ . Let  $x_k \in C_k$ . The  $x_k$ 's must be distinct (otherwise  $\bigcap_{c \in K} c \neq \emptyset$ ). Thus, there is a partition of  $\{x_k : k \in K\}$  to two parts  $Y_1, Y_2$  such that  $\text{conv}(Y_1) \cap \text{conv}(Y_2) \neq \emptyset$ . But

$$\text{conv}(Y_1) \subseteq \bigcap_{k \in K: x_k \in Y_2} k \quad \text{and} \quad \text{conv}(Y_2) \subseteq \bigcap_{k \in K: x_k \in Y_1} k,$$

by construction. This is a contradiction, so  $|K| < r$ . ◁

We observe that

▷ **Claim 14.** Let  $C$  be a convexity space and  $B$  be its half-spaces. If the Radon number of  $C$  is  $r$  then the VC dimension of  $B$  is smaller than  $r$ .

Proof. Let  $Y \subseteq X$  be of size  $r$ . The set  $Y$  can thus be partitioned to  $Y_1, Y_2$  so that  $\text{conv}(Y_1) \cap \text{conv}(Y_2) \neq \emptyset$ . Assume, towards a contradiction, that there is  $b \in B$  so that  $b \cap Y = Y_1$ . Since  $B$  consists of half-spaces<sup>8</sup>, there is  $\bar{b} \in B \subseteq C$  so that  $Y_2 = \bar{b} \cap Y$ . This implies that  $\text{conv}(Y_1) \cap \text{conv}(Y_2) = \emptyset$ , which is a contradiction. Thus, for all  $b \in B$  we have  $b \cap Y \neq Y_1$  which means that the VC dimension is less than  $|Y| = r$ . ◁

<sup>8</sup> Here we use a more general definition of half-spaces, as in the definition of locally-separable in Section 5.

## B The chromatic number of the Kneser graph

Here we prove a lower bound on the chromatic number of the Kneser graph (which is weaker than Lovasz's). We follow an argument of Alon [1], who informed us that a similar argument was independently found by Szemerédi. We focus on the following case, but the argument applies more generally.

► **Theorem 15.** *For  $n$  be divisible by 4 we have  $\chi(KG_{n,n/4}) > n/10$ .*

The first step in the proof is the following lemma proved by Kleitman [23]. A family  $F \subseteq 2^X$  is called intersecting if  $f \cap f' \neq \emptyset$  for all  $f, f' \in F$ .

► **Lemma 16.** *If  $F_1, \dots, F_s \subset 2^{[n]}$ , where each  $F_i$  is intersecting, then*

$$\left| \bigcup_{i \in [s]} F_i \right| \leq 2^n - 2^{n-s}.$$

The lemma can be proved by induction on  $s$ . The case  $s = 1$  just says that an intersecting family has size at most  $2^{n-1}$ . The induction step is based on correlation of monotone events (for more details see, e.g. [7]).

**Proof of Theorem 15.** Consider a proper coloring of  $KG_{n,n/4}$  with  $s$  colors. Let  $V_1, \dots, V_s$  be the partition of the vertices to color classes. Each  $V_i$  is an intersecting family. Let  $F_i$  be the family of sets  $u \subseteq [n]$  that contain some set in  $V_i$ . Each  $F_i$  is also intersecting. By the lemma above,

$$2^n - \left| \bigcup_{i \in [s]} F_i \right| \geq 2^{n-s}.$$

On the other hand, the complement of  $\bigcup_{i \in [s]} F_i$  is of size less than  $\sum_{k=0}^{n/4} \binom{n}{k} \leq 2^{nH(1/4)}$ , where  $H(p) = -p \log(p) - (1-p) \log(1-p)$  is the binary entropy function. Hence,

$$s > n(1 - H(1/4)) \geq n/10. \quad \blacktriangleleft$$

# Dynamic Planar Point Location in External Memory

**J. Ian Munro**

Cheriton School of Computer Science, University of Waterloo, Canada  
imunro@uwaterloo.ca

**Yakov Nekrich**

Cheriton School of Computer Science, University of Waterloo, Canada  
yakov.nekrich@gmail.com

---

## Abstract

In this paper we describe a fully-dynamic data structure for the planar point location problem in the external memory model. Our data structure supports queries in  $O(\log_B n(\log \log_B n)^3)$  I/Os and updates in  $O(\log_B n(\log \log_B n)^2)$  amortized I/Os, where  $n$  is the number of segments in the subdivision and  $B$  is the block size. This is the first dynamic data structure with almost-optimal query cost. For comparison all previously known results for this problem require  $O(\log_B^2 n)$  I/Os to answer queries. Our result almost matches the best known upper bound in the internal-memory model.

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

**Keywords and phrases** Data Structures, Dynamic Data Structures, Planar Point Location, External Memory

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.52

**Related Version** A full version of the paper is available at <http://arxiv.org/abs/1903.06601>.

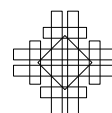
## 1 Introduction

Planar point location is a classical computational geometry problem with a number of important applications. In this problem we keep a polygonal subdivision  $\Pi$  of the two-dimensional plane in a data structure; for an arbitrary query point  $q$ , we must be able to find the face of  $\Pi$  that contains  $q$ . In this paper we study the dynamic version of this problem in the external memory model. We show that a planar subdivision can be maintained under insertions and deletions of edges, so that the cost of queries and updates is close to  $O(\log_B n)$ , where  $n$  is the number of segments in the subdivision and  $B$  is the block size.

Planar point location problem was studied extensively in different computational models. Dynamic internal-memory data structures for general subdivisions were described by Bentley [11], Cheng and Janardan [16], Baumgarten et al. [9], Arge et al. [3], and Chan and Nekrich [13]. Table 1 lists previous results. We did not include in this table many other results for special cases of the point location problem, such as the data structures for monotone, convex, and orthogonal subdivisions, e.g., [25, 26, 18, 17, 21, 20, 14]. The currently best data structure [13] achieves<sup>1</sup>  $O(\log n)$  query time and  $O(\log^{1+\varepsilon} n)$  update time or  $O(\log^{1+\varepsilon} n)$  query time and  $O(\log n)$  update time; the best query-update trade-off described in [13] is  $O(\log n \log \log n)$  randomized query time and  $O(\log n \log \log n)$  update time. See Table 1.

---

<sup>1</sup> In this paper  $\log n$  denotes the binary logarithm of  $n$  when the logarithm base is not specified.



■ **Table 1** Previous results on dynamic planar point location in internal memory. Entries marked  $\dagger$  and  $\ddagger$  require amortization and (Las Vegas) randomization respectively,  $\varepsilon > 0$  is an arbitrarily small constant. Results marked  $*$  are in the RAM model, all other results are in the pointer machine model. Space usage is measured in words.

| Reference             | Space      | Query Time               | Insertion Time                         | Deletion Time                          |             |
|-----------------------|------------|--------------------------|----------------------------------------|----------------------------------------|-------------|
| Bentley [11]          | $n \log n$ | $\log^2 n$               | $\log^2 n$                             | $\log^2 n$                             |             |
| Cheng–Janardan [16]   | $n$        | $\log^2 n$               | $\log n$                               | $\log n$                               |             |
| Baumgarten et al. [9] | $n$        | $\log n \log \log n$     | $\log n \log \log n$                   | $\log^2 n$                             | $\dagger$   |
| Arge et al. [3]       | $n$        | $\log n$                 | $\log^{1+\varepsilon} n$               | $\log^{2+\varepsilon} n$               | $\dagger$   |
| Arge et al. [3]       | $n$        | $\log n$                 | $\log n (\log \log n)^{1+\varepsilon}$ | $\log^2 n / \log \log n$               | $\ddagger*$ |
| Chan and Nekrich [13] | $n$        | $\log n (\log \log n)^2$ | $\log n \log \log n$                   | $\log n \log \log n$                   |             |
| Chan and Nekrich [13] | $n$        | $\log n$                 | $\log^{1+\varepsilon} n$               | $\log^{1+\varepsilon} n$               |             |
| Chan and Nekrich [13] | $n$        | $\log n$                 | $\log^{1+\varepsilon} n$               | $\log n (\log \log n)^{1+\varepsilon}$ | $*$         |
| Chan and Nekrich [13] | $n$        | $\log^{1+\varepsilon} n$ | $\log n$                               | $\log n$                               |             |
| Chan and Nekrich [13] | $n$        | $\log n \log \log n$     | $\log n \log \log n$                   | $\log n \log \log n$                   | $\ddagger*$ |

In the external memory model [2] the data can be stored in the internal memory of size  $M$  or on the external disk. Arithmetic operations can be performed only on data in the internal memory. Every input/output operation (I/O) either reads a block of  $B$  contiguous words from the disk into the internal memory or writes  $B$  words from the internal memory into disk. Measures of efficiency in this model are the number of I/Os needed to solve a problem and the amount of used disk space.

Goodrich et al. [22] presented a linear-space static external data structure for point location in a monotone subdivision with  $O(\log_B n)$  query cost. Arge et al. [5] designed a data structure for a general subdivision with the same query cost. Data structures for answering a batch of point location queries were considered in [22] and [8]. Only three external-memory results are known for the dynamic case. The data structure of Agarwal, Arge, Brodal, and Vitter [1] supports queries on monotone subdivisions in  $O(\log_B^2 n)$  I/Os and updates in  $O(\log_B^2 n)$  I/Os amortized. Arge and Vahrenhold [7] considered the case of general subdivisions; they retain the same cost for queries and insertions as [1] and reduce the deletion cost to  $O(\log_B n)$ . Arge, Brodal, and Rao [4] reduced the insertion cost to  $O(\log_B n)$ . Thus all previous dynamic data structures did not break  $O(\log_B^2 n)$  query cost barrier. For comparison the first internal-memory data structure with query time close to logarithmic was presented by Baumgarten et al [9] in 1994. See Table 2. All previous data structures use  $O(n)$  words of space (or  $O(n/B)$  blocks of  $B$  words<sup>2</sup>).

In this paper we show that it is possible to break the  $O(\log_B^2 n)$  barrier for the dynamic point location problem. Our data structure answers queries in  $O(\log_B n (\log \log_B n)^3)$  I/Os, supports updates in  $O(\log_B n (\log \log_B n)^2)$  I/Os amortized, and uses linear space. Thus we achieve close to logarithmic query cost and a query-update trade-off almost matching the state-of-the-art upper bounds in the internal memory model. Our result is within double-logarithmic factors from optimal. Additionally we describe a data structure that supports point location queries in an orthogonal subdivision with  $O(\log_B n \log \log_B n)$  query cost and  $O(\log_B n \log \log_B n)$  amortized update cost. The computational model used in this paper is the standard external memory model [2].

<sup>2</sup> Space usage of external-memory data structures is frequently measured in disk blocks of  $B$  words. In this paper we measure the space usage in words. But the space usage of  $O(n)$  words is equivalent to  $O(n/B)$  blocks of space.

■ **Table 2** Previous and new results on dynamic planar point location in external memory. G denotes most general subdivisions, M denotes monotone subdivision, and O denotes orthogonal subdivision. Space usage is measured in words and update cost is amortized.

| Reference               | Space | Query Cost                   | Insertion Cost               | Deletion Cost                |   |
|-------------------------|-------|------------------------------|------------------------------|------------------------------|---|
| Agarwal et al [1]       | $n$   | $\log_B^2 n$                 | $\log_B^2 n$                 | $\log_B^2 n$                 | M |
| Arge and Vahrenhold [7] | $n$   | $\log_B^2 n$                 | $\log_B^2 n$                 | $\log_B n$                   | G |
| Arge et al [4]          | $n$   | $\log_B^2 n$                 | $\log_B n$                   | $\log_B n$                   | G |
| This paper              | $n$   | $\log_B n (\log \log_B n)^3$ | $\log_B n (\log \log_B n)^2$ | $\log_B n (\log \log_B n)^2$ | G |
| This paper              | $n$   | $\log_B n \log \log_B n$     | $\log_B n \log \log_B n$     | $\log_B n \log \log_B n$     | O |

## 2 Overview

### 2.1 Overall Structure

As in the previous works, we concentrate on answering *vertical ray shooting* queries. The successor segment of a point  $q$  in a set  $S$  of non-intersecting segments is the first segment that is hit by a ray emanating from  $q$  in the  $+y$ -direction. Symmetrically, the predecessor segment of  $q$  in  $S$  is the first segment hit by a ray emanating from  $q$  in the  $-y$  direction. A vertical ray shooting query for a point  $q$  on a set of segments  $S$  asks for the successor segment of  $q$  in  $S$ . If we know the successor segment or the predecessor segment of  $q$  among all segments of a subdivision  $\Pi$ , then we can answer a point location query on  $\Pi$  (i.e., identify the face of  $\Pi$  containing  $q$ ) in  $O(\log_B n)$  I/Os [7]. In the rest of this paper we will show how to answer vertical ray shooting queries on a dynamic set of non-intersecting segments.

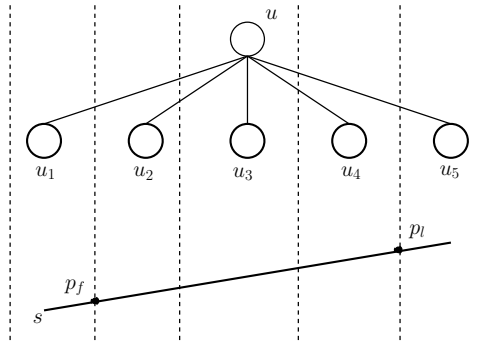
Our base data structure is a variant of the segment tree. Let  $\mathcal{S}$  be a set of segments. We store a tree  $\mathcal{T}$  on  $x$ -coordinates of segment endpoints. Every leaf contains  $\Theta(B)$  segment endpoints and every internal node has  $r = \Theta(B^\delta)$  children for  $\delta = 1/8$ . Thus the height of  $\mathcal{T}$  is  $O(\log_B n)$ . We associate a vertical slab with every node  $u$  of  $\mathcal{T}$ . The slab of the root node is  $[x_{\min}, x_{\max}] \times \mathbb{R}$ , where  $x_{\min}$  and  $x_{\max}$  denote the  $x$ -coordinates of the leftmost and the rightmost segment endpoints. The slab of an internal node  $u$  is divided into  $\Theta(B^\delta)$  slabs that correspond to the children of  $u$ . A segment  $s$  *spans* the slab of a node  $u$  (or simply *spans*  $u$ ) if it crosses its vertical boundaries.

A segment  $s$  is assigned to an internal node  $u$ , if  $s$  spans at least one child  $u_i$  of  $u$  but does not span  $u$ . We assign  $s$  to a leaf node  $\ell$  if at least one endpoint of  $s$  is stored in  $\ell$ . All segments assigned to a node  $u$  are trimmed to slab boundaries of children and stored in a *multi-slab* data structure  $C(u)$ : Suppose that a segment  $s$  is assigned to  $u$  and it spans the children  $u_f, \dots, u_l$  of  $u$ . Then we store the segment  $s_u = [p_f, p_l]$  in  $C(u)$ , where  $p_f$  is the point where  $s$  intersect the left slab boundary of  $u_f$  and  $p_l$  is the point where  $s$  intersects the right boundary of  $u_l$ . See Fig. 1. Each segment is assigned to  $O(\log_B n)$  nodes of  $\mathcal{T}$ .

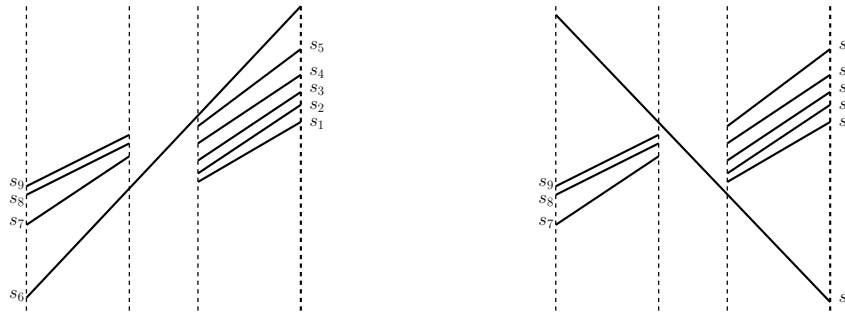
In order to answer a vertical ray shooting query for a point  $q$ , we identify the leaf  $\ell$  such that the slab of  $\ell$  contains  $q$ . Then we visit all nodes  $u$  on the path  $\pi_\ell$  from the root of  $\mathcal{T}$  to  $\ell$  and answer vertical ray shooting queries in multi-slab structures  $C(u)$ .

### 2.2 Our Approach

Thus our goal is to answer  $O(\log_B n)$  ray shooting queries in multi-slab structures along a path in the segment tree  $\mathcal{T}$  with as few I/Os as possible. Segments stored in a multi-slab are not comparable in the general case; see Fig. 2. It is possible to impose a total order  $\prec$  on all segments in the following sense: let  $l$  be a vertical line that intersects segments  $s_1$  and  $s_2$ ; if the intersection of  $l$  with  $s_1$  is above the intersection of  $l$  with  $s_2$ , then  $s_2 \prec s_1$ .



■ **Figure 1** Segment  $s$  is assigned to node  $u$ . The trimmed segment  $[p_f, p_l]$  is stored in  $C(u)$ .



■ **Figure 2** Left: example of segment order in a multi-slab;  $s_1 \prec s_2 \prec s_3 \prec s_4 \prec s_5 \prec s_6 \prec s_7 \prec s_8 \prec s_9$ . Right: a deletion and an insertion of one new segment in a multi-slab changes the order of segments to  $s_7 \prec s_8 \prec s_9 \prec s_0 \prec s_1 \prec s_2 \prec s_3 \prec s_4 \prec s_5$ .

We can find such a total order in  $O((K/B) \log_{M/B} K)$  I/Os, where  $K$  is the number of segments [8, Lemma 3]. But this ordering is not stable under updates: even a single deletion and a single insertion can lead to significant changes in the order of segments. See Fig. 2. Therefore it is hard to apply standard techniques, such as fractional cascading [15, 23], in order to speed-up ray shooting queries. Previous external-memory solutions in [1, 4] essentially perform  $O(\log_B n)$  independent searches in the nodes of a segment tree or an interval tree in order to answer a query. Each search takes  $O(\log_B n)$  I/Os, hence the total query cost is  $O(\log_B^2 n)$ .

Internal memory data structures achieve  $O(\log n)$  query cost using dynamic fractional cascading [15, 23]. Essentially the difference with external memory is as follows: since we aim for  $O(\log_2 n)$  query cost in internal memory, we can afford to use base tree  $\mathcal{T}$  with small node degree. In this special case the segments stored in sets  $C(u)$ ,  $u \in \mathcal{T}$ , can be ordered resp. divided into a small number of ordered sets. When the order of segments in  $C(u)$  is known, we can apply the fractional cascading technique [15, 23] to speed up queries. Unfortunately dynamic fractional cascading does not work in the case when the total order of segments in  $C(u)$  is not known. Hence we cannot use previous internal memory solutions of the point location problem [16, 9, 3, 13] to decrease the query cost in external memory.

In this paper we propose a different approach. Searching in a multi-slab structure  $C(u)$  is based on a weighted search among segments of  $C(u)$ . Weights of segments are chosen in such way that the total cost of searching in all multi-slab structures along a path  $\pi$  is logarithmic. We also use fractional cascading, but this technique plays an auxiliary role: we

apply fractional cascading to compute the weights of segments and to navigate between the tree nodes. Interestingly, fractional cascading is usually combined with the union-split-find data structure, which is not used in our construction.

This paper is structured as follows. In Section 3 we show how our new technique, that will be henceforth called weighted telescoping search, can be used to solve the static vertical ray shooting problem. Next we turn to the dynamic case. In our exposition we assume, for simplicity, that the set of segment  $x$ -coordinates is fixed, i.e., the tree  $\mathcal{T}$  does not change. We also assume that the block size  $B$  is sufficiently large,  $B > \log^8 n$ . We show how our static data structure from Section 3 can be modified to support insertions in Section 4. To maintain the order of segments in a multi-slab under insertions we pursue the following strategy: when a new segment is inserted into the multi-slab structure  $C(u)$ , we split it into a number of *unit* segments, such that every unit segment spans exactly one child of  $u$ . Unit segments can be inserted into a multi-slab so that the order of other segments is not affected. The number of unit segments per inserted segment can be large; however we can use buffering to reduce the cost of updates.<sup>3</sup> We need to make some further changes in our data structure in order to support deletions; the fully-dynamic solution for large  $B$  is described in Section 5. The main result of Section 5, summed up in Lemma 2, is the data structure that answers queries in  $O(\log_B n \log \log_B n)$  I/Os; insertions and deletions are supported in  $O(\log_B^2 n)$  and  $O(\log_B n)$  amortized I/Os respectively. We show how to reduce the cost of insertions in Section 6. We address some missing technical details and consider the case of small block size  $B$  in Section 7. In the full version of this paper [24] we will show how the space usage can be reduced to linear and address some issues related to updates of bridge segments. The special case of vertical ray shooting among horizontal segments is also considered in the full version.

### 3 Ray Shooting: Static Structure

In this section we show how the weighted telescoping search can be used to solve the static point location problem. Let  $\mathcal{T}$  be the tree, defined in Section 2.1, with node degree  $r = B^\delta$  for  $\delta = 1/8$ . Let  $C(u)$  be the set of segments that span at least one child of  $u$  but do not span  $u$ .

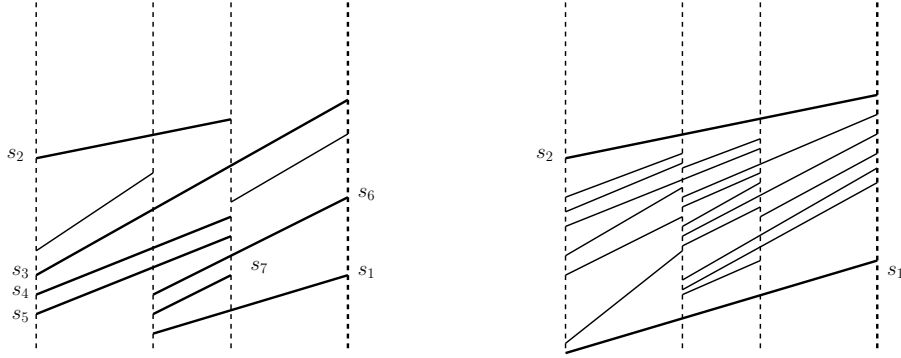
**Augmented Catalogs.** We keep augmented catalogs  $AC(u) \supset C(u)$  in every node  $u$ . Each  $AC(u)$  is divided into subsets  $AC_{ij}(u)$  for  $1 \leq i \leq j \leq r$ ;  $AC_{ij}(u)$  contains segments that span children  $u_i, \dots, u_j$  of  $u$  and only those children. Augmented catalogs  $AC(u)$  satisfy the following properties:

- (i) If a segment  $s \in (AC(u) \setminus C(u))$ , then  $s \in C(v)$  for an ancestor  $v$  of  $u$  and  $s$  spans  $u$ .
- (ii) Let  $E_i(u) = AC(u) \cap AC(u_i)$  for a child  $u_i$  of  $u$ . For any  $f$  and  $l$ ,  $f \leq i \leq l$ , there are at most  $d = O(r^4)$  elements of  $AC_{fl}(u)$  between any two consecutive elements of  $E_i(u)$ .
- (iii) If  $i \neq j$ , then  $E_i(u) \cap E_j(u) = \emptyset$ .

Elements of  $E_i(u)$  for some  $1 \leq i \leq r$  will be called down-bridges; elements of the set  $UP(u) = AC(u) \cap AC(\text{par}(u))$ , where  $\text{par}(u)$  denotes the parent node of  $u$ , are called up-bridges. We will say that a sub-list of a catalog  $AC(u)$  bounded by two up-bridges is a *portion* of  $AC(u)$ . We refer to e.g., [3] or [13] for an explanation how we can construct

<sup>3</sup> As a side remark, this approach works with weighted telescoping search, but it would not work with the standard fractional cascading used in internal-memory solutions [16, 9, 3, 13]. The latter technique relies on a union-split-find data structure (USF) and it is not known how to combine buffering with USF.





■ **Figure 3** Computation of segment weights. Left: segments  $s_1$  and  $s_2$  are down-bridges from  $E_2(u)$ . Segments  $s_3, s_4, s_5, s_6,$  and  $s_7$  are in  $AC(u) \setminus E_2(u)$ . For  $3 \leq j \leq 7$ ,  $weight_2(s_j, u) = W(s_1, s_2, u_2)/d$ . Right: portion of  $AC(u_2)$  for the child  $u_2$  of  $u$ .  $W(s_1, s_2, u_2)$  is equal to the total weight of all segments between  $s_1$  and  $s_2$ . If  $u_2$  is a leaf node, then  $W(s_1, s_2, u_2)$  equals the total number of segments in  $AC(u_2)$  that are situated between  $s_2$  and  $s_1$ .

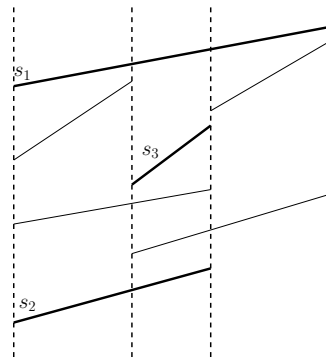
and maintain  $AC(u)$ . We assume in this section that all segments in every catalog  $AC(u)$  are ordered. We can easily order a set  $AC_{fl}(u)$  or any set of segments that cross the same vertical line  $\ell$ : the order of segments is determined by ( $y$ -coordinates of) intersection points of segments and  $\ell$ . Therefore we will speak of e.g., the largest/smallest segments in such a set.

**Element weights.** We assign the weight to each element of  $AC(u)$  in a bottom-to-top manner: All segments in a set  $AC(\ell)$  for every leaf node  $\ell$  are assigned weight 1. Consider a segment  $s \in AC_{fl}(u)$ , i.e., a segment that spans children  $u_f, \dots, u_l$  of some internal node  $u$ . For  $f \leq i \leq l$  let  $s_1$  denote the largest bridge in  $E_i(u)$  that is (strictly) smaller than  $s$  and let  $s_2$  denote the smallest bridge in  $E_i(u)$  that is (strictly) larger than  $s$ ; we let  $W(s_1, s_2, u_i) = \sum_{s_1 < s' < s_2} weight(s', u_i)$ , where the sum is over all segments  $s' \in AC(u_i)$  and  $weight_i(s, u) = W(s_1, s_2, u_i)/d$ . See Fig. 3 for an example. We set  $weight(s, u) = \sum_{i=f}^l weight_i(s, u)$ . We keep a weighted search tree for every portion  $\mathcal{P}(u)$  of the list  $AC(u)$ . By a slight misuse of notation this tree will also be denoted by  $\mathcal{P}(u)$ . Thus every catalog  $AC(u)$  is stored in a forest of weighted trees  $\mathcal{P}_j(u)$  where every tree corresponds to a portion of  $AC(u)$ <sup>4</sup>. We also store a data structure supporting finger searches on  $AC(u)$ .

**Weighted Trees.** Each weighted search tree is implemented as a biased  $(a, b)$ -tree with parameters  $a = B^\delta/2$  and  $b = B^\delta$  [10, 19]. The depth of a leaf  $\lambda$  in a biased  $(B^\delta/2, B^\delta)$ -tree is bounded by  $O(\log_B(W/w_\lambda))$ , where  $w_\lambda$  is the weight of an element in the leaf  $\lambda$  and  $W$  is the total weight of all elements in the tree. Every internal node  $\nu$  has  $B^\delta$  children and every leaf holds  $\Theta(B)$  segments<sup>5</sup>. In each internal node  $\nu$  we keep  $B^{3\delta}$  segments  $\nu.max_{jk}[i]$ . For every child  $\nu_i$  of  $\nu$  and for all  $j$  and  $k$ ,  $1 \leq j \leq k \leq r$ ,  $\nu.max_{jk}[i]$  is the highest segment from  $AC_{jk}$  in the subtree of  $\nu_i$ ; if there are no segments from  $AC_{jk}$  in the subtree of  $\nu_i$ , then  $\nu.max_{jk}[i] = \text{NULL}$ . Using values of  $\nu.max$  we can find, for any node  $\nu$  of the biased search tree, the child  $\nu_i$  of  $\nu$  that holds the successor segment of the query point  $q$ . Hence

<sup>4</sup> In most cases we will omit the subindex and will speak of a weighted tree  $\mathcal{P}(u)$  because it will be clear from the context what portion of  $AC(u)$  is used.

<sup>5</sup> In the standard biased  $(a, b)$ -tree [10, 19], every leaf holds one element. But we can modify it so that every leaf holds  $\Theta(B)$  different elements (segments). The weight of a leaf  $\lambda$  is the total weight of all segments stored in  $\lambda$ .



■ **Figure 4** Segments in a group. Segments  $s_1$ ,  $s_2$ , and  $s_3$  are stored in  $V_2$ :  $s_1$  and  $s_2$  are the highest and the lowest segments that span the second child  $u_2$ ;  $s_3$  is a bridge segment from  $E_2(u)$ .

we can find the smallest segment  $n(u)$  in a portion  $\mathcal{P}(u)$  that is above a query point  $q$  in  $O(\log_B(W_P/\omega_n))$  I/Os where  $W_P$  is the total weight of all segments in  $\mathcal{P}(u)$  and  $\omega_n$  is the weight of  $n(u)$ .

**Additional Structures.** When the segment  $n(u)$  is known, we will need to find the bridges that are closest to  $n(u)$  in order to continue the search. We keep a list  $V_i(u) \subseteq AC(u)$  for each node  $u$  and for every  $i$ ,  $1 \leq i \leq r$ .  $V_i(u)$  contains all segments of  $E_i(u)$  and some additional segments chosen as follows:  $AC(u)$  is divided into groups so that each group consists of  $\Theta(r^6)$  consecutive segments; the only exception is the last group in  $AC(u)$  that contains  $O(r^6)$  segments (here we use the fact that segments in  $AC(u)$  are ordered). We choose the constant in such way that every group but the last one contains  $d \cdot r^2$  segments. If a group  $G$  contains a segment that spans  $u_i$ , then we select the highest segment from  $G$  that spans  $u_i$  and the lowest segment from  $G$  that spans  $u_i$ ; we store both segments in  $V_i$ . See Fig. 4. For every segment in  $V_i$  we also store a pointer to its group in  $AC(u)$ . We keep  $V_i$  in a B-tree that supports finger search queries.

Suppose that we know the successor segment  $n(u)$  of a query point  $q$  in  $AC(u)$ . We can find the successor segment  $b_n(u)$  of  $q$  in  $E_i(u)$  using  $V_i$ : Let  $G$  denote the group that contains  $n(u)$ . We search in  $G$  for the segment  $b_n(u) \geq n(u)$  using finger search. If  $b_n(u)$  is not in  $G$ , we consider the highest segment  $s_1 \in G$  that spans  $u_i$ . By definition of  $AC(u)$ , there are at most  $dr^2$  segments between  $n(u)$  and  $b_n(u)$ . We can find  $b_n(u)$  in  $O(\log_B(dr^2)) = O(1)$  I/Os by finger search on  $V_i$  using  $s_1$  as the finger. Using a similar procedure, we can find the highest bridge segment  $b_p(u) \leq n(u)$  in  $E_i(u)$ .

**Queries.** A vertical ray shooting query for a point  $q = (q_x, q_y)$  is answered as follows. Let  $\ell$  denote the leaf such that the slab of  $\ell$  contains  $q$ . We visit all nodes  $v_0, v_1, \dots, v_h$  on the root-to-leaf path  $\pi(\ell)$  where  $v_0$  is the root node and  $v_h = \ell$ . We find the segment  $n(v_i)$  in every visited node, where  $n(v_i)$  is the successor segment of  $q$  in  $AC(v_i)$ . Suppose that  $v_{i+1}$  is the  $j$ -th child of  $v_i$ ;  $n(v_i)$  spans the  $j$ -th child of  $v_i$ . First we search for  $n(v_0)$  in the weighted tree of  $AC(v_0)$ . Next, using the list  $V_j$ , we identify the smallest bridge  $b_n(v_0) \in E_j(v_0)$  such that  $b_n(v_0) \geq n(v_0)$  and the largest bridge segment  $b_p(v_0) \in E_j(v_0)$  such that  $b_p(v_0) \leq n(v_0)$ . The index  $j$  is chosen so that  $v_1$  is the  $j$ -th child of  $v_0$ . We execute the same operations in nodes  $v_1, \dots, v_h$ . When we are in a node  $v_i$  we consider the portion  $\mathcal{P}(v_i)$  between bridges  $b_p(v_{i-1})$  and  $b_n(v_{i-1})$ ; we search in the weighted tree of  $\mathcal{P}(v_i)$  for the successor segment  $n(v_i)$  of  $q$ . Then we identify the lowest bridge  $b_n(v_i) \geq n(v_i)$  and the highest bridge  $b_p(v_i) \leq n(v_i)$ . When all  $n(v_i)$  are computed, we find the lowest segment  $n^*$  among  $n(v_i)$ . Since  $\cup_{i=0}^h AC(v_i) = \cup_{i=0}^h C(v_i)$ ,  $n^*$  is the successor segment of a query point  $q$ .

The cost of a ray shooting query can be estimated as follows. Let  $\omega_i$  denote the weight of  $n(v_i)$ . Let  $W_i$  denote the total weight of all segments of  $\mathcal{P}(v_i)$  (we assume that  $\mathcal{P}(v_0) = AC(v_0)$ ). Search for  $n(v_i)$  in the weighted tree  $\mathcal{P}(v_i)$  takes  $O(\log_B(W_i/\omega_i))$  I/Os. By definition of weights,  $\omega_i \geq W_{i+1}/d$ . Hence

$$\begin{aligned} \sum_{i=0}^h \log_B(W_i/\omega_i) &= \log_B W_0 + \sum_{i=0}^{h-1} (\log_B W_{i+1} - \log_B \omega_i) - \log_B \omega_h \\ &\leq \log_B(W_0/\omega_h) + 2(h+1) \log_B r. \end{aligned}$$

We have  $\omega_h = 1$  and we will show below that  $W_0 \leq n$ . Since  $r = B^\delta$ ,  $h = O(\log_B n)$  and  $\log_B r = O(1)$ . Hence the sum above can be bounded by  $O(\log_B n)$ . When  $n(v_i)$  is known, we can find  $b_p(v_i)$  and  $b_n(v_i)$  in  $O(1)$  I/Os, as described above. Hence the total cost of answering a query is  $O(\log_B n)$ . Since every segment is stored in  $O(\log_B n)$  lists  $AC(u)$ , the total space usage is  $O(n \log_B n)$ .

It remains to prove that  $W_0 \leq n$ . We will show by induction that the total weight of all elements on every level of  $\mathcal{T}$  is bounded by  $n$ : Every element in a leaf node has weight 1; hence their total weight does not exceed  $n$ . Suppose that, for some  $k \geq 1$ , the total weight of all elements on level  $k-1$  does not exceed  $n$ . Consider an arbitrary node  $v$  on level  $k$ , let  $v_1, \dots, v_r$  be the children of  $v$ , and let  $m_i$  denote the total weight of elements in  $AC(v_i)$ . Every element in  $AC(v_i)$  contributes  $1/d$  fraction of its weight to at most  $d$  different elements in  $AC(v)$ . Hence  $\sum_{e \in AL(v)} \text{weight}_i(v) \leq m_i$  and the total weight of all elements in  $AC(v)$  does not exceed  $\sum_{i=1}^r m_i$ . Hence, for any level  $k \geq 1$ , the total weight of  $AC(v)$  for all nodes  $v$  on level  $k$  does not exceed  $n$ . Hence the total weight of  $AC(u_0)$  for the root node  $u_0$  is also bounded by  $n$ .

► **Lemma 1.** *There exists an  $O(n \log_B n)$ -space static data structure that supports point location queries on  $n$  non-intersecting segments in  $O(\log_B n)$  I/Os.*

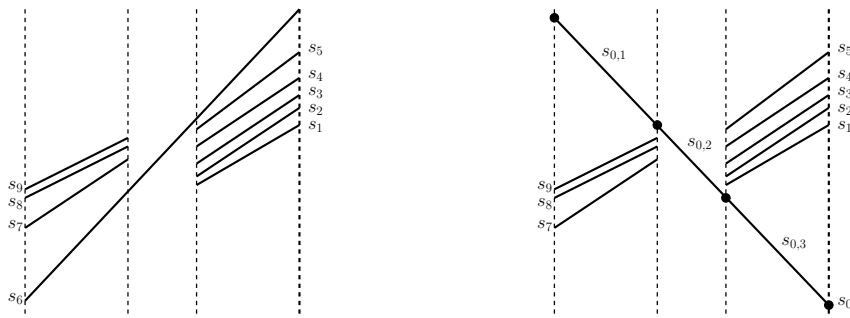
The result of Lemma 1 is not new. However we will show below that the data structure described in this section can be dynamized.

#### 4 Semi-Dynamic Ray Shooting for $B \geq \log^8 n$ : Main Idea

Now we turn to the dynamic problem. In Sections 4 and 5 we will assume<sup>6</sup> that  $B \geq \log^8 n$ .

**Overview.** The main challenge in dynamizing the static data structure from Section 3 is the order of segments. Deletions and insertions of segments can lead to significant changes in the segment order, as explained in Section 2. However segment insertions within a slab are easy to handle in one special case. We will say that a segment  $s \in AC(u)$  is a *unit* segment if  $s \in AC_{ii}(u)$  for some  $1 \leq i \leq r$ . In other words a unit segment spans exactly one child  $u_i$  of  $u$ . Let  $L_i(u) = \cup_{f \leq i \leq l} AC_{fl}(u)$  denote the conceptual list of all segments that span  $u_i$ . When a unit segment  $s \in AC_{ii}(u)$  is inserted, we find the segments  $s_p$  and  $s_n$  that precede and follow  $s$  in  $L_i(u)$ ; we insert  $s$  at an arbitrary position in  $AC(u)$  so that  $s_p < s < s_n$ . It is easy to see that the correct order of segments is maintained: the correct order is maintained for the segments that span  $u_i$  and other segments are not affected.

<sup>6</sup> Probably a smaller power of log can be used, but we consider  $B \geq \log^8 n$  to simplify the analysis.



■ **Figure 5** Example from Fig. 2 revisited. Left: original segment order  $s_1 \prec s_2 \prec s_3 \prec s_4 \prec s_5 \prec s_6 \prec s_7 \prec s_8 \prec s_9$ . Right: segment  $s_6$  is deleted, the new inserted segment  $s_0$  is split into unit segments. The new segment order is e.g.,  $s_{0,3} \prec s_1 \prec s_2 \prec s_4 \prec s_5 \prec s_{0,2} \prec s_7 \prec s_8 \prec s_9 \prec s_{0,1}$ . Thus new unit segments are inserted, but the relative order of other segments does not change.

An arbitrary segment  $s$  that is to be inserted into  $AC(u)$  can be represented as  $B^\delta$  unit segments. See Fig. 5 for an example. However we cannot afford to spend  $B^\delta$  operations for an insertion. To solve this problem, we use bufferization: when a segment is inserted, we split it into  $B^\delta$  unit segments and insert them into a buffer  $\mathcal{B}$ . A complete description of the update procedure is given below.

**Buffered Insertions.** We distinguish between two categories of segments, *old* segments and *new* segments. We know the total order in the set of old segments in the portion  $\mathcal{P}(u)$  (and in the list  $AC(u)$ ). New segments are represented as a union of up to  $r$  unit segments. When the number of new segments in a portion  $\mathcal{P}(u)$  exceeds the threshold that will be specified below, we re-build  $\mathcal{P}(u)$ : we compute the order of old and new segments and declare all segments in  $\mathcal{P}(u)$  to be old.

As explained in Section 3 every portion  $\mathcal{P}(u)$  of  $AC(u)$  is stored in a biased search tree data structure. Each node of  $\mathcal{P}(u)$  has a buffer  $\mathcal{B}(\nu)$  that can store up to  $B^{3\delta}$  segments. When a new segment is inserted into  $\mathcal{P}(u)$ , we split it into unit segments and add them to the insertion buffer of  $\nu_r$ , where  $\nu_r$  is the root node of  $\mathcal{P}(u)$ . When the buffer of an internal node  $\nu$  is full, we *flush* it, i.e., we move all segments from  $\mathcal{B}(\nu)$  to buffers in the children of  $\nu$ . We keep values  $\nu.\max_{kj}[i]$ , defined in Section 3, for all internal nodes  $\nu$ . All  $\nu.\max_{kl}[\cdot]$  and all segments in  $\mathcal{B}(\nu)$  fit into one block of memory; hence we can flush the buffer of an internal node in  $O(B^\delta)$  I/Os. When the buffer of an internal node is flushed, we do not change the shape of the tree. When the buffer  $\mathcal{B}(\lambda)$  of a leaf node  $\lambda$  is full, we insert segments from  $\mathcal{B}(\lambda)$  into the set of segments stored in  $\lambda$ . If necessary we create a new leaf  $\lambda'$  and update the weights of  $\lambda$  and  $\lambda'$ . We can update the biased search tree  $\mathcal{P}(u)$  in  $O(\log n)$  time. We also update data structures  $V_i$  for  $i = 1, \dots, r$ . Since a leaf node contains the segments from at most two different groups, we can update all  $V_i$  in  $O(r)$  I/Os. The biased tree is updated in  $O(\log n)$  I/Os. The total amortized cost of a segment insertion into a portion  $\mathcal{P}(u)$  is  $O(1 + \frac{\log n+r}{B^{3\delta}} + \frac{\log_B n}{B^{2\delta}}) = O(1)$  because  $B^\delta > \log n$ .

When the number of new segments in  $\mathcal{P}(u)$  is equal to  $n_{old}/r$ , where  $n_{old}$  is the number of old segments in  $\mathcal{P}(u)$ , we rebuild  $\mathcal{P}(u)$ . Using the method from [8], we order *all* segments in  $\mathcal{P}(u)$  and update the biased tree. Sorting of segments takes  $O((n_{old}/B) \log_{M/B} n_{old}) = o(n_{old})$  I/Os. We can re-build the weighted tree  $\mathcal{P}(u)$  in  $O((n_{old}/B^{3\delta}) \log n_{old}) = o(n_{old})$  I/Os by computing the weights of leaves and inserting the leaves into the new tree one-by-one.

When a new segment  $s$  is inserted, we identify all nodes  $u_i$  where  $s$  must be stored. For every corresponding list  $AC(u_i)$ , we find the portion  $\mathcal{P}(u_i)$  where  $s$  must be stored. This takes  $O(\log_B^2 n)$  I/Os in total. Then we insert the trimmed segment  $s$  into each portion as described above. The total insertion cost is  $O(\log_B^2 n)$ . Queries are supported in the same way as in the static data structure described in Section 3. The only difference is that biased tree nodes have associated buffers. Many technical aspects are not addressed in this section. We fill in the missing details and provide the description of the data structure that also supports deletions in Section 5.

## 5 Ray Shooting for $B \geq \log^8 n$ : Fully-Dynamic Structure

Now we give a complete description of the fully-dynamic data structure for vertical ray shooting queries. Deletions are also implemented using bufferization: deleted segments are inserted into deletion buffers  $\mathcal{D}(\nu)$  that are kept in the nodes of trees  $\mathcal{P}(u)$ . Deletion buffers are processed similarly to the insertion buffers. There are, however, a number of details that were not addressed in the previous section. When a new bridge  $E_i$  is inserted we need to change weights for a number of segments. When the segment  $n(u)$  is found, we need to find the bridges  $b_p(u)$  and  $b_n(u)$ . The complete solution that addresses all these issues is more involved. First, we apply weighted search only to segments from  $E(u) = \cup_{i=1}^r E_i(u)$ . We complete the search and find the successor segment in  $AC(u)$  using some auxiliary sets stored in the nodes of  $\mathcal{P}(u)$ . Second, we use a special data structure to find the bridges  $b_p(u)$  and  $b_n(u)$ . We start by describing the changed structure of weighted trees  $\mathcal{P}(u)$ .

Segments stored in the leaves of  $\mathcal{P}(u)$  are divided into weighted and unweighted segments. Weighted segments are segments from  $E(u)$ , i.e., weighted segments are used as down-bridges. All other segments are unweighted. Every leaf contains  $\Theta(r^2)$  weighted segments. There are at  $\Omega(r^2)$  and  $O(r^4)$  unweighted segments between any two weighted segments. Hence the total number of segments in a leaf is between  $\Omega(r^4)$  and  $O(r^6)$ . Only weighted segments in a leaf have non-zero weights. Weights of weighted segments are computed in the same way as explained in Section 3. Hence the weight of a leaf  $\lambda$  is the total weight of all weighted segments in  $\lambda$ . The search for a successor of  $q$  in  $\mathcal{P}(u)$  is organized in such way that it ends in the leaf holding the successor of  $q$  in  $E(u)$ . Then we can find the successor of  $q$  in  $AC(u)$  using auxiliary data stored in the nodes of  $\mathcal{P}(u)$ .

We keep the following auxiliary sets and buffers in nodes  $\nu$  of every weighted tree  $\mathcal{P}(u)$ . Let  $AC_{fl}(u, \nu)$  denote the set of segments from  $AC_{fl}(u)$  that are stored in leaf descendants of a node  $\nu$ .

- (i) Sets  $\text{Max}_{fl}(\nu)$  and  $\text{Min}_{fl}(\nu)$  for all  $f, l$  such that  $1 \leq f \leq l \leq r$  and for all nodes  $\nu$ .  $\text{Max}_{fl}(\nu)$  ( $\text{Min}_{fl}(\nu)$ ) contains  $\min(r^4, |AC_{fl}(u, \nu)|)$  highest (lowest) segments from  $AC_{fl}(u, \nu)$ . For every segment  $s$  in sets  $\text{Max}_{fl}(\nu)$  and  $\text{Min}_{fl}(\nu)$  we record the index  $i$  such that  $s \in E_i(u)$  (or NULL if  $s$  is not a bridge segment).
- (ii) The set  $\text{Nav}(\nu)$  for an internal node  $\nu$  is the union of all sets  $\text{Max}_{fl}(\nu_i)$  and  $\text{Min}_{fl}(\nu_i)$  for all children  $\nu_i$  of  $\nu$ .
- (iii) The set  $\text{Max}'_{fl}(\nu)$ ,  $1 \leq f \leq l \leq r$  contains highest segments from  $AC_{fl}(u, \nu)$  that are not stored in any set  $\text{Max}'_{fl}(u, \mu)$  for an ancestor  $\mu$  of  $\nu$ . Either  $\text{Max}'_{fl}(\nu)$  holds at least  $r^4$  and at most  $2r^4$  segments or  $\text{Max}'_{fl}(\nu)$  holds less than  $r^4$  segments and  $\text{Max}'_{fl}(\rho)$  for all descendants  $\rho$  of  $\nu$  are empty. In other words,  $\text{Max}'_{fl}(\cdot)$  are organized as external priority search trees [6]. The set  $\text{Min}'_{fl}(\nu)$  is defined in the same way with respect to the lowest segments. We use  $\text{Max}'$  and  $\text{Min}'$  to maintain sets Max and Min.
- (iv) Finally we keep an insertion buffer  $\mathcal{B}(\nu)$  and a deletion buffer  $\mathcal{D}(\nu)$  in every node  $\nu$ .

**Deletions.** If an old segment  $s$  is deleted, we insert it into the deletion buffer  $\mathcal{D}(\nu_R)$  of the root node  $\nu_R$ . If a new segment  $s$  is deleted, we split  $s$  into  $O(r)$  unit segments and insert them into  $\mathcal{D}(\nu_R)$ . When one or more segments are inserted into  $\mathcal{D}(\nu_r)$ , we also update sets  $\text{Max}_{fl}(\nu_R)$  and  $\text{Min}_{fl}(\nu_R)$ . For any node  $\nu \in \mathcal{P}(u)$ , when the number of segments in  $\mathcal{D}(\nu)$  exceeds  $r^3$ , we flush both  $\mathcal{D}(\nu)$  and  $\mathcal{B}(\nu)$  using the following procedure. First we identify segments  $s \in \mathcal{B}(\nu) \cap \mathcal{D}(\nu)$  and remove such  $s$  from both  $\mathcal{B}(\nu)$  and  $\mathcal{D}(\nu)$ . Next we move segments from  $\mathcal{B}(\nu)$  and  $\mathcal{D}(\nu)$  to buffers  $\mathcal{B}(\nu_i)$  and  $\mathcal{D}(\nu_i)$  in the children  $\nu_i$  of  $\nu$ . For every child  $\nu_i$  of  $\nu$ , first we update sets  $\text{Max}'_{fl}(\nu_i)$  by removing segments from  $\mathcal{D}(\nu_i)$  (resp. inserting segments from  $\mathcal{B}(\nu_i)$ ) if necessary. Then we take care that the size of  $\text{Max}'_{fl}(\nu_i)$  is not too small. If some  $\text{Max}'_{fl}(\nu_i)$  contains less than  $r^4$  segments and more than 0 segments, we move up segments from the children of  $\nu_i$  into  $\nu_i$ , so that the total size of  $\text{Max}'_{fl}(\nu_i)$  becomes equal to  $2r^4$  or all segments are moved from the corresponding sets  $\text{Max}'_{fl}(\cdot)$  in the children of  $\nu_i$  into  $\text{Max}'_{fl}(\nu_i)$ . We recursively update  $\text{Max}'_{fl}(\cdot)$  in each child of  $\nu_i$  using the same procedure.

Next, we update sets  $\text{Max}_{fl}(\nu_i)$ . We compute  $\mathcal{M}_{fl} = \cup \text{Max}'_{fl}(\mu)$  where the union is taken over all proper ancestors  $\mu$  of  $\nu$ . Every segment in  $\text{Max}_{fl}(\nu)$  is either from  $\text{Max}'_{fl}(\nu)$  or from  $\text{Max}'_{fl}(\mu)$  for a proper ancestor  $\mu$  of  $\nu$ . Hence we can compute all  $\text{Max}_{fl}(\nu_i)$  when  $\mathcal{M}_{fl}$  and  $\text{Max}'_{fl}(\nu_i)$  are known. Sets  $\text{Min}'_{fl}(\nu_i)$  and  $\text{Min}_{fl}(\nu_i)$  are updated in the same way. Finally we update the set  $\text{Nav}(\nu)$  by collecting segments from  $\text{Max}_{fl}(\nu_i)$  and  $\text{Min}_{fl}(\nu_i)$ .

All segments needed to re-compute sets after flushing buffers  $\mathcal{D}(\nu)$  and  $\mathcal{B}(\nu)$  fit into one block of space. Hence we can compute the set  $\mathcal{M}$  in  $O(\log_B n) = O(r)$  I/Os and all sets in each node  $\nu_i$  in  $O(1)$  I/Os. The set  $\text{Nav}(\nu)$  is updated in  $O(r)$  I/Os. Since each node has  $O(r)$  children, the total number of I/Os needed to flush a buffer is  $O(r)$ . Every segment can be divided into up to  $r$  unit segments and each unit segment can contribute to  $\log_B n$  buffer flushes. Hence the total amortized cost per segment is  $O(\frac{r^2 \log_B n}{r^3}) = O(1)$ . We did not yet take into account the cost of refilling the buffers  $\text{Max}'$ ; using the analysis similar to the analysis in [12, Section 4], we can estimate the cost of re-filling  $\text{Max}'$  as  $O(\frac{\log_B n}{r^3}) = o(1)$ .

We do not store buffers in the leaf nodes. Let  $S(\lambda)$  be the set of segments kept in a leaf  $\lambda$  and let  $S_W(\lambda)$  be the set of weighted segments stored in  $\lambda$ . When we move segments from  $\mathcal{B}(\nu)$  or  $\mathcal{D}(\nu)$  to its leaf child  $\lambda$ , we update  $S(\lambda)$  accordingly. This operation changes the weight of  $\lambda$ . Hence we need to update the weighted tree  $\mathcal{P}(u)$  in  $O(\log n)$  I/Os. Sets  $\text{Max}_{fl}(\cdot)$  and  $\text{Min}_{fl}(\cdot)$  are also updated.

After an insertion of new segments into a leaf node, we may have to insert or remove some bridges in  $E_i(u)$  for  $1 \leq i \leq r$ . When we insert a new bridge  $b$  into  $E_i(u)$ , we must split some portion  $\mathcal{P}(u_i)$  into two new portions,  $\mathcal{P}_1(u_i)$  and  $\mathcal{P}_2(u_i)$ . Additionally we must change the weights of the bridge segments in  $E_i(u)$  that precede and follow  $b$ . The cost of splitting  $\mathcal{P}(u_i)$  is  $O(\log n)$ . We also need  $O(\log n)$  I/Os to change the weights of two neighbor bridges. Hence the total cost of inserting a new bridge is  $O(\log n)$ . We insert a bridge at most once per  $O(r)$  insertions into  $AC(u)$  because every new segment is divided into up to  $r$  unit segments. We remove a bridge at most once after  $O(r)$  deletions. See [13] for the description of the method to maintain bridges in catalogs  $AC(u)$ . Thus the total amortized cost incurred by a bridge insertion or deletion is  $O(\frac{\log n}{r}) = O(1)$ .

**Insertions.** Insertions are executed in a similar way. A new inserted segment is split into  $O(r)$  unit segments that are inserted into the buffer  $\mathcal{B}(\nu_R)$  for the root node  $\nu_R$ . The buffers and auxiliary sets are updated and flushed in the same way as in the case of deletions. When the number of new segments in some portion  $\mathcal{P}(u)$  is equal to  $n_{\text{old}}/r$ , where  $n_{\text{old}}$  is the number of old segments in  $\mathcal{P}(u)$ , we rebuild  $\mathcal{P}(u)$ . As explained in Section 4, rebuilding of  $\mathcal{P}(u)$  incurs an amortized cost of  $o(1)$ .

**Queries.** The search for the successor segment  $n(u)$  in the weighted tree  $\mathcal{P}(u)$  consists of two stages. Suppose that the query point  $q$  is in the slab of the  $i$ -th child  $u_i$  of  $u$ . First we find the successor  $b_n(u)$  of  $q$  in  $E_i(u)$  by searching in  $\mathcal{P}(u)$ . We traverse the path from the root to the leaf  $\lambda_n$  holding  $b_n(u)$ . In every node  $\nu$  we select its leftmost child  $\nu_j$ , such that  $\text{Max}_{fl}(\nu_j)$  for some  $f \leq i \leq l$  contains a segment  $s$  that is above  $q$  and  $s$  is not deleted (i.e.,  $s \notin \mathcal{D}(\mu)$  for all ancestors  $\mu$  of  $\nu$ ). The size of each set  $\text{Max}_{fl}(\nu_k)$  is larger than the total size of all  $\mathcal{D}(\mu)$  in all ancestors  $\mu$  of  $\nu$ . Hence every  $\text{Max}_{fl}(\nu_i)$  contains some elements that are not deleted unless the set  $C_{fl}(u, \nu_i)$  is empty. Therefore we select the correct child  $\nu_j$  in every node. Since  $\mathcal{P}(u)$  is a biased search tree [10, 19], the total cost of finding the leaf  $\lambda_n$  is bounded by  $O(\log(W_P/\omega_\lambda)) = O(\log(W_P/\omega_n))$  where  $\omega_\lambda$  is the total weight of all segments in  $\lambda_n$  and  $\omega_n \leq \omega_\lambda$  is the weight of the bridge segment  $b_n(u)$ .

During the second stage we need to find the successor segment  $n(u)$  of  $q$  in  $AC(u)$ . The distance between  $n(u)$  and  $b_n(u)$  in  $AC(u)$  can be arbitrarily large. Nevertheless  $n(u)$  is stored in one of the sets  $\text{Nav}(\mu)$  for some ancestor  $\mu$  of  $\lambda_n$ . Suppose that  $n(u)$  is an unweighted segment stored in a leaf  $\lambda'$  of  $\mathcal{P}(u)$  and let  $\mu$  denote the lowest common ancestor of  $\lambda$  and  $\lambda'$ . Let  $\mu_k$  be the child of  $\mu$  that is an ancestor of  $\lambda'$ . There are at most  $r^4$  segments in  $AC_{fl}(u)$  between  $n(u)$  and  $b_n(u)$ . Hence,  $n(u)$  is stored in the set  $\text{Max}_{fl}(\mu_k)$ . Hence,  $n(u)$  is also stored in  $\text{Nav}(\mu)$ . We visit all ancestors  $\mu$  of  $\lambda_n$  and compute  $\mathcal{D} = \cup_{\mu} \mathcal{D}(\mu)$ . Then we visit all ancestors one more time and find the successor of  $q$  in  $\text{Nav}(\mu) \setminus \mathcal{D}$ . The asymptotic query cost remains the same because we only visit the nodes between  $\lambda_n$  and the root and each node is visited a constant number of times.

We need to consider one additional special case. It is possible that there are no bridge segments  $s \in E_i(u)$  stored in the leaves of  $\mathcal{P}(u)$ . In this case there are at most  $r^2$  segments in  $AC_{fl}(u)$  for every pair  $f, l$ , satisfying  $f \leq i \leq l$ , stored in the leaves of  $\mathcal{P}(u)$ . For each portion  $\mathcal{P}(u)$ , if there are at most  $r^2$  segments in  $AC'_{fl}(u) \cap \mathcal{P}(u)$ , we keep the list of all such segments. All such lists fit into one block of memory. We also keep the list of indexes  $i$ , such that  $E_i(u) \cap \mathcal{P}(u)$  is empty. Suppose that we need to find the successor of  $q$  and  $\mathcal{P}(u) \cap E_i(u)$  is empty. Then we simply examine all segments in  $AC_{fl}(u) \cap \mathcal{P}(u)$  for all  $f \leq i \leq l$  and find the successor of  $q$  in  $O(1)$  I/Os.

When  $n(u)$  is known, we need to find  $b_p(u)$  and  $b_n(u)$ , if  $b_n(u)$  was not computed at the previous step. It is not always possible to find these bridges using  $\mathcal{P}(u)$  because  $b_p(u)$  and  $b_n(u)$  can be outside of  $\mathcal{P}(u)$ . To this end, we use the data structure for colored union-split-find problem on a list (list-CUSF) that will be described in the full version of this paper [24]. We keep the list  $V(u)$  containing all down-bridges from  $E_i(u)$ , for  $1 \leq i \leq r$ , and all up-bridges from  $UP(u)$ . Each segment in  $e \in V(u)$  is associated to an interval; a segment  $e \in V_i(u)$  is associated to an interval  $[i, i]$  and a segment from  $UP(u)$  is associated to a dummy interval  $[-1, -1]$ . For any segment  $e \in V(u)$  we can find the preceding/following segment associated to an interval  $[i, i]$  for any  $i$ ,  $1 \leq i \leq r$ , in  $O(\log \log_B n)$  I/Os. Updates of  $V(u)$  are supported in  $O(\log \log_B n)$  I/Os. Since we insert or remove bridge segments once per  $r^2$  updates, the amortized cost of maintaining the list-CUSF structure is  $O(1)$ .

**Summing up.** By the same argument as in Section 3, weighted searches in all nodes take  $O(\log_B n)$  I/Os in total. Additionally we spend  $(\log \log_B n)$  I/Os in every node with a query to list-CUSF. Thus the total query cost is  $O(\log_B n \log \log_B n)$ . When a segment is deleted, we remove it from  $O(\log_B n)$  lists  $AC(u)$  and from secondary structures (weighted trees etc.) in these nodes. The deletions take  $O(1)$  I/Os per node or  $O(\log_B n)$  I/Os in total. When a segment is inserted, it must be inserted into  $O(\log_B n)$  lists  $AC(u)$ . We first have to spend  $O(\log_B n)$  I/Os to find the portion  $\mathcal{P}(u)$  of each  $AC(u)$  where it must be stored. When  $\mathcal{P}(u)$



is known, an insertion takes  $O(1)$  amortized I/Os as described above. The total cost of an insertion is  $O(\log_B^2 n)$  I/Os. Since every segment is stored in  $O(\log_B n)$  lists, the total space is  $O(n \log_B n)$ .

► **Lemma 2.** *If  $B > \log^8 n$ , then there exists an  $O(n \log_B n)$  space data structure that supports vertical ray shooting queries on a dynamic set of  $n$  non-intersecting segments in  $O(\log_B n \log \log_B n)$  I/Os. Insertions and deletions of segments are supported in  $O(\log_B^2 n)$  and  $O(\log_B n)$  amortized I/Os respectively.*

## 6 Faster Insertions

When a new segment  $s$  is inserted into our data structure, we need to find the position of  $s$  in  $O(\log_B n)$  lists  $AC(u)$  (to be precise, we need to know the portion  $\mathcal{P}(u)$  of  $AC(u)$  that contains  $s$ ). When positions of  $s$  in  $AC(u)$  are known, we can finish the insertion in  $O(\log_B n)$  I/Os. In order to speed-up insertions, we use the multi-colored segment tree of Chan and Nekrich [13]. Segments in lists  $C(u)$  are assigned colors  $\chi$ , so that the total number of different colors is  $O(\log H)$  where  $H = O(\log_B n)$  is the height of the segment tree. Let  $C_\chi(u)$  denote the set of segments of color  $\chi$  in  $C(u)$ . We apply the technique of Sections 3- 5 to each color separately. That is, we create augmented lists  $AC_\chi(u)$  and construct weighted search trees  $\mathcal{P}_\chi(u)$  for each color separately. The query cost is increased by factor  $O(\log H)$ , the number of colors. The deletion cost is also increased by  $O(\log H)$  factor because we update the data structure for each color separately. When a new segment  $s$  is inserted, we insert it into some lists  $AC_{\chi_i}(u_i)$  where  $u_i$  is the node such that  $s$  spans  $u_i$  but does not span its parent and  $\chi_i$  is some color (the same segment can be assigned different colors  $\chi_i$  in different nodes  $u_i$ ). We can find the position of  $s$  in all  $AC_{\chi_i}(u_i)$  with  $O(\log_B n \log H + H \cdot t_{\text{usf}}) = O(\log_B n \log \log_B n)$  I/Os where  $t_{\text{usf}} = O(\log \log_B n)$  is the query cost in a union-split-find data structure in the external memory model. See [13] for a detailed description.

► **Lemma 3.** *If  $B > \log^8 n$ , then there exists an  $O(n \log_B n)$  space data structure that supports vertical ray shooting queries on a dynamic set of non-intersecting segments in  $O(\log_B n (\log \log_B n)^2)$  I/Os. Insertions and deletions of segments can be supported in  $O(\log_B n \log \log_B n)$  amortized I/Os.*

## 7 Missing Details

Using the method from [13] we can reduce the space usage of our data structure to linear at the cost of increasing the query and update complexity by  $O(\log \log_B n)$  factor. The resulting data structure supports queries in  $O(\log_B n (\log \log_B n)^2)$  I/Os and updates in  $O(\log_B n (\log \log_B n)^3)$  amortized I/Os. Details will be provided in the full version [24].

In our exposition we assumed for simplicity that the tree  $\mathcal{T}$  does not change, i.e., the set of  $x$ -coordinates of segment endpoints is fixed and known in advance. To support insertions of new  $x$ -coordinate, we can replace the static tree  $\mathcal{T}$  with a weight-balanced tree with node degree  $\Theta(r) = \Theta(B^\delta)$ . We also assumed that the block size  $B$  is large,  $B > \log^8 n$ . If  $B \leq \log^8 n$ , the linear-space internal memory data structure [13] achieves  $O(\log n (\log \log n)^2) = O(\log_B n (\log \log_B n)^3)$  query cost and  $O(\log n \log \log n) = O(\log_B n (\log \log_B n)^2)$  update cost because  $\log n = O(\log_B n \log \log_B n)$  and  $\log \log n = O(\log \log_B n)$  for  $B \leq \log^8 n$ . Thus we obtain our main result.

► **Theorem 4.** *There exists an  $O(n)$  space data structure that supports vertical ray shooting queries on a dynamic set of  $n$  non-intersecting segments in  $O(\log_B n (\log \log_B n)^3)$  I/Os. Insertions and deletions of segments are supported in  $O(\log_B n (\log \log_B n)^2)$  amortized I/Os.*

---

## References

- 1 Pankaj K. Agarwal, Lars Arge, Gerth Stølting Brodal, and Jeffrey Scott Vitter. I/O-efficient dynamic point location in monotone planar subdivisions. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, pages 11–20, 1999.
- 2 Alok Aggarwal and Jeffrey Scott Vitter. The Input/Output Complexity of Sorting and Related Problems. *Commun. ACM*, 31(9):1116–1127, 1988. doi:10.1145/48529.48535.
- 3 Lars Arge, Gerth Stølting Brodal, and Loukas Georgiadis. Improved dynamic planar point location. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 305–314, 2006. doi:10.1109/FOCS.2006.40.
- 4 Lars Arge, Gerth Stølting Brodal, and S. Srinivasa Rao. External Memory Planar Point Location with Logarithmic Updates. *Algorithmica*, 63(1-2):457–475, 2012. doi:10.1007/s00453-011-9541-2.
- 5 Lars Arge, Andrew Danner, and Sha-Mayn Teh. I/O-efficient point location using persistent B-trees. *ACM Journal of Experimental Algorithmics*, 8, 2003. doi:10.1145/996546.996549.
- 6 Lars Arge, Vasilis Samoladas, and Jeffrey Scott Vitter. On Two-Dimensional Indexability and Optimal Range Search Indexing. In *Proc. 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 346–357, 1999. doi:10.1145/303976.304010.
- 7 Lars Arge and Jan Vahrenhold. I/O-efficient dynamic planar point location. *Computational Geometry*, 29(2):147–162, 2004. doi:10.1016/j.comgeo.2003.04.001.
- 8 Lars Arge, Darren Erik Vengroff, and Jeffrey Scott Vitter. External-Memory Algorithms for Processing Line Segments in Geographic Information Systems. *Algorithmica*, 47(1):1–25, 2007. doi:10.1007/s00453-006-1208-z.
- 9 Hanna Baumgarten, Hermann Jung, and Kurt Mehlhorn. Dynamic point location in general subdivisions. *J. Algorithms*, 17(3):342–380, 1994. doi:10.1006/jagm.1994.1040.
- 10 Samuel W. Bent, Daniel Dominic Sleator, and Robert Endre Tarjan. Biased Search Trees. *SIAM J. Comput.*, 14(3):545–568, 1985. doi:10.1137/0214041.
- 11 Jon Louis Bentley. Algorithms for Klee’s rectangle problems. Unpublished manuscript, Department of Computer Science, Carnegie-Mellon University, 1977.
- 12 Gerth Stølting Brodal. External Memory Three-Sided Range Reporting and Top-k Queries with Sublogarithmic Updates. In *Proc. 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 23:1–23:14, 2016. doi:10.4230/LIPIcs.STACS.2016.23.
- 13 Timothy M. Chan and Yakov Nekrich. Towards an Optimal Method for Dynamic Planar Point Location. In *Proc. 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 390–409, 2015. doi:10.1109/FOCS.2015.31.
- 14 Timothy M. Chan and Konstantinos Tsakalidis. Dynamic Planar Orthogonal Point Location in Sublogarithmic Time. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry (SoCG 2018)*, volume 99 of *LIPIcs*, pages 25:1–25:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.SocG.2018.25.
- 15 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986. doi:10.1007/BF01840440.
- 16 Siu-Wing Cheng and Ravi Janardan. New results on dynamic planar point location. *SIAM J. Comput.*, 21(5):972–999, 1992. doi:10.1137/0221057.
- 17 Yi-Jen Chiang, Franco P. Preparata, and Roberto Tamassia. A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM J. Comput.*, 25(1):207–233, 1996. doi:10.1137/S0097539792224516.

- 18 Yi-Jen Chiang and Roberto Tamassia. Dynamization of the trapezoid method for planar point location in monotone subdivisions. *Int. J. Comput. Geometry Appl.*, 2(3):311–333, 1992. doi:10.1142/S0218195992000184.
- 19 Joan Feigenbaum and Robert Endre Tarjan. Two New Kinds of Biased Search Trees. *Bell Systems Technical Journal*, 62(10):3139–3158, 1983.
- 20 Yoav Giora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms*, 5(3):28, 2009. doi:10.1145/1541885.1541889.
- 21 Michael T. Goodrich and Roberto Tamassia. Dynamic trees and dynamic point location. *SIAM J. Comput.*, 28(2):612–636, 1998. doi:10.1137/S0097539793254376.
- 22 Michael T. Goodrich, Jyh-Jong Tsay, Darren Erik Vengroff, and Jeffrey Scott Vitter. External-Memory Computational Geometry (Preliminary Version). In *Proc. 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 714–723, 1993. doi:10.1109/SFCS.1993.366816.
- 23 Kurt Mehlhorn and Stefan Näher. Dynamic fractional cascading. *Algorithmica*, 5(2):215–241, 1990. doi:10.1007/BF01840386.
- 24 J. Ian Munro and Yakov Nekrich. Dynamic Planar Point Location in External Memory. *CoRR*, abs/1903.06601, 2019. arXiv:1903.06601.
- 25 Franco P. Preparata and Roberto Tamassia. Fully dynamic point location in a monotone subdivision. *SIAM J. Comput.*, 18(4):811–830, 1989. doi:10.1137/0218056.
- 26 Franco P. Preparata and Roberto Tamassia. Efficient point location in a convex spatial cell-complex. *SIAM J. Comput.*, 21(2):267–280, 1992. doi:10.1137/0221020.



# Efficient Algorithms for Ortho-Radial Graph Drawing

Benjamin Niedermann 

University of Bonn, Germany  
niedermann@uni-bonn.de

Ignaz Rutter 

University of Passau, Germany  
rutter@fim.uni-passau.de

Matthias Wolf 

Karlsruhe Institute of Technology, Germany  
matthias.wolf@kit.edu

---

## Abstract

Orthogonal drawings, i.e., embeddings of graphs into grids, are a classic topic in Graph Drawing. Often the goal is to find a drawing that minimizes the number of bends on the edges. A key ingredient for bend minimization algorithms is the existence of an *orthogonal representation* that allows to describe such drawings purely combinatorially by only listing the angles between the edges around each vertex and the directions of bends on the edges, but neglecting any kind of geometric information such as vertex coordinates or edge lengths.

Barth et al. [2] have established the existence of an analogous *ortho-radial representation* for *ortho-radial drawings*, which are embeddings into an ortho-radial grid, whose gridlines are concentric circles around the origin and straight-line spokes emanating from the origin but excluding the origin itself. While any orthogonal representation admits an orthogonal drawing, it is the circularity of the ortho-radial grid that makes the problem of characterizing valid ortho-radial representations all the more complex and interesting. Barth et al. prove such a characterization. However, the proof is existential and does not provide an efficient algorithm for testing whether a given ortho-radial representation is valid, let alone actually obtaining a drawing from an ortho-radial representation.

In this paper we give quadratic-time algorithms for both of these tasks. They are based on a suitably constrained left-first DFS in planar graphs and several new insights on ortho-radial representations. Our validity check requires quadratic time, and a naive application of it would yield a quartic algorithm for constructing a drawing from a valid ortho-radial representation. Using further structural insights we speed up the drawing algorithm to quadratic running time.

**2012 ACM Subject Classification** Mathematics of computing → Graph algorithms

**Keywords and phrases** Graph Drawing, Ortho-Radial Graph Drawing, Ortho-Radial Representation, Topology-Shape-Metrics, Efficient Algorithms

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.53

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1903.05048>.

**Funding** *Matthias Wolf*: Matthias Wolf was funded by the Helmholtz Program Storage and Cross-linked Infrastructures, Topic 6 Superconductivity, Networks and System Integration.

## 1 Introduction

Grid drawings of graphs embed graphs into grids such that vertices map to grid points and edges map to internally disjoint curves on the grid lines that connect their endpoints. Orthogonal grids, whose grid lines are horizontal and vertical lines, are popular and widely used in graph drawing. Among others, orthogonal graph drawings are applied in VLSI design (e.g., [31, 6]), diagrams (e.g., [4, 19, 14, 33]), and network layouts (e.g., [27, 23]). They have



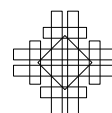
© Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

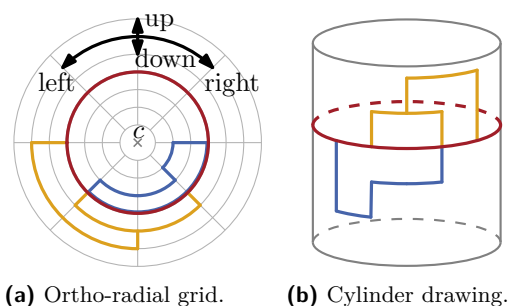
Editors: Gill Barequet and Yusu Wang; Article No. 53; pp. 53:1–53:14

Leibniz International Proceedings in Informatics

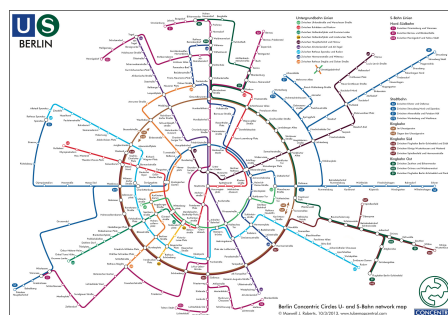


LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** An ortho-radial drawing of a graph on a grid (a) and its equivalent interpretation as an orthogonal drawing on a cylinder (b).



■ **Figure 2** Metro map of Berlin using an ortho-radial layout<sup>1</sup>. Image copyright by Maxwell J. Roberts. Reproduced with permission.

been extensively studied with respect to their construction and properties (e.g., [30, 7, 8, 26, 1]). Moreover, they have been generalized to arbitrary planar graphs with degree higher than four (e.g., [29, 18, 9]).

Ortho-radial drawings are a generalization of orthogonal drawings to grids that are formed by concentric circles and straight-line spokes from the center but excluding the center. Equivalently, they can be viewed as graphs drawn in an orthogonal fashion on the surface of a standing cylinder, see Figure 1, or a sphere without poles. Hence, they naturally bring orthogonal graph drawings to the third dimension.

Among other applications, ortho-radial drawings are used to visualize network maps; see Figure 2. Especially, for metro systems of metropolitan areas they are highly suitable. Their inherent structure emphasizes the city center, the metro lines that run in circles as well as the metro lines that lead to suburban areas. While the automatic creation of metro maps has been extensively studied for other layout styles (e.g., [22, 25, 32, 17]), this is a new and wide research field for ortho-radial drawings.

Adapting existing techniques and objectives from orthogonal graph drawings is a promising step to open up that field. One main objective in orthogonal graph drawing is to minimize the number of bends on the edges. The core of a large fraction of the algorithmic work on this problem is the *orthogonal representation*, introduced by Tamassia [28], which describes orthogonal drawings listing (i) the angles formed by consecutive edges around each vertex and (ii) the directions of bends along the edges. Such a representation is *valid* if (I) the angles around each vertex sum to  $360^\circ$ , and (II) the sum of the angles around each face with  $k$  vertices is  $(k - 2) \cdot 180^\circ$  for internal faces and  $(k + 2) \cdot 180^\circ$  for the outer face. The necessity of the first condition is obvious and the necessity of the latter follows from the sum of inner/outer angles of any polygon with  $k$  corners. It is thus clear that any orthogonal drawing yields a valid orthogonal representation, and Tamassia [28] showed that the converse holds true as well; for a valid orthogonal representation there exists a corresponding orthogonal drawing that realizes this representation. Moreover, the proof is constructive and allows the efficient construction of such a drawing, a process that is referred to as *compaction*.

Altogether this enables a three-step approach for computing orthogonal drawings, the so-called *Topology-Shape-Metrics Framework*, which works as follows. First, fix a *topology*, i.e., combinatorial embedding of the graph in the plane (possibly planarizing it if it is non-planar);

<sup>1</sup> Note that ortho-radial drawings exclude the center of the grid, which is slightly different to the concentric circles maps by Maxwell J. Roberts.

second, determine the *shape* of the drawing by constructing a valid orthogonal representation with few bends; and finally, compactify the orthogonal representation by assigning suitable vertex coordinates and edge lengths (*metrics*). As mentioned before, this reduces the problem of computing an orthogonal drawing of a planar graph with a fixed embedding to the purely combinatorial problem of finding a valid orthogonal representation, preferably with few bends. The task of actually creating a corresponding drawing in polynomial time is then taken over by the framework. It is this approach that is at the heart of a large body of literature on bend minimization algorithms for orthogonal drawings (e.g., [5, 15, 13, 16, 10, 11, 12]).

Very recently Barth et al. [2] proposed a generalization of orthogonal representations to ortho-radial drawings, called *ortho-radial* representations, with the goal of establishing an ortho-radial analogue of the TSM framework for ortho-radial drawings. They show that a natural generalization of the validity conditions (I) and (II) above is not sufficient, and introduce a third, less local condition that excludes so-called *monotone cycles*, which do not admit an ortho-radial drawing. They show that these three conditions together fully characterize ortho-radial drawings. Before that, characterizations for bend-free ortho-radial drawings were only known for paths, cycles and theta graphs [21]. Further, for the special case that each internal face is a rectangle, a characterization for cubic graphs was known [20].

With the result by Barth et al. finding an ortho-radial drawing for a planar graph with fixed-embedding reduces to the purely combinatorial problem of finding a valid ortho-radial representation. In particular, since bends can be seen as additionally introduced vertices subdividing edges, finding an ortho-radial drawing with minimum number of bends reduces to finding a valid ortho-radial representation with minimum number of such additionally introduced vertices. In this sense, the work by Barth et al. constitutes a major step towards computing ortho-radial drawings with minimum number of bends.

Yet, it is here where their work still contains a major gap. While the work of Barth et al. shows that valid ortho-radial representations fully characterize ortho-radial drawings, it is unclear if it can be checked efficiently whether a given ortho-radial representation is valid. Moreover, while their existential proof of a corresponding drawing is constructive, it needs to repeatedly test whether certain ortho-radial representations are valid.

**Contribution and Outline.** We develop such a test running in quadratic time, thus implementing the compaction step of the TSM framework with polynomial running time. While this does not yet directly allow us to compute ortho-radial drawings with few bends, our result paves the way for a purely combinatorial treatment of bend minimization in ortho-radial drawings, thus enabling the same type of tools that have proven highly successful in minimizing bends in orthogonal drawings.

At the core of our validity testing algorithm are several new insights into the structure of ortho-radial representations. The algorithm itself is a left-first DFS that uses suitable constraints to determine candidates for monotone cycles in such a way that if a given ortho-radial representation contains a monotone cycle, then one of the candidates is monotone. While it may be obvious to use a DFS for finding cycles in general, it is far from clear how such a search works for monotone cycles in ortho-radial representations. Plugging this test as a black box into the drawing algorithm of Barth et al. yields an  $O(n^4)$ -time algorithm for computing a drawing from a valid ortho-radial representation, where  $n$  is the number of vertices. Using further structural insights on the augmentation process we improve the running time of this algorithm to  $O(n^2)$ . Hence, our result is not only of theoretical interest, but the algorithm can be actually deployed. We believe that the algorithm is a useful intermediate step for providing initial network layouts to map designers and layout algorithms such as force directed algorithms; see also Section 5.



In Section 2 we present preliminaries that are used throughout the paper. First we formally define ortho-radial representations and recall the most important results from [2]. Afterwards we show that for the purpose of validity checking and determining the existence of a monotone cycle, we can restrict ourselves to so-called *normalized instances*. In Section 3 we give a validity test for ortho-radial representations that runs in  $O(n^2)$  time. Afterwards, in Section 4, we revisit the rectangulation procedure from [2] and show that using the techniques from Section 3 it can be implemented to run in  $O(n^2)$  time, improving over a naive application which would yield running time  $O(n^4)$ . Together with [2] this enables a purely combinatorial treatment of ortho-radial drawings. We conclude with a summary and some open questions in Section 5.

## 2 Preliminaries

We first formally introduce ortho-radial drawings and ortho-radial representations. Afterwards we present two transformations that we use to simplify the discussion of symmetric cases.

### 2.1 Ortho-Radial Drawings and Representations

We use the same definitions and conventions on ortho-radial drawings as presented by Barth et al. [2]; for the convenience of the reader we briefly repeat them here. In particular, we only consider drawings and representations without bends on the edges. As argued in [2], this is not a restriction, since it is always possible to transform a drawing/representation with bends into one without bends by subdividing edges so that a vertex is placed at each bend.

We are given a planar 4-graph  $G = (V, E)$  with  $n$  vertices and fixed embedding, where a graph is a 4-graph if it has only vertices with degree at most four. We define that a path  $P$  in  $G$  is always simple, while a cycle  $C$  may contain vertices multiple times but may not cross itself. All cycles are oriented clockwise, so that their interiors are locally to the right. A cycle is part of its interior and exterior. We denote the subpath of  $P$  from  $u$  to  $v$  by  $P[u, v]$  assuming that  $u$  and  $v$  are included. For any path  $P = v_1, \dots, v_k$  its *reverse* is  $\bar{P} = v_k, \dots, v_1$ . The concatenation of two paths  $P_1$  and  $P_2$  is written as  $P_1 + P_2$ . For a cycle  $C$  in  $G$  that contains any edge at most once, the subpath  $C[e, e']$  between two edges  $e$  and  $e'$  on  $C$  is the unique path on  $C$  that starts with  $e$  and ends with  $e'$ . If the start vertex  $u$  of  $e$  is contained in  $C$  only once, we also write  $C[u, e']$ , because then  $e$  is uniquely defined by  $u$ . Similarly, if the end vertex  $v$  of  $e'$  is contained in  $C$  only once, we also write  $C[e, v]$ . We also use this notation to refer to subpaths of simple paths.

In an ortho-radial drawing  $\Delta$  of  $G$  each edge is directed and drawn either clockwise, counter-clockwise, towards the center or away from the center. Hence, using the metaphor of a cylinder, the edges point *right*, *left*, *down* or *up*, respectively. Moreover, *horizontal edges* point left or right, while *vertical edges* point up or down; see Figure 1.

We distinguish two types of simple cycles. If the center of the grid lies in the interior of a simple cycle, the cycle is *essential* and otherwise *non-essential*. Further, there is an unbounded face in  $\Delta$  and a face that contains the center of the grid; we call the former the *outer face* and the latter the *central face*; in our drawings we mark the central face using a small “x”. All other faces are *regular*.

For two edges  $uv$  and  $vw$  incident to the same vertex  $v$ , we define the *rotation*  $\text{rot}(uvw)$  as 1 if there is a right turn at  $v$ , 0 if  $uvw$  is straight and  $-1$  if there is a left turn at  $v$ . In the special case that  $u = w$ , we have  $\text{rot}(uvw) = -2$ .

The rotation of a path  $P = v_1, \dots, v_k$  is the sum of the rotations at its internal vertices, i.e.,  $\sum_{i=2}^{k-1} \text{rot}(v_{i-1}v_iv_{i+1})$ . Similarly, for a cycle  $C = v_1, \dots, v_k, v_1$ , its rotation is the sum of the rotations at all its vertices (where we define  $v_0 = v_k$  and  $v_{k+1} = v_1$ ), i.e.,

$\text{rot}(C) = \sum_{i=1}^k \text{rot}(v_{i-1}v_iv_{i+1})$ . We observe that  $\text{rot}(P) = \text{rot}(P[s, e]) + \text{rot}(P[e, t])$  for any path  $P$  from  $s$  to  $t$  and any edge  $e$  on  $P$ . Further, we have  $\text{rot}(\overline{P}) = -\text{rot}(P)$ . For a face  $f$  we use  $\text{rot}(f)$  to denote the rotation of the facial cycle that bounds  $f$  (oriented such that  $f$  lies on the right side of the cycle).

As introduced by Barth et al. [2], an *ortho-radial representation*  $\Gamma$  of a 4-planar graph  $G$  fixes the central and outer face of  $G$  as well as a reference edge  $e^*$  on the outer face such that the outer face is locally to the left of  $e^*$ . Following the convention established by Barth et al. [2] the reference edge always points right. Further,  $\Gamma$  specifies for each face  $f$  of  $G$  a list  $H(f)$  that contains for each edge  $e$  of  $f$  the pair  $(e, a)$ , where  $a \in \{90^\circ, 180^\circ, 270^\circ, 360^\circ\}$ . The interpretation of  $(e, a)$  is that the edge  $e$  is directed such that the interior of  $f$  locally lies to the right of  $e$  and  $a$  specifies the angle inside  $f$  from  $e$  to the following edge. The notion of rotations can be extended to these descriptions since we can compute the angle at a vertex  $v$  enclosed by edges  $uv$  and  $vw$  by summing the corresponding angles in the faces given by the  $a$ -values. For such a description to be an ortho-radial representation, two local conditions need to be satisfied:

1. The angle sum of all edges around each vertex given by the  $a$ -fields is  $360^\circ$ .
2. For each face  $f$ , we have

$$\text{rot}(f) = \begin{cases} 4, & f \text{ is a regular face} \\ 0, & f \text{ is the outer or the central face but not both} \\ -4, & f \text{ is both the outer and the central face.} \end{cases}$$

These conditions ensure that angles are assigned correctly around vertices and inside faces, which implies that all properties of rotations mentioned above hold. An ortho-radial representation  $\Gamma$  of a graph  $G$  is *drawable* if there is a drawing  $\Delta$  of  $G$  embedded as specified by  $\Gamma$  such that the corresponding angles in  $\Delta$  and  $\Gamma$  are equal and the reference edge  $e^*$  points to the right. Unlike for orthogonal representations the two conditions do not guarantee that the ortho-radial representation is drawable. Therefore, Barth et al. [2] introduced a third condition, which is formulated in terms of labelings of essential cycles.

For a simple, essential cycle  $C$  in  $G$  and a path  $P$  from the target vertex  $s$  of the reference edge  $e^*$  to a vertex  $v$  on  $C$  the *labeling*  $\ell_C^P$  assigns to each edge  $e$  on  $C$  the label  $\ell_C^P(e) = \text{rot}(e^* + P + C[v, e])$ . In this paper we always assume that  $P$  is *elementary*, i.e.,  $P$  intersects  $C$  only at its endpoints. For these paths the labeling is independent of the actual choice of  $P$ , which was shown by Barth et al. [2]. We therefore drop the superscript  $P$  and write  $\ell_C(e)$  for the labeling of an edge  $e$  on an essential cycle  $C$ . We call an essential cycle *monotone* if either all its labels are non-negative or all its labels are non-positive. A monotone cycle is a *decreasing* cycle if it has at least one strictly positive label, and it is an *increasing* cycle if  $C$  has at least one strictly negative label. An ortho-radial representation is *valid* if it contains neither decreasing nor increasing cycles. The validity of an ortho-radial representation ensures that on each essential cycle with at least one non-zero label there is at least one edge pointing up and one pointing down. The main theorem of Barth et al. [2] can be stated as follows.<sup>2</sup>

► **Proposition 1** (Reformulation of Theorem 5 in [3]). *An ortho-radial representation is drawable if and only if it is valid.*

<sup>2</sup> In the following we refer to the full version [3] of [2], when citing lemmas and theorems.

To that end, Barth et al. [3] prove the following results among others. Since we use them throughout this paper, we restate them for the convenience of the reader. Both assume ortho-radial representations that are not necessarily valid.

► **Proposition 2** (Lemma 12 in [3]). *Let  $C_1$  and  $C_2$  be two essential cycles and let  $H = C_1 + C_2$  be the subgraph of  $G$  formed by these two cycles. For any common edge  $vw$  of  $C_1$  and  $C_2$  where  $v$  lies on the central face of  $H$ , the labels of  $vw$  are equal, i.e.,  $\ell_{C_1}(vw) = \ell_{C_2}(vw)$ .*

► **Proposition 3** (Lemma 16 in [3]). *Let  $C$  and  $C'$  be two essential cycles that have at least one common vertex. If all edges on  $C$  are labeled with 0,  $C'$  is neither increasing nor decreasing.*

Proposition 2 is a useful tool for comparing the labels of two interwoven essential cycles. For example, if  $C_1$  is decreasing, we can conclude for all edges of  $C_2$  that also lie on  $C_1$  and that are incident to the central face of  $H$  that they have non-negative labels. Proposition 3 is useful in the scenario where we have an essential cycle  $C$  with non-negative labels, and a decreasing cycle  $C'$  that shares a vertex with  $C$ . We can then conclude that  $C$  is also decreasing. In particular, these two propositions together imply that the central face of the graph  $H$  formed by two decreasing cycles is bounded by a decreasing cycle.

## 2.2 Symmetries and Normalization

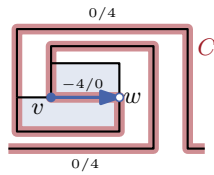
In our arguments we frequently exploit certain symmetries. For an ortho-radial representation  $\Gamma$  we introduce two new ortho-radial representations, its *flip*  $\bar{\Gamma}$  and its *mirror*  $\hat{\Gamma}$ . Geometrically, viewed as a drawing on a cylinder, a flip corresponds to rotating the cylinder by  $180^\circ$  around a line perpendicular to the axis of the cylinder so that it is upside down, whereas mirroring corresponds to mirroring it at a plane that is parallel to the axis of the cylinder. Intuitively, the first transformation exchanges left/right and top/bottom, and thus preserves monotonicity of cycles, while the second transformation exchanges left/right but not top/bottom, and thus maps increasing cycles to decreasing ones and vice versa. This intuition is indeed true with the correct definitions of  $\bar{\Gamma}$  and  $\hat{\Gamma}$ , but due to the non-locality of the validity condition for ortho-radial representations and the dependence on a reference edge this requires some care.

Moreover, in the following sections we restrict ourselves to instances with minimum degree 2, which we call *normalized* instances. To this end we replace each degree-1 vertex  $v$  by a 4-cycle forming a rectangle. Since  $v$  is not part of any simple essential cycle, this does not affect the validity of the representation. Normalized instances do not contain  $360^\circ$  turns at vertices, which simplifies our arguments. See the full version of this paper for details [24].

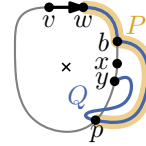
## 3 Finding Monotone Cycles

The two conditions for ortho-radial representations are local and checking them can easily be done in linear time. We therefore assume in this section that we are given a planar 4-graph  $G$  with an ortho-radial representation  $\Gamma$ . The condition for validity however references all essential cycles of which there may be exponentially many. We present an algorithm that checks whether  $\Gamma$  contains a monotone cycle and computes such a cycle if one exists. The main difficulty is that the labels on a decreasing cycle  $C$  depend on an elementary path  $P$  from the reference edge to  $C$ . However, we know neither the path  $P$  nor the cycle  $C$  in advance, and choosing a specific cycle  $C$  may rule out certain paths  $P$  and vice versa.

We only describe how to search for decreasing cycles; increasing cycles can be found by searching for decreasing cycles in the mirrored representation. A decreasing cycle  $C$  is *outermost* if it is not contained in the interior of any other decreasing cycle. Clearly, if  $\Gamma$  contains a decreasing cycle, then it also has an outermost one. We first show that in this case this cycle is uniquely determined.



■ **Figure 3** The search from  $vw$  finds the non-decreasing cycle  $C$ . Edges are labeled  $\ell_C(e)/\tilde{\ell}(e)$ .



■ **Figure 4** Path  $Q$  and its prefix  $P$  that leaves  $C$  once and ends at a vertex  $p$  of  $C$ .

► **Lemma 4.** *If  $\Gamma$  contains a decreasing cycle, there is a unique outermost decreasing cycle.*

**Proof.** Assume that  $\Gamma$  has two outermost decreasing cycles  $C_1$  and  $C_2$ , i.e.,  $C_1$  does not lie in the interior of  $C_2$  and vice versa. Let  $C$  be the cycle bounding the outer face of the subgraph  $H = C_1 + C_2$  that is formed by the two decreasing cycles. By construction,  $C_1$  and  $C_2$  lie in the interior of  $C$ , and we claim that  $C$  is a decreasing cycle contradicting that  $C_1$  and  $C_2$  are outermost. To that end, we show that  $\ell_C(e) = \ell_{C_1}(e)$  for any edge  $e$  that belongs to both  $C$  and  $C_1$ , and  $\ell_C(e) = \ell_{C_2}(e)$  for any edge  $e$  that belongs to both  $C$  and  $C_2$ . Hence, all edges of  $C$  have a non-negative label since  $C_1$  and  $C_2$  are decreasing. By Proposition 3 there is at least one label of  $C$  that is positive, and hence  $C$  is a decreasing cycle.

It remains to show that  $\ell_C(e) = \ell_{C_1}(e)$  for any edge  $e$  that belongs to both  $C$  and  $C_1$ ; the case that  $e$  belongs to both  $C$  and  $C_2$  can be handled analogously. Let  $\Gamma_H$  be the ortho-radial representation  $\Gamma$  restricted to  $H$ . We flip the cylinder to exchange the outer face with the central face and vice versa. More precisely, the reverse edge  $\bar{e}$  of  $e$  lies on the central face of the flipped representation  $\bar{\Gamma}_H$  of  $\Gamma_H$ . Further, it proves that  $\bar{\ell}_{\bar{C}}(\bar{e}) = \ell_C(e)$  and  $\bar{\ell}_{\bar{C}_1}(\bar{e}) = \ell_{C_1}(e)$ , where  $\bar{\ell}$  is the labeling in  $\bar{\Gamma}_H$ . Hence, by Proposition 2 we obtain  $\bar{\ell}_{\bar{C}}(\bar{e}) = \bar{\ell}_{\bar{C}_1}(\bar{e})$ . Flipping back the cylinder, we obtain  $\ell_C(e) = \ell_{C_1}(e)$ . ◀

The core of our algorithm is an adapted left-first DFS. Given a directed edge  $e$  it determines the outermost decreasing cycle  $C$  in  $\Gamma$  such that  $C$  contains  $e$  in the given direction and  $e$  has the smallest label among all edges on  $C$ , if such a cycle exists. By running this test for each directed edge of  $G$  as the start edge, we find a decreasing cycle if one exists.

Our algorithm is based on a DFS that visits each vertex at most once. A left-first search maintains for each visited vertex  $v$  a reference edge  $\text{ref}(v)$ , the edge of the search tree via which  $v$  was visited. Whenever it has a choice which vertex to visit next, it picks the first outgoing edge in clockwise direction after the reference edge that leads to an unvisited vertex. In addition to that, we employ a filter that ignores certain outgoing edges during the search. To that end, we define for all outgoing edges  $e$  incident to a visited vertex  $v$  a *search label*  $\tilde{\ell}(e)$  by setting  $\tilde{\ell}(e) = \tilde{\ell}(\text{ref}(v)) + \text{rot}(\text{ref}(v) + e)$  for each outgoing edge  $e$  of  $v$ . In our search we ignore edges with negative search labels. For a given directed edge  $vw$  in  $G$  we initialize the search by setting  $\text{ref}(w) = vw$ ,  $\tilde{\ell}(vw) = 0$  and then start searching from  $w$ .

Let  $T$  denote the directed search tree with root  $w$  constructed by the DFS in this fashion. If  $T$  contains  $v$ , then this determines a *candidate cycle*  $C$  containing the edge  $vw$ . If  $C$  is a decreasing cycle, which we can easily check by determining an elementary path from the reference edge to  $C$ , we report it. Otherwise, we show that there is no outermost decreasing cycle  $C$  such that  $vw$  lies on  $C$  and has the smallest label among all edges on  $C$ .

It is necessary to check that  $C$  is essential and decreasing. For example the cycle in Figure 3 is found by the search and though it is essential, it is nondecreasing. This is caused by the fact that the label of  $vw$  is actually  $-4$  on this cycle but the search assumes it to be 0.



(a) The edge  $vw$  lies on the outer face of  $H$ . (b) The edge  $vw$  does not lie on the outer face of  $H$ .

■ **Figure 5** The two possible embeddings of the subgraph formed by the decreasing cycle  $C$  and the path  $P$ , which was found by the search.

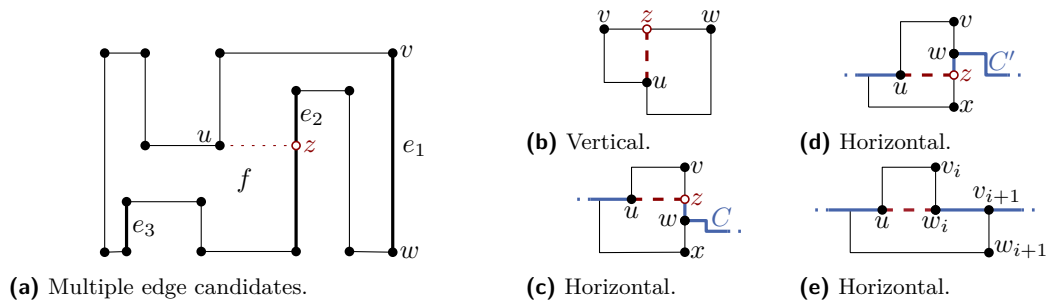
► **Lemma 5.** *Assume  $\Gamma$  contains a decreasing cycle. Let  $C$  be the outermost decreasing cycle of  $\Gamma$  and let  $vw$  be an edge on  $C$  with the minimum label, i.e.,  $\ell_C(vw) \leq \ell_C(e)$  for all edges  $e$  of  $C$ . Then the left-first DFS from  $vw$  finds  $C$ .*

**Proof.** Assume that the search does not find  $C$ . Let  $T$  be the tree formed by the edges visited by the search. Since the search does not find  $C$  by assumption, a part of  $C[w, v]$  does not belong to  $T$ . Let  $xy$  be the first edge on  $C[w, v]$  that is not visited, i.e.,  $C[w, x]$  is a part of  $T$  but  $xy \notin T$ . There are two possible reasons for this. Either  $\tilde{\ell}(xy) < 0$  or  $y$  has already been visited before via another path  $Q$  from  $w$  with  $Q \neq C[w, y]$ . The case  $\tilde{\ell}(xy) < 0$  can be excluded as follows. By the construction of the labels  $\tilde{\ell}$ , for any path  $P$  from  $w$  to a vertex  $z$  in  $T$  and any edge  $e'$  incident to  $z$  we have  $\tilde{\ell}(e') = \text{rot}(vw + P + e')$ . In particular,  $\tilde{\ell}(xy) = \text{rot}(C[vw, xy]) = \ell_C(xy) - \ell_C(vw) \geq 0$  since the rotation can be rewritten as a label difference (see [3, Obs. 7]) and  $vw$  has the smallest label on  $C$ .

Hence,  $T$  contains a path  $Q$  from  $w$  to  $x$  that was found by the search before and  $Q$  does not completely lie on  $C$ . There is a prefix of  $Q$  (possibly of length 0) lying on  $C$  followed by a subpath not on  $C$  until the first vertex  $p$  of  $Q$  that again belongs to  $C$ ; see Figure 4. We set  $P = Q[w, p]$  and denote the vertex where  $P$  leaves  $C$  by  $b$ . By construction the edge  $vw$  lies on  $C[p, b]$ . The subgraph  $H = P + C$  that is formed by the decreasing cycle  $C$  and the path  $P$  consists of the three internally vertex-disjoint paths  $P[b, p]$ ,  $C[b, p]$  and  $\overline{C}[b, p]$  between  $b$  and  $p$ . Since edges that are further left are preferred during the search, the clockwise order of these paths around  $b$  and  $p$  is fixed. In  $H$  there are three faces, bounded by  $C$ ,  $\overline{C}[b, p] + \overline{P}[p, b]$  and  $P[b, p] + \overline{C}[p, b]$ , respectively. Since  $C$  is an essential cycle and a face in  $H$ , it is the central face and one of the two other faces is the outer face. These two possibilities are shown in Figure 5. We denote the cycle bounding the outer face but in which the edges are directed such that the outer face lies locally to the left by  $C'$ . That is, the boundary of the outer face is  $\overline{C}'$ . We distinguish cases based on which of the two possible cycles constitutes  $\overline{C}'$ .

If  $\overline{C}' = \overline{C}[b, p] + \overline{P}[p, b]$  forms the outer face of  $H$ ,  $vw$  lies on  $C'$  as illustrated in Figure 5a and we show that  $C'$  is a decreasing cycle, which contradicts the assumption that  $C$  is the outermost decreasing cycle. Since  $P$  is simple and lies in the exterior of  $C$ , the path  $P$  is contained in  $C'$ , which means  $C'[w, p] = P$ . The other part of  $C'$  is formed by  $C[p, w]$ . Since  $C$  forms the central face of  $H$ , the labels of the edges on  $C[p, w]$  are the same for  $C$  and  $C'$  by Proposition 2. In particular,  $\ell_C(vw) = \ell_{C'}(vw)$  and all the labels of edges on  $C[p, w]$  are non-negative because  $C$  is decreasing. The label of any edge  $e$  on both  $C'$  and  $P$  is  $\ell_{C'}(e) = \ell_{C'}(vw) + \text{rot}(vw + P[w, e]) = \ell_C(vw) + \tilde{\ell}(e) \geq 0$ . Thus, the labeling of  $C'$  is non-negative. Further, not all labels of  $C'$  are 0 since otherwise  $C$  would not be a decreasing cycle by Proposition 3. Hence,  $C'$  is decreasing and contains  $C$  in its interior, a contradiction.

If  $\overline{C}' = \overline{C}[p, b] + P[b, p]$ , the edge  $vw$  does not lie on  $C'$ ; see Figure 5b. We show that  $C'$  is a decreasing cycle containing  $C$  in its interior, again contradicting the choice of  $C$ . As above, Proposition 2 implies that the common edges of  $C$  and  $C'$  have the same labels



■ **Figure 6** Examples of augmentations. (a) The candidate edges of  $u$  are  $e_1$ ,  $e_2$  and  $e_3$ . (b) Insertion of vertical edge  $uz$ . (c)  $\Gamma_{vw}^u$  contains a decreasing cycle. (d)  $\Gamma_{vw}^u$  is valid. (e) Insertion of horizontal edge  $uw_i$  because there is a horizontal path from  $w_i$  to  $u$ .

on both cycles. It remains to show that all edges  $xy$  on  $\bar{P}[p, b]$  have non-negative labels. To establish this we use paths to the edge that follows  $b$  on  $C$ . This edge  $bc$  has the same label on both cycles and thus provides a handle on  $\ell_{C'}(xy)$ . We make use of the following equations, which follow immediately from the definition of the (search) labels.

$$\begin{aligned} \ell_{C'}(bc) &= \ell_{C'}(xy) + \text{rot}(\bar{P}[xy, db]) + \text{rot}(dbc) = \ell_{C'}(xy) - \text{rot}(P[bd, yx]) - \text{rot}(cbd) \\ \ell_C(bc) &= \ell_C(vw) + \text{rot}(C[vw, ab]) + \text{rot}(abc) \\ \tilde{\ell}(yx) &= \text{rot}(C[vw, ab]) + \text{rot}(abd) + \text{rot}(P[bd, yx]) \end{aligned}$$

Since  $\ell_C(bc) = \ell_{C'}(bc)$  and  $\text{rot}(abd) = -\text{rot}(dba)$ , we thus get

$$\begin{aligned} \ell_{C'}(xy) &= \ell_C(vw) + \text{rot}(C[vw, ab]) + \text{rot}(abc) + \text{rot}(P[bd, yx]) + \text{rot}(cbd) \\ &= \ell_C(vw) + \tilde{\ell}(yx) + \text{rot}(dba) + \text{rot}(abc) + \text{rot}(cbd). \end{aligned}$$

Since  $\ell_C(vw) \geq 0$  and  $\tilde{\ell}(yx) \geq 0$  (as  $yx$  was not filtered out), it follows that  $\ell_{C'}(xy) \geq \text{rot}(abc) + \text{rot}(dba) + \text{rot}(cbd) = 2$  as this is the sum of clockwise rotations around a degree-3 vertex. Hence,  $C'$  is decreasing and contains  $C$  in its interior, a contradiction. Since both embeddings of  $H$  lead to a contradiction, we obtain a contradiction to our initial assumption that the search fails to find  $C$ . ◀

The left-first DFS clearly runs in  $O(n)$  time. In order to guarantee that the search finds a decreasing cycle if one exists, we run it for each of the  $O(n)$  directed edges of  $G$ . Since some edge must have the lowest label on the outermost decreasing cycle, Lemma 5 guarantees that we eventually find a decreasing cycle if one exists. Increasing cycles can be found by finding decreasing cycles in the mirror representation  $\hat{\Gamma}$ .

► **Theorem 6.** *Let  $G$  be a planar 4-graph on  $n$  vertices and let  $\Gamma$  be an ortho-radial representation of  $G$ . It can be determined in  $O(n^2)$  time whether  $\Gamma$  is valid.*

## 4 Rectangulation

The core of the algorithm for drawing a valid ortho-radial representation  $\Gamma$  of a graph  $G$  by Barth et al. [2] is a *rectangulation procedure* that successively augments  $G$  with new vertices and edges to a graph  $G^*$  along with a valid ortho-radial representation  $\Gamma^*$  where every face of  $G^*$  is a *rectangle*. A regular face is a rectangle if it has exactly four turns, which are all right turns. The outer and central faces are rectangles if they have no turns. The ortho-radial representation  $\Gamma^*$  is then drawn by computing flows in two flow networks [3, Thm. 18].



To facilitate the analysis, we briefly sketch the augmentation procedure. Here it is crucial that we assume our instances to be normalized; in particular they do not have degree-1 vertices. The augmentation algorithm works by augmenting non-rectangular faces one by one, thereby successively removing concave angles at the vertices until all faces are rectangles. Consider a face  $f$  with a left turn (i.e., a concave angle) at  $u$  such that the following two turns when walking along  $f$  (in clockwise direction) are right turns; see Figure 6. We call  $u$  a *port* of  $f$ . We define a set of *candidate edges* that contains precisely those edges  $vw$  of  $f$ , for which  $\text{rot}(f[u, vw]) = 2$ ; see Figure 6a. We treat this set as a sequence, where the edges appear in the same order as in  $f$ , beginning with the first candidate after  $u$ . The *augmentation*  $\Gamma_{vw}^u$  with respect to a candidate edge  $vw$  is obtained by splitting the edge  $vw$  into the edges  $vz$  and  $zw$ , where  $z$  is a new vertex, and adding the edge  $uz$  in the interior of  $f$  such that the angle formed by  $zu$  and the edge following  $u$  on  $f$  is  $90^\circ$ . The direction of the new edge  $uz$  in  $\Gamma_{vw}^u$  is the same for all candidate edges. If this direction is vertical, we call  $u$  a *vertical port* and otherwise a *horizontal port*. We note that any vertex with a concave angle in a face becomes a port during the augmentation process. In particular, the incoming edge of the vertex determines whether the port is horizontal or vertical. The condition for candidates guarantees that  $\Gamma_{vw}^u$  is an ortho-radial representation. It may, however, not be valid. The crucial steps in [2] are establishing the following facts.

- Fact 1)** Let  $u$  be a vertical port. Augmenting with the first candidate never produces a monotone cycle [3, Lemma 21].
- Fact 2)** Let  $u$  be a horizontal port. Augmenting with the first candidate never produces an increasing cycle [3, Lemma 22] and augmenting with the last candidate never produces a decreasing cycle [3, Lemma 24].
- Fact 3)** Let  $u$  be a horizontal port. If two consecutive candidates  $e_i = v_i w_i$  and  $e_{i+1} = v_{i+1} w_{i+1}$  produce a decreasing and an increasing cycle, respectively, then  $w_i, v_{i+1}$  and  $u$  lie on a path that starts at  $w_i$  or  $v_{i+1}$ , and whose edges all point right; see Figure 6e. A suitable augmentation can be found in  $O(n)$  time. [3, Lemmas 25, 26].

It thus suffices to test for each candidate whether  $\Gamma_{vw}^u$  is valid until either such a *valid augmentation* is found or we find two consecutive candidate edges where the first produces a decreasing cycle and the second produces an increasing cycle. Then, Fact 3 yields the desired valid augmentation. Since each valid augmentation reduces the number of concave angles, we obtain a rectangulation after  $O(n)$  valid augmentations. Moreover, there are  $O(n)$  candidates for each augmentation, each of which can be tested for validity (and increasing/decreasing cycles can be detected) in  $O(n^2)$  time by Theorem 6. Thus, the augmentation algorithm can be implemented to run in  $O(n^4)$  time.

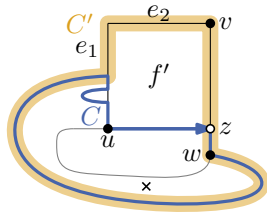
In the remainder of this section we outline an improvement to  $O(n^2)$  time, which is achieved in two steps. First, we show that due to the nature of augmentations the validity test can be done in  $O(n)$  time. Second, for each augmentation we execute a post-processing that reduces the number of validity tests to  $O(n)$  in total. The detailed constructions and omitted proofs of both steps are deferred to [24].

#### 4.1 1st Improvement – Faster Validity Test

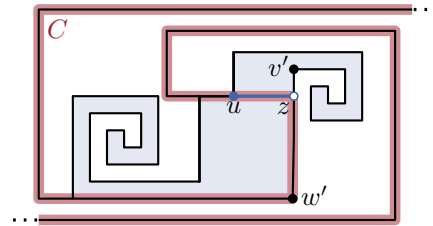
Recall from above that, once we picked a port  $u$  in a face  $f$ , there is an ordered set of candidate edges  $e_1, \dots, e_k$ . We know from [2] that one of the augmentations  $\Gamma_{e_i}^u$  leads to a valid augmentation. We improve on naively testing the validity of  $\Gamma_{e_i}^u$  in quadratic time.

If  $u$  is a vertical port (Figure 6b), Fact 1 guarantees that  $\Gamma_{e_1}^u$  is valid, so no validity test is required. If  $u$  is a horizontal port (Figure 6c–e), we assume w.l.o.g. that the inserted edge points to the right; otherwise we consider the flipped representation  $\bar{\Gamma}$ .





■ **Figure 7** A decreasing cycle  $C$  that uses  $uz$  and an essential cycle  $C'$  derived from  $C$ .



■ **Figure 8** Here, the insertion of the edge  $uz$  to the last candidate  $v'w'$  introduces an increasing cycle  $C$  with  $l_C(uz) = -4$ .

We show that in this case we can strongly restrict the direction of a monotone cycle as well as its label for the new edge  $uz$ . We start with the detection of decreasing cycles and show that for the augmentation with the first candidate edge  $e_1 = vw$  a decreasing cycle  $C$  in  $\Gamma_{vw}^u$  may only use the new edge  $uz$  in the direction from  $u$  to  $z$  and in this case its label is 0.

► **Lemma 7.** *Let  $vw$  be the first candidate on  $f$  after  $u$ . If  $\Gamma_{vw}^u$  contains a decreasing cycle  $C$ , then  $C$  contains  $uz$  in this direction and  $l_C(uz) = 0$ .*

**Proof.** We first consider the case that  $C$  uses  $uz$  (and not  $zu$ ) and assume that  $l_C(uz) \neq 0$ ; see Figure 7. Since  $uz$  points right,  $l_C(uz)$  is divisible by 4. Together with  $l_C(uz) \geq 0$  because  $C$  is decreasing, we obtain  $l_C(uz) \geq 4$ . By Lemma 14 of [3] there is an essential cycle  $C'$  without  $uz$  in the subgraph  $H$  that is formed by the new rectangular face  $f'$  and  $C$ . The labels of any common edge  $e$  of  $C$  and  $C'$  are equal and  $l_{C'}(e) = l_C(e) \geq 0$ . All other edges of  $C'$  lie on  $f'$ . Since  $f'$  is rectangular, the labels of these edges differ by at most 1 from  $l_C(uz)$ . By assumption it is  $l_C(uz) \geq 4$  and therefore  $l_{C'}(e) \geq 3$  for all edges  $e \in C' \cap f'$ . Hence,  $C'$  is a decreasing cycle in  $G$  contradicting the validity of  $\Gamma$ . If  $zu \in C$ , it is  $l_C(zu) \geq 2$  and a similar argument yields a decreasing cycle in  $\Gamma$ . ◀

While the same statement does not generally hold for all candidates, it does hold if the first candidate creates a decreasing cycle.

► **Lemma 8.** *Let  $vw$  be the first candidate and  $v'w'$  be another candidate. Denote the edge inserted in  $\Gamma_{v'w'}^u$  by  $uz'$ . If  $\Gamma_{vw}^u$  contains a decreasing cycle, any decreasing cycle  $C'$  in  $\Gamma_{v'w'}^u$  uses  $uz'$  in this direction and  $l_{C'}(uz') = 0$ .*

Altogether, we can efficiently test which of the candidates produce decreasing cycles as follows. By Lemma 7, if the first candidate is not valid, then  $\Gamma_{e_1}^u$  has a decreasing cycle that contains the new edge  $uz$  with label 0, which is hence the minimum label for all edges on the cycle. This can be tested in  $O(n)$  time by Lemma 5. Fact 2 guarantees that we either find a valid augmentation or a decreasing cycle. In the former case we are done, in the second case Lemma 8 allows us to similarly restrict the labels of  $uz$  to 0 for the remaining candidate edges, thus allowing us to detect decreasing cycles in  $\Gamma_{e_i}^u$  in  $O(n)$  time for  $i = 2, \dots, k$ .

It is tempting to use the mirror symmetry to exchange increasing and decreasing cycles to deal with increasing cycles in an analogous fashion. However, this fails as mirroring invalidates the property that  $u$  is followed by two right turns in clockwise direction. For example, in Figure 8 inserting the edge to the last candidate introduces an increasing cycle  $C$  with  $l_C(uz) = -4$ . We therefore give a direct algorithm for detecting increasing cycles in this case.

Let  $e_i = v_iw_i$  and  $e_{i+1} = v_{i+1}w_{i+1}$  be two consecutive candidates for  $u$  such that  $\Gamma_{e_i}^u$  contains a decreasing cycle but  $\Gamma_{e_{i+1}}^u$  does not. If  $\Gamma_{e_{i+1}}^u$  contains an increasing cycle, then by Fact 3 the vertices  $w_i, v_{i+1}$  and  $u$  lie on a path that starts at  $w_i$  or  $v_{i+1}$ , and whose edges

all point right. The presence of such a horizontal path  $P$  can clearly be checked in linear time, thus allowing us to also detect increasing cycles provided that the previous candidate produced a decreasing cycle. If  $P$  exists, we insert the edge  $uw_i$  or  $uv_{i+1}$  depending on whether  $P$  starts at  $w_i$  or  $v_{i+1}$ , respectively; see Figure 6e for the first case. By Proposition 3 this does not produce monotone cycles. Otherwise, if  $P$  does not exist, the augmentation  $\Gamma_{e_{i+1}}^u$  is valid. In both cases we have resolved the horizontal port  $u$  successfully.

Summarizing, the overall algorithm for augmenting from a horizontal port  $u$  now works as follows. By exploiting Lemmas 7 and 8, we test the candidates in the order as they appear on  $f$  until we find the first candidate  $e$  for which  $\Gamma_e^u$  does not contain a decreasing cycle. Using Fact 3 we either find that  $\Gamma_e^u$  is valid, or we find a horizontal path as described above. In both cases this allows us to determine an edge whose insertion does not introduce a monotone cycle. Since in each test for a decreasing cycle the edge  $uz$  can be restricted to have label 0, each of the tests takes linear time. This improves the running time of the rectangulation algorithm to  $O(n^3)$ .

Instead of linearly searching for a suitable candidate for  $u$  we can employ a binary search on the candidates, which reduces the overall running time to  $O(n^2 \log n)$ .

## 4.2 2nd Improvement – 2-Phase Augmentation Step

We add a second phase to our augmentation step that post-processes the resulting augmentation after each step to reduce the total number of validity tests to  $O(n)$ .

More precisely, the first phase of the augmentation step inserts a new edge  $uz$  in a given ortho-radial representation  $\Gamma$  for a port  $u$  as before; we denote the resulting valid ortho-radial representation by  $\Gamma'$ . Afterwards, if  $u$  is a horizontal port, we apply the second phase on  $u$ . Let  $e_1, \dots, e_k$  be the candidates of  $u$ , where  $e_k$  is the candidate for the first validity test that does not fail in the first phase. We call  $e_1, e_{k-1}$  and  $e_k$  *boundary candidates* and the others *intermediate candidates*. The second phase augments  $\Gamma'$  such that afterwards each intermediate candidate belongs to a rectangle in the resulting ortho-radial representation  $\Gamma''$ . Further,  $\Gamma''$  has fewer vertices with concave angles becoming horizontal ports and at most two more vertices with concave angles becoming vertical ports during the remaining augmentation process. As the second phase is skipped for vertical ports,  $O(n)$  augmentation steps are executed overall. Moreover, each edge can be an intermediate candidate for at most one vertex, which yields that there are  $O(n)$  intermediate candidates over all augmentation steps. Finally, for each port there are at most three boundary candidates, which yields  $O(n)$  boundary candidates over all augmentation steps. Assigning the validity tests to their candidates, the algorithm executes  $O(n)$  validity tests overall. Altogether, this yields  $O(n^2)$  running time in total.

► **Theorem 9.** *Given a valid ortho-radial representation  $\Gamma$  of a graph  $G$ , a corresponding rectangulation can be computed in  $O(n^2)$  time.*

In particular, using Corollary 19 from [3], given a graph  $G$  with valid ortho-radial representation  $\Gamma$ , a corresponding ortho-radial drawing  $\Delta$  can be computed in  $O(n^2)$  time.

## 5 Conclusion

In this paper, we have described an algorithm that checks the validity of an ortho-radial representation in  $O(n^2)$  time. In the positive case, we can also produce a corresponding drawing in the same running time, whereas in the negative case we find a monotone cycle. This answers an open question of Barth et al. [2] and allows for a purely combinatorial

treatment of the bend minimization problem for ortho-radial drawings. It is an interesting open question whether the running time can be improved to near-linear. However, our main open question is how to find valid ortho-radial representations with few bends.

---

## References

- 1 Md. Jawaherul Alam, Stephen G. Kobourov, and Debajyoti Mondal. Orthogonal layout with optimal face complexity. *Computational Geometry*, 63:40–52, 2017.
- 2 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings. In Boris Aronov and Matthew J. Katz, editors, *Computational Geometry (SoCG'17)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- 3 Lukas Barth, Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Towards a Topology-Shape-Metrics Framework for Ortho-Radial Drawings. *CoRR*, arXiv:1703.06040, 2017. arXiv:1703.06040.
- 4 Carlo Batini, Enrico Nardelli, and Roberto Tamassia. A layout algorithm for data flow diagrams. *IEEE Transactions on Software Engineering*, SE-12(4):538–546, 1986.
- 5 P. Bertolazzi, G. Di Battista, and W. Didimo. Computing orthogonal drawings with the minimum number of bends. *IEEE Transactions on Computers*, 49(8):826–840, 2000.
- 6 Sandeep N. Bhatt and Frank Thomson Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer and System Sciences*, 28(2):300–343, 1984.
- 7 Therese Biedl. New lower bounds for orthogonal graph drawings. In Franz J. Brandenburg, editor, *Graph Drawing (GD'96)*, Lecture Notes of Computer Science, pages 28–39. Springer Berlin Heidelberg, 1996.
- 8 Therese Biedl and Goos Kant. A better heuristic for orthogonal graph drawings. *Computational Geometry*, 9(3):159–180, 1998.
- 9 Therese C. Biedl, Brendan P. Madden, and Ioannis G. Tollis. The three-phase method: A unified approach to orthogonal graph drawing. In Giuseppe DiBattista, editor, *Graph Drawing (GD'97)*, Lecture Notes in Computer Science, pages 391–402. Springer Berlin Heidelberg, 1997.
- 10 Thomas Bläsius, Ignaz Rutter, and Dorothea Wagner. Optimal orthogonal graph drawing with convex bend costs. *ACM Transactions on Algorithms*, 12(3):33, 2016.
- 11 Thomas Bläsius, Sebastian Lehmann, and Ignaz Rutter. Orthogonal graph drawing with inflexible edges. *Computational Geometry*, 55:26–40, 2016.
- 12 Yi-Jun Chang and Hsu-Chun Yen. On Bend-Minimized Orthogonal Drawings of Planar 3-Graphs. In Boris Aronov and Matthew J. Katz, editors, *Computational Geometry (SoCG'17)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- 13 Sabine Cornelsen and Andreas Karrenbauer. Accelerated Bend Minimization. In Marc van Kreveld and Bettina Speckmann, editors, *Graph Drawing (GD'12)*, Lecture Notes of Computer Science, pages 111–122. Springer Berlin Heidelberg, 2012.
- 14 Markus Eiglsperger, Carsten Gutwenger, Michael Kaufmann, Joachim Kupke, Michael Jünger, Sebastian Leipert, Karsten Klein, Petra Mutzel, and Martin Siebenhaller. Automatic Layout of UML Class Diagrams in Orthogonal Style. *Information Visualization*, 3(3):189–208, 2004.
- 15 Markus Eiglsperger, Michael Kaufmann, and Martin Siebenhaller. A Topology-shape-metrics Approach for the Automatic Layout of UML Class Diagrams. In *Software Visualization (SoftVis'03)*, pages 189–ff. ACM, 2003.
- 16 Stefan Felsner, Michael Kaufmann, and Pavel Valtr. Bend-optimal orthogonal graph drawing in the general position model. *Computational Geometry*, 47(3, Part B):460–468, 2014. Special Issue on the 28th European Workshop on Computational Geometry (EuroCG 2012).
- 17 Martin Fink, Herman Haverkort, Martin Nöllenburg, Maxwell Roberts, Julian Schuhmann, and Alexander Wolff. Drawing Metro Maps Using Bézier Curves. In W Didimo and M Patrignani, editors, *Graph Drawing (GD'13)*, Lecture Notes in Computer Science, pages 463–474. Springer International Publishing, 2013.

- 18 Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In Franz J. Brandenburg, editor, *Graph Drawing (GD'96)*, Lecture Notes in Computer Science, pages 254–266. Springer Berlin Heidelberg, 1996.
- 19 Carsten Gutwenger, Michael Jünger, Karsten Klein, Joachim Kupke, Sebastian Leipert, and Petra Mutzel. A New Approach for Visualizing UML Class Diagrams. In *Symposium on Software Visualization (SoftVis'03)*, pages 179–188, New York, NY, USA, 2003. ACM.
- 20 Madieh Hasheminezhad, S. Mehdi Hashemi, Brendan D. McKay, and Maryam Tahmasbi. Rectangular-Radial Drawings of Cubic Plane Graphs. *Computational Geometry: Theory and Applications*, 43:767–780, 2010.
- 21 Madieh Hasheminezhad, S. Mehdi Hashemi, and Maryam Tahmasbi. Ortho-radial drawings of graphs. *Australasian Journal of Combinatorics*, 44:171–182, 2009.
- 22 Seok-Hee Hong, Damian Merrick, and Hugo A. D. do Nascimento. Automatic Visualisation of Metro Maps. *Journal of Visual Languages and Computing*, 17(3):203–224, 2006.
- 23 S. Kieffer, T. Dwyer, K. Marriott, and M. Wybrow. HOLA: Human-like Orthogonal Network Layout. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):349–358, 2016.
- 24 Benjamin Niedermann, Ignaz Rutter, and Matthias Wolf. Efficient Algorithms for Ortho-Radial Graph Drawing. *CoRR*, arXiv:1903.05048, 2019. [arXiv:1903.05048](https://arxiv.org/abs/1903.05048).
- 25 Martin Nöllenburg and Alexander Wolff. Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *Transactions on Visualization and Computer Graphics*, 17(5):626–641, 2011.
- 26 Achilleas Papakostas and Ioannis G. Tollis. Algorithms for area-efficient orthogonal drawings. *Computational Geometry*, 9(1):83–110, 1998.
- 27 Ulf Rüegg, Steve Kieffer, Tim Dwyer, Kim Marriott, and Michael Wybrow. Stress-Minimizing Orthogonal Layout of Data Flow Diagrams with Ports. In Christian Duncan and Antonios Symvonis, editors, *Graph Drawing (GD'14)*, Lecture Notes in Computer Science, pages 319–330. Springer Berlin Heidelberg, 2014.
- 28 R. Tamassia. On Embedding a Graph in the Grid with the Minimum Number of Bends. *Journal on Computing*, 16(3):421–444, 1987.
- 29 Roberto Tamassia, Giuseppe Di Battista, and Carlo Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61–79, 1988.
- 30 Roberto Tamassia, Ioannis G. Tollis, and Jeffrey Scott Vitter. Lower bounds for planar orthogonal drawings of graphs. *Information Processing Letters*, 39(1):35–40, 1991.
- 31 L. G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, 30(02):135–140, 1981.
- 32 Yu-Shuen Wang and Ming-Te Chi. Focus+Context Metro Maps. *Transactions on Visualization and Computer Graphics*, 17(12):2528–2535, 2011.
- 33 Michael Wybrow, Kim Marriott, and Peter J. Stuckey. Orthogonal Connector Routing. In David Eppstein and Emden R. Gansner, editors, *Graph Drawing (GD'10)*, Lecture Notes in Computer Science, pages 219–231. Springer Berlin Heidelberg, 2010.

# On the Chromatic Number of Disjointness Graphs of Curves

János Pach

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland  
Rényi Institute, Budapest, Hungary  
janos.pach@epfl.ch

István Tomon

École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland  
istvan.tomon@epfl.ch

---

## Abstract

Let  $\omega(G)$  and  $\chi(G)$  denote the clique number and chromatic number of a graph  $G$ , respectively. The *disjointness graph* of a family of curves (continuous arcs in the plane) is the graph whose vertices correspond to the curves and in which two vertices are joined by an edge if and only if the corresponding curves are disjoint. A curve is called *x-monotone* if every vertical line intersects it in at most one point. An *x-monotone* curve is *grounded* if its left endpoint lies on the *y*-axis.

We prove that if  $G$  is the disjointness graph of a family of grounded *x-monotone* curves such that  $\omega(G) = k$ , then  $\chi(G) \leq \binom{k+1}{2}$ . If we only require that every curve is *x-monotone* and intersects the *y*-axis, then we have  $\chi(G) \leq \frac{k+1}{2} \binom{k+2}{3}$ . Both of these bounds are best possible. The construction showing the tightness of the last result settles a 25 years old problem: it yields that there exist  $K_k$ -free disjointness graphs of *x-monotone* curves such that any proper coloring of them uses at least  $\Omega(k^4)$  colors. This matches the upper bound up to a constant factor.

**2012 ACM Subject Classification** Mathematics of computing → Graph theory

**Keywords and phrases** string graph, chromatic number, intersection graph

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.54

**Related Version** <https://arxiv.org/abs/1811.09158>

**Funding** *János Pach*: Research partially supported by Swiss National Science Foundation grants no. 200020-162884 and 200021-175977.

*István Tomon*: Research partially supported by Swiss National Science Foundation grants no. 200020-162884 and 200021-175977.

**Acknowledgements** We would like to thank Andrew Suk, Gábor Tardos, Géza Tóth and Bartosz Walczak for fruitful discussions.

## 1 Introduction

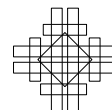
Given a family of sets,  $\mathcal{C}$ , the *intersection graph* of  $\mathcal{C}$  is the graph, whose vertices correspond to the elements of  $\mathcal{C}$ , and two vertices are joined by an edge if the corresponding sets have a nonempty intersection. Also, the *disjointness graph* of  $\mathcal{C}$  is the complement of the intersection graph of  $\mathcal{C}$ , that is, two vertices are joined by an edge if the corresponding sets are disjoint. As usual, we denote the *clique number*, the *independence number*, and the *chromatic number* of a graph  $G$  by  $\omega(G)$ ,  $\alpha(G)$  and  $\chi(G)$ , respectively.



© János Pach and István Tomon;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 54; pp. 54:1–54:17



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



## 1.1 Clique number vs. chromatic number

Computing these parameters for intersection graphs of various classes of geometric objects (segments, boxes, disks etc.) or for other geometrically defined graphs (such as visibility graphs) is a computationally hard problem and a classic topic in computational and combinatorial geometry [1, 5, 10, 17, 23, 24]. There are many interesting results connecting the clique number and the chromatic number of geometric intersection graphs, starting with a beautiful theorem of Asplund and Grünbaum [2], which states that every intersection graph  $G$  of axis-parallel rectangles in the plane satisfies  $\chi(G) \leq 4(\omega(G))^2$ .

A family  $\mathcal{G}$  of graphs is  $\chi$ -bounded if there exists a function  $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  such that every  $G \in \mathcal{G}$  satisfies  $\chi(G) \leq f(\omega(G))$ . In this case, say that the function  $f$  is  $\chi$ -bounding for  $\mathcal{G}$ . Using this terminology, the result of Asplund and Grünbaum [2] mentioned above can be rephrased as follows: The family of intersection graphs of axis-parallel rectangles in the plane is  $\chi$ -bounded with bounding function  $f(k) = 4k^2$ . (It is conjectured that the same is true with bounding function  $f(k) = O(k)$ .) However, an ingenious construction of Burling [4] shows that the family of intersection graphs of axis-parallel boxes in  $\mathbb{R}^3$  is *not*  $\chi$ -bounded. The  $\chi$ -boundedness of intersection graphs of chords of a circle was established by Gyárfás [14, 15]; see also Kostochka *et al.* [22, 19, 20].

Computing the chromatic number of the *disjointness graph* of a family of objects,  $\mathcal{C}$ , is equivalent to determining the *clique cover number* of the corresponding intersection graph  $G$ , that is, the minimum number of cliques whose vertices together cover the vertex set of  $G$ . This problem can be solved in polynomial time only for some very special families (for instance, if  $\mathcal{C}$  consists of intervals along a line or arcs along a circle [13]). On the other hand, the problem is known to be NP-complete if  $\mathcal{C}$  is a family of chords of a circle [16, 12] or a family of unit disks in the plane [38, 6], and in many other cases. There is a vast literature providing approximation algorithms or inapproximability results for the clique cover number [8, 9].

## 1.2 Families of curves

A *curve* or *string* in  $\mathbb{R}^2$  is the image of a continuous function  $\phi : [0, 1] \rightarrow \mathbb{R}^d$ . A curve  $C \subset \mathbb{R}^2$  is called *x-monotone* if every vertical line intersects  $C$  in at most one point. Note that any convex set can be approximated arbitrarily closely by *x-monotone* curves, so the notion of *x-monotone* curve extends the notion of convex sets. We say that  $C$  is *grounded at the curve*  $L$  if one of the endpoints of  $C$  is in  $L$ , and this is the only intersection point of  $C$  and  $L$ . A *grounded x-monotone curve* is an *x-monotone* curve that is contained in the half-plane  $\{x \geq 0\}$ , and whose left endpoint lies on the vertical line  $\{x = 0\}$ .

It was first suggested by Erdős in the 1970s, and remained the prevailing conjecture for 40 years, that the family of intersection graphs of curves (the family of so-called “string graphs”) is  $\chi$ -bounded [3, 21]. There were many promising facts pointing in this direction. Extending earlier results of McGuinness [28], Suk [37], and Lasoń *et al.* [27], Rok and Walczak [35, 36] proved the conjecture for *grounded* families of curves. Nevertheless, in 2014, Pawlik *et al.* [34] disproved Erdős’s conjecture. They managed to modify Burling’s above-mentioned construction to obtain a sequence of finite families of *segments* in the plane whose intersection graphs,  $G_n$ , are triangle-free (that is,  $\omega(G_n) = 2$ ), but their chromatic numbers tend to infinity, as  $n \rightarrow \infty$ .

Recently, Pach, Tardos and Tóth [32] proved that the family of *disjointness graphs* of curves in the plane is not  $\chi$ -bounded either; see also [30]. However, the situation is different if we restrict our attention to *x-monotone* curves. It was shown in [33, 26] that the family

of disjointness graphs of  $x$ -monotone curves in the plane is  $\chi$ -bounded with a bounding function  $f(k) = k^4$ . For *grounded  $x$ -monotone* curves, the same proof provides a better bounding function:  $f(k) = k^2$ . These results proved 25 years ago were not likely to be tight. However, in spite of many efforts, no-one has managed to improve them or to show that they are optimal.

### 1.3 Our results

The aim of the present paper is to fill this gap. We proved, much to our surprise, that the order of magnitude of the last two bounds cannot be improved. In fact, in the case of grounded  $x$ -monotone curves, we determined the exact value of the best bounding function for every  $k \geq 2$ . To the best of our knowledge, this is the first large family of non-perfect geometric disjointness graphs, for which one can precisely determine the best bounding function.

► **Theorem 1.** *Let  $G$  be the disjointness graph of a family of grounded  $x$ -monotone curves. If  $\omega(G) = k$ , then  $\chi(G) \leq \binom{k+1}{2}$ .*

► **Theorem 2.** *For every positive integer  $k \geq 2$ , there exists a family  $\mathcal{C}$  of grounded  $x$ -monotone curves such that if  $G$  is the disjointness graph of  $\mathcal{C}$ , then  $\omega(G) = k$  and  $\chi(G) = \binom{k+1}{2}$ .*

It turns out that disjointness graphs of grounded  $x$ -monotone curves can be characterized by two total orders defined on their vertex sets that satisfy some special properties. This observation is the key idea behind the proof of the above two theorems.

The disjointness graph of any collection of  $x$ -monotone curves, each of which intersects a given vertical line (the  $y$ -axis, say), is the intersection of two disjointness graphs of grounded  $x$ -monotone curves. The methods used for proving Theorems 1 and 2 can be extended to such disjointness graphs and yield sharp bounds.

► **Theorem 3.** *Let  $G$  be the disjointness graph of a family  $\mathcal{C}$  of  $x$ -monotone curves such that all elements of  $\mathcal{C}$  have nonempty intersection with a vertical line  $l$ . If  $\omega(G) = k$ , then  $\chi(G) \leq \frac{k+1}{2} \binom{k+2}{3}$ .*

► **Theorem 4.** *For every positive integer  $k \geq 2$ , there exists a family  $\mathcal{C}$  of  $x$ -monotone curves such that all elements of  $\mathcal{C}$  have nonempty intersection with a vertical line  $l$ , the disjointness graph  $G$  of  $\mathcal{C}$  satisfies  $\omega(G) = k$ , and  $\chi(G) = \frac{k+1}{2} \binom{k+2}{3}$ .*

As we have mentioned before, according to [33, 26],  $k^4$  is a bounding function for disjointness graphs of *any* family of  $x$ -monotone curves. Theorem 4 implies that the order of magnitude of this bounding function is best possible. Actually, we can obtain a little more.

► **Theorem 5.** *For any positive integer  $k$ , let  $f(k)$  denote the smallest  $m$  such that any  $K_{k+1}$ -free disjointness graph of  $x$ -monotone curves can be properly colored with  $m$  colors. Then we have*

$$\frac{k+1}{2} \binom{k+2}{3} \leq f(k) \leq k^2 \binom{k+1}{2}.$$

Here the lower and upper bounds differ by a factor of less than 6, and there is some hope that one can determine the exact value of  $f(k)$ . The lower bound follows directly from Theorem 4.

Our paper is organized as follows. In Section 2, we prove Theorem 1 and the upper bound in Theorem 5. The existence of the graphs satisfying Theorem 2 is proved in Section 3, using probabilistic techniques. The proofs of Theorems 3 and 4 are presented in Sections 4 and 5, respectively. The last section contains open problems and concluding remarks.



## 2 A bounding function for grounded curves – Proofs of Theorems 1 and 5

First, we establish Theorem 1. As usual, we denote the set  $\{1, 2, \dots, n\}$  by  $[n]$ .

An *ordered graph*  $G_{<}$  is a graph, whose vertex set is endowed with a total ordering  $<$ . Ordered graphs are often more suitable for modelling geometric configurations than unordered ones; see, e.g., [11, 31]. To model families of grounded  $x$ -monotone curves, we introduce a class of ordered graphs.

► **Definition 6.** An ordered graph  $G_{<}$  is called a *semi-comparability graph*, if it has no 4 vertices  $a, b, c, d \in V(G_{<})$  such that  $a < b < c < d$  and  $ab, bc, cd \in E(G_{<})$ , but  $ac, bd \notin E(G_{<})$ .

An unordered graph  $G$  is said to be a *semi-comparability graph*, if its vertex set has a total ordering  $<$  such that  $G_{<}$  is a semi-comparability graph.

Obviously, every *comparability graph* (that is, every graph whose edge set consists of all comparable pairs of a partially ordered set) is a semi-comparability graph.

► **Lemma 7.** The disjointness graph of every family  $\mathcal{C}$  of grounded  $x$ -monotone curves is a semi-comparability graph.

**Proof.** Let  $G$  be the disjointness graph of  $\mathcal{C}$ . Identify the vertices of  $G$  with the elements of  $\mathcal{C}$ . For any  $\gamma \in \mathcal{C}$ , let  $(0, y_\gamma)$  be the left endpoint of  $\gamma$ . Slightly perturbing the curves if necessary, we can assume without loss of generality that no two  $y_\gamma$ s coincide. Let  $<$  be the total ordering on  $V(G)$ , according to which  $\gamma < \gamma'$  if and only if  $y_\gamma < y_{\gamma'}$ .

Suppose for contradiction that there exist 4 curves  $a, b, c, d$  such that  $a < b < c < d$  and  $ab, bc, cd \in E(G)$ , but  $ac, bd \notin E(G)$ . Then  $a$  and  $c$  must intersect, which means that  $a$ ,  $c$ , and the ground line  $x = 0$  enclose a region  $A$ . Since  $b$  does not intersect either of  $a$  or  $c$ , it must lie in  $A$ . In order to intersect  $b$ ,  $d$  has to cross  $c$ , which is a contradiction. ◀

By the dual of Dilworth's theorem [7], also known as Mirsky's theorem [29], comparability graphs are perfect. Thus, any comparability graph  $G$  can be properly colored with  $\omega(G)$  colors. While not all semi-comparability graphs are perfect, they are  $\chi$ -bounded.

► **Lemma 8.** For any semi-comparability graph  $G$  with  $\omega(G) = k$ , we have  $\chi(G) \leq \binom{k+1}{2}$ .

**Proof.** Fix an ordering  $<$  of  $V(G)$  such that  $G_{<}$  is a semi-comparability graph. For every  $v \in V(G)$ , let  $f(v)$  denote the size of the largest clique with minimal element  $v$ . Then  $f(v) \in [k]$ . For  $i = 1, \dots, k$ , let  $V_i = \{v \in G : f(v) = i\}$ .

The main observation is that  $G[V_i]$  is a partial order. Indeed, suppose to the contrary that there exist 3 vertices  $a, b, c \in V_i$  such that  $a < b < c$  and  $ab, bc \in E(G)$ , but  $ac \notin E(G)$ . Let  $C \subset V(G)$  be a clique of size  $i$  with minimal element  $c$ . If  $d \in C \setminus \{c\}$ , then  $b$  and  $d$  must be joined by an edge, otherwise the quadruple  $a, b, c, d$  satisfies the conditions  $ab, bc, cd \in E(G)$  and  $ac, bd \notin E(G)$ . Thus,  $b$  is joined to every vertex in  $C$  by an edge, which means that  $C \cup \{b\}$  is a clique of size  $i + 1$  with minimal element  $b$ , contradicting our assumption that  $b \in V_i$ .

Hence, every  $G[V_i]$  is a partial order. Using the fact that  $G[V_i]$  does not contain a clique of size  $i + 1$ , by Mirsky's theorem [7] we obtain that  $\chi(G[V_i]) \leq i$ . Summing up for all  $i$ , we get that  $\chi(G) \leq \sum_{i=1}^k \chi(G[V_i]) \leq \binom{k+1}{2}$ , as required. ◀

The combination of Lemmas 7 and 8 immediately implies Theorem 1.

Next, we prove the upper bound in Theorem 5.

► **Theorem 9.** *Let  $G$  be the disjointness graph of a collection of  $x$ -monotone curves with  $\omega(G) = k$ . Then we have  $\chi(G) \leq k^2 \binom{k+1}{2}$ .*

**Proof.** Let  $\mathcal{C}$  be a collection of  $x$ -monotone curves satisfying the conditions in the theorem. For any  $\gamma \in \mathcal{C}$ , let  $x(\gamma)$  denote the projection of  $\gamma$  to the  $x$ -axis. For  $\alpha, \beta \in \mathcal{C}$ , let  $\alpha \prec \beta$  if  $\min x(\alpha) < \min x(\beta)$  and  $\max x(\alpha) < \max x(\beta)$ .

Suppose that  $\alpha$  and  $\beta$  are disjoint. Let  $\alpha <_1 \beta$  if  $\alpha \prec \beta$  and  $\alpha$  is below  $\beta$ , that is, if on every vertical line that intersects both  $\alpha$  and  $\beta$ , the intersection point of  $\alpha$  lies below the intersection point of  $\beta$ . Let  $\alpha <_2 \beta$  if  $\alpha \prec \beta$  and  $\beta$  is below  $\alpha$ . Clearly,  $<_1$  and  $<_2$  are partial orders.

As  $\omega(G) \leq k$ , the size of the longest chains with respect to  $<_1$  and  $<_2$  is at most  $k$ . Therefore, the vertices of  $G$  can be colored with  $k^2$  colors such that each color class is an antichain in both  $<_1$  and  $<_2$ .

It remains to show that each of these color classes can be properly colored with  $\binom{k+1}{2}$  colors. Let  $\mathcal{C}' \subset \mathcal{C}$  such that no two elements of  $\mathcal{C}'$  are comparable by  $<_1$  or  $<_2$ . Then, if  $\alpha, \beta \in \mathcal{C}'$ , then either  $\alpha$  and  $\beta$  intersect, or one of the intervals  $x(\alpha)$  or  $x(\beta)$  contains the other. In either case,  $x(\alpha)$  and  $x(\beta)$  have a nonempty intersection, so any two elements of  $\{x(\gamma) : \gamma \in \mathcal{C}'\}$  intersect. Hence,  $\bigcap_{\gamma \in \mathcal{C}'} x(\gamma)$  is nonempty, and there exists a vertical line  $l$  that intersects every element of  $\mathcal{C}'$ .

Let  $G'$  denote the disjointness graph of  $\mathcal{C}'$ . Order the elements of  $\mathcal{C}'$  with respect to their intersections with  $l$ , from bottom to top. We claim that the resulting ordered graph  $G'_<$  is a *semi-comparability graph*. Indeed, suppose to the contrary that there are four vertices  $a, b, c, d \in V(G')$  such that  $a < b < c < d$  and  $ab, bc, cd \in E(G')$ , but  $ac, bd \notin E(G')$ . Without loss of generality, suppose that the length of  $x(b)$  is larger than the length of  $x(c)$ ; the other case can be handled similarly. As  $bc \in E(G')$ , we have  $x(c) \subset x(b)$  and  $b$  is below  $c$ , so every vertical line intersecting  $c$  intersects  $b$  as well, and its intersection with  $b$  lies below its intersection with  $c$ . Also, as  $ab \in E(G')$ , we have that  $a$  is below  $b$ . But then  $a$  and  $c$  must be disjoint, contradicting the condition  $ac \notin E(G')$ .

Thus, we can apply Lemma 8 to conclude that  $G'$  can be properly colored with  $\binom{k+1}{2}$  colors. This completes the proof. ◀

Let  $g(n)$  denote the maximal number  $m$  such that every collection of  $n$  convex sets in the plane contains  $m$  elements that are either pairwise disjoint, or pairwise intersecting. Larman et al. [26] proved that  $g(n) \geq n^{1/5}$ , while the best known upper bound, due to Kynčl [25] is  $g(n) < n^{\log 8 / \log 169} \approx n^{0.405}$ . Theorem 9 implies the following modest improvement on the lower bound.

► **Corollary 10.** *Every collection of  $n$   $x$ -monotone curves (or convex sets) in the plane contains  $((2 + o(1))n)^{1/5} \approx 1.15n^{1/5}$  elements that are either pairwise disjoint or pairwise intersecting.*

**Proof.** In every graph  $G$  on  $n$  vertices, we have  $\alpha(G)\chi(G) \geq n$ . In view of Theorem 9, this implies that if  $\mathcal{C}$  is a collection of  $n$   $x$ -monotone curves and  $G$  is the disjointness graph of  $\mathcal{C}$ , then we have

$$\alpha(G)(\omega(G))^3 \frac{\omega(G) + 1}{2} \geq n.$$

Therefore,  $\max\{\alpha(G), \omega(G)\} \geq ((2 + o(1))n)^{1/5}$ , as claimed. ◀

Many attempts were made to improve the order of magnitude of the lower bound on  $g(n)$ . It appeared to be conceivable to cover the disjointness graph  $G$  of any collection of  $x$ -monotone curves with fewer than 4 comparability graphs, which would have yielded  $\chi(G) \leq (\omega(G))^3$  and  $g(n) \geq n^{1/4}$ . These hopes are shattered by Theorem 4.

### 3 Magical graphs – Proof of Theorem 2

The converse of Lemma 7 is not true: not every semi-comparability graph can be realized as the disjointness graph of a collection of grounded  $x$ -monotone curves. See Section 6, for further discussion. To characterize such disjointness graphs, we need to introduce a new family of graphs.

A graph  $G_{<_1, <_2}$  with two total orderings,  $<_1$  and  $<_2$ , on its vertex set is called *double-ordered*. If the orderings  $<_1, <_2$  are clear from the context, we shall write  $G$  instead of  $G_{<_1, <_2}$ .

► **Definition 11.** A double-ordered graph  $G_{<_1, <_2}$  is called *magical* if for any three distinct vertices  $a, b, c \in V(G)$  with  $a <_1 b <_1 c$ , if  $ab, bc \in E(G)$  and  $ac \notin E(G)$ , then  $b <_2 a$  and  $b <_2 c$ .

A graph  $G$  is said to be *magical*, if there exist two total orders  $<_1, <_2$  on  $V(G)$  such that  $G_{<_1, <_2}$  is magical. In this case, we say that the pair  $(<_1, <_2)$  witnesses  $G$ .

It easily follows from the above definition that if  $G_{<_1, <_2}$  is magical, then  $G_{<_1}$  is a semi-comparability graph.

► **Lemma 12.** If  $\mathcal{C}$  is a collection of grounded  $x$ -monotone curves, then the disjointness graph of  $\mathcal{C}$  is magical.

**Proof.** Let  $G$  be the disjointness graph of  $\mathcal{C}$ , and identify the vertices of  $G$  with the elements of  $\mathcal{C}$ . For any  $\gamma \in \mathcal{C}$ , let  $(0, y_\gamma)$  be the endpoint of  $\gamma$  lying on the vertical axis  $\{x = 0\}$ , and let  $(x_\gamma, y'_\gamma)$  be the other endpoint of  $\gamma$ .

Define the total orderings  $<_1$  and  $<_2$  on  $V(G)$ , as follows. Let  $\gamma <_1 \gamma'$  if and only if  $y_\gamma < y_{\gamma'}$ , and let  $\gamma <_2 \gamma'$  if and only if  $x_\gamma < x_{\gamma'}$ .

Suppose that for a triple  $a, b, c \in \mathcal{C}$  we have that  $a <_1 b <_1 c$  and  $ab, bc \in E(G)$ , but  $ac \notin E(G)$ . Then  $a$  and  $c$  intersect. Hence,  $a, c$ , and the ground curve  $\{x = 0\}$  enclose a region  $A$ , and  $b \subset A$ . This implies that the  $x$ -coordinate of the right endpoint of  $b$  is smaller than the  $x$ -coordinates of the right endpoints of  $a$  and  $c$ . Therefore, we have  $b <_2 a$  and  $b <_2 c$ , showing that  $G$  is magical. ◀

► **Lemma 13.** Let  $G$  be a magical graph. Then there exists a family  $\mathcal{C}$  of grounded  $x$ -monotone curves such that the disjointness graph of  $\mathcal{C}$  is isomorphic to  $G$ .

**Proof.** Let  $n$  be the number of vertices of  $G$ . Let  $<_1$  and  $<_2$  be total orderings on  $V(G)$  witnessing that  $G$  is magical. For any vertex  $v \in V(G)$ , let  $y(v) \in [n]$  denote the position of  $v$  in the ordering  $<_1$ , and let  $x(v)$  denote the position of  $v$  in the ordering  $<_2$ .

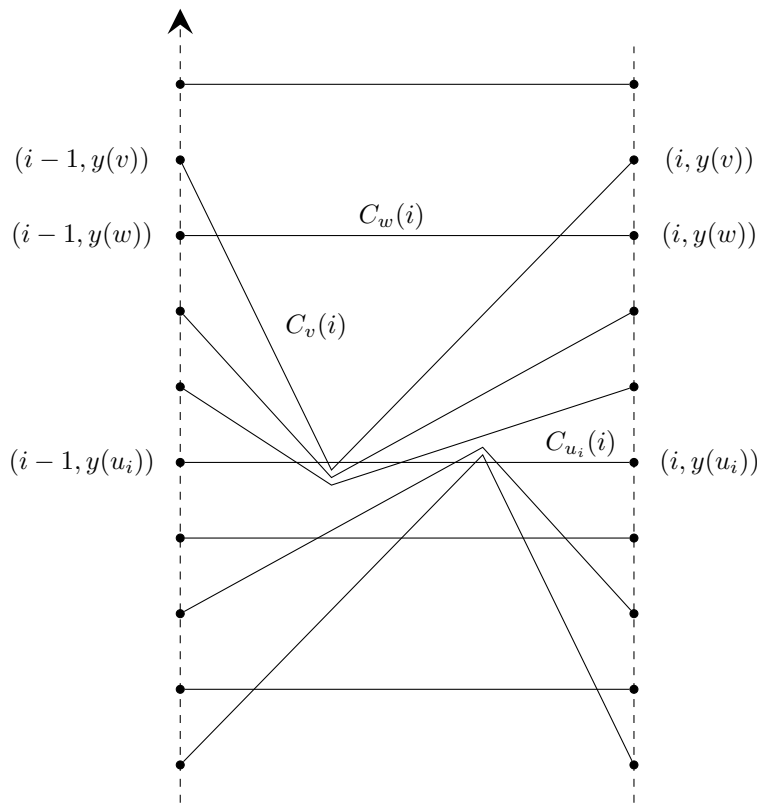
For any  $v \in V(G)$ , we define an  $x$ -monotone curve  $C_v$ , which will be composed of  $x(v)$  smaller  $x$ -monotone pieces,  $C_v(1), \dots, C_v(x(v))$ , such that  $C_v(i)$  starts at the point  $(i-1, y(v))$ , and ends at the point  $(i, y(v))$ . The pieces  $C_v(i)$  are defined as follows.

Let  $u_i \in V(G)$  be such that  $x(u_i) = i$ . If  $u_i = v$  or there is an edge between  $u_i$  and  $v$ , then let  $C_v(i)$  be the horizontal line segment connecting  $(i-1, y(v))$  and  $(i, y(v))$ . Otherwise, let  $C_v(i)$  be the polygonal curve consisting of two segments whose three vertices are

$$(i-1, y(v)), \quad \left(i - \frac{2}{3}, y(u_i) - \frac{1}{10} + \frac{y(v)}{10n}\right), \quad (i, y(v)) \quad \text{if } y(u_i) < y(v),$$

or

$$(i-1, y(v)), \quad \left(i - \frac{1}{3}, y(u_i) + \frac{y(v)}{10n}\right), \quad (i, y(v)) \quad \text{if } y(u_i) > y(v).$$



■ **Figure 1** An illustration of the curves  $C_v(i)$  in the proof of Lemma 13.

See Figure 1 for an illustration. One can easily check the following property of the curves  $\{C_v(i)\}_{v \in V(G)}$ . If  $v, w \in V(G)$  are distinct vertices such that  $C_v(i)$  and  $C_w(i)$  intersect, then

- (i)  $x(v), x(w) \geq i$ .
- (ii) Exactly one of  $v$  and  $w$  is joined to  $u_i$  in  $G$ . Without loss of generality, assume that it is  $w$ .
- (iii) Then  $y(u_i) \leq y(w) < y(v)$  or  $y(v) < y(w) \leq y(u_i)$ .

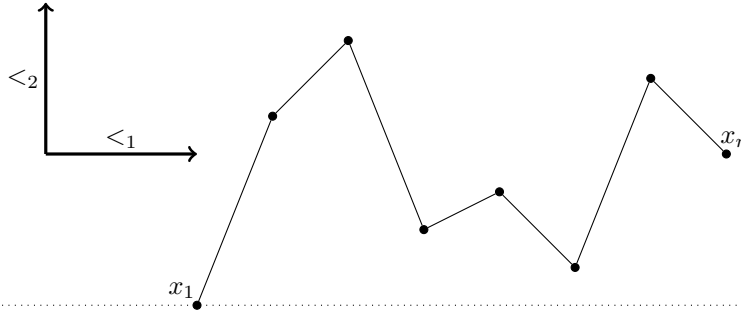
Now we show that  $G$  is the disjointness graph of  $\mathcal{C} = \{C_v : v \in V(G)\}$ .

If  $v$  and  $w$  are not joined by an edge in  $G$ , then  $C_v(\min\{x(v), x(w)\})$  and  $C_w(\min\{x(v), x(w)\})$  intersect by definition, so  $C_v$  and  $C_w$  have a nonempty intersection.

Our task is reduced to showing that if  $v$  and  $w$  are joined by an edge, then  $C_v$  and  $C_w$  do not intersect. Suppose to the contrary that  $C_v$  and  $C_w$  intersect. Then there exists  $i \in [\min\{x(v), x(w)\} - 1]$  such that  $C_v(i)$  and  $C_w(i)$  intersect. Then either  $y(u_i) \leq y(v), y(w)$  or  $y(u_i) \geq y(v), y(w)$ . Without loss of generality, let  $y(u_i) \leq y(v), y(w)$ , the other case can be handled in a similar manner. Again, without loss of generality, we can assume that  $y(w) < y(v)$ . Then  $C_v(i)$  intersects  $C_{u_i}(i)$ , and  $C_w(i)$  is disjoint from  $C_{u_i}(i)$ , or equivalently,  $u_i w \in E(G)$ , but  $u_i v \notin E(G)$ . However, this is impossible, because  $vw \in E(G)$ , so the triple  $u_i, w, v$  would contradict the assumption that  $G$  is magical. ◀

By Lemma 13, in order to prove Theorem 2, it is enough to verify the corresponding statement for magical graphs. In other words, we have to prove the following.

► **Theorem 14.** *For every positive integer  $k \geq 2$ , there exists a magical graph  $G$  such that  $\omega(G) = k$  and  $\chi(G) = \binom{k+1}{2}$ .*



■ **Figure 2** A mountain path. The dotted line shows the minimum of  $x_1$  and  $x_r$  in  $<_2$ , so all the other points of the path must be above it.

The rest of this section is devoted to the proof of this theorem. The proof is probabilistic and is inspired by a construction of Korándi and Tomon [18]. We shall consider a random double-ordered graph with certain parameters, and show that the smallest magical graph covering its edges meets the requirements in Theorem 14. To accomplish this plan, we first examine how the smallest magical graph covering the edges of a given double-ordered graph looks.

Let  $G_{<_1, <_2}$  be a double-ordered graph. A sequence of vertices  $x_1, \dots, x_r \in V(G)$  is said to form a *mountain-path*, if  $x_1 <_1 \dots <_1 x_r$ ,  $x_i x_{i+1} \in E(G)$  for every  $i$ , where  $1 \leq i < r$ , and either  $x_1 <_2 x_2, \dots, x_{r-1}$  or  $x_r <_2 x_2, \dots, x_{r-1}$ . See Figure 2.

► **Lemma 15.** *Let  $G_{<_1, <_2}$  be a double-ordered graph. There exists a unique minimal graph  $G'_{<_1, <_2}$  on  $V(G)$  such that  $E(G) \subset E(G')$  and  $G'_{<_1, <_2}$  is magical. Moreover, if  $u, v \in V(G)$ , then  $u$  and  $v$  are joined by an edge in  $G'$  if and only if there exists a mountain-path connecting  $u$  and  $v$ .*

**Proof.** Let  $H = H_{<_1, <_2}$  be any magical graph on the vertex set  $V(G)$  such that  $E(G) \subseteq E(H)$ . Let  $x_1, \dots, x_r$  be a mountain-path in  $G$  with  $x_1 <_2 x_r$ . Using the definition of magical graphs, it is easy to prove by induction on  $i$  that  $x_1$  and  $x_i$  are joined by an edge in  $E(H)$ , for every  $i > 1$ . Therefore, we have  $x_1 x_r \in E(H)$ . (We can proceed similarly if  $x_r <_2 x_1$ .)

With a slight abuse of notation, from now on let  $H = H_{<_1, <_2}$  denote the double-ordered graph on  $V(G)$ , in which  $u$  and  $v$  are joined by an edge if and only if there exists a mountain-path connecting  $u$  to  $v$ . We will show that  $H$  is magical, that is, for every triple  $u, v, w \in V(G)$ , the following holds: if  $u <_1 v <_1 w$  such that  $uv, vw \in E(H)$ , and  $u <_2 v$  or  $w <_2 v$ , then  $uw \in E(H)$ . As  $uv, vw \in E(H)$ , there exist two mountain-paths  $u = x_1, x_2, \dots, x_r = v$  and  $v = x_r, x_{r+1}, \dots, x_s = w$ . By the assumption that  $u <_2 v$  or  $w <_2 v$ , the path  $u = x_1, \dots, x_s = w$  is a mountain-path as well, so  $uw \in E(H)$ . ◀

For the rest of the discussion, we need to introduce a few parameters that depend on  $k$ . Set  $\lambda = 1/k^2$ ,  $t = 100k^2 \log k$ ,  $h = 3t^{k^2} k^{2k^2+8}$ ,  $n = 9h$  and  $p = t/n$ .

Let  $S = \{(a, b) \in [k]^2 : a + b \geq k + 1\}$ . For each  $(a, b) \in S$ , let  $A_{a,b}$  be a set of  $n$  arbitrary points in the interior of the unit square  $[ak + b, ak + b + 1] \times [bk + a, bk + a + 1]$  with distinct  $x$  and  $y$  coordinates, see Figure 3. Let  $V = \bigcup_{(a,b) \in S} A_{a,b}$ , and let  $<_1$  and  $<_2$  be the total orderings on  $V$  induced by the  $x$  and  $y$  coordinates of the elements of  $V$ , respectively. A pair of vertices  $\{u, v\}$  in  $V$  is called *available* if  $u \in A_{a,b}, v \in A_{a',b'}$  with  $(a, b) \neq (a', b')$ .

Let  $G_0$  denote the *random graph* on  $V$  in which every available pair of vertices is connected by an edge with probability  $p$ , independently from each other.  $G_0$  does not have any edge whose endpoints belong to the same set  $A_{a,b}$ . Let  $G'_{<_1, <_2}$  be the minimal magical graph on  $V$  containing all edges of  $G_0$ .



## 54:10 On the Chromatic Number of Disjointness Graphs of Curves

Consider the graph  $P = P_{uv} \cup P_{vw} \cup P_{uw}$ . It is a connected graph, but not a tree, because there are two distinct paths between  $u$  and  $w$ :  $P_{uv} \cup P_{vw}$  and  $P_{uw}$ . Hence, we have  $|E(P)| \geq |V(P)|$ . Let  $\mathcal{P}$  denote the set of all such graphs  $P$  that appear in  $G_0$  with positive probability. Then

$$\begin{aligned} \mathbb{P}(\{u, v, w\} \text{ induces a triangle in } G') &= \mathbb{P}(P \text{ is a subgraph of } G_0 \text{ for some } P \in \mathcal{P}) \\ &\leq \sum_{P \in \mathcal{P}} \mathbb{P}(P \text{ is a subgraph of } G_0). \end{aligned}$$

For a fixed  $P \in \mathcal{P}$ , every edge of  $P$  is present in  $G_0$  independently with probability  $p$ . Hence, the probability that  $P$  is a subgraph of  $G_0$  is  $p^{|E(P)|}$ , which is at most  $p^{|V(P)|}$ . The number of graphs in  $\mathcal{P}$  with exactly  $m$  vertices is at most  $\binom{|V|}{m-3} < (k^2 n)^{m-3}$ , as each member of  $\mathcal{P}$  contains the vertices  $u, v, w$ . Finally, every member of  $\mathcal{P}$  has at most  $3|S| \leq 3k^2$  vertices, so we can write

$$\sum_{P \in \mathcal{P}} \mathbb{P}(P \text{ is a subgraph of } G_0) \leq \sum_{m=3}^{3k^2} p^m (k^2 n)^{m-3} < 3t^{k^2} k^{2k^2+2} n^{-3}.$$

Since the number of holes in  $V$  is at most  $\binom{|V|}{3} < |V|^3 < k^6 n^3$ , we obtain

$$\mathbb{E}(N) < 3t^{k^2} k^{2k^2+8} = h. \quad \triangleleft$$

Applying Markov's inequality, the probability that  $V$  contains more than  $3h$  holes that induce a triangle in  $G'$  is at most  $1/3$ . Hence, there exists a magical graph  $G'$  on  $V$  such that  $G'$  has no independent set of size  $(1 + \lambda)n$ , and  $G'$  contains at most  $3h$  triangles whose vertices form a hole. By deleting a vertex of each such hole in  $G'$ , we obtain a magical graph  $G$  with at least  $|S|n - 3h$  vertices, which has no triangle whose vertices form a hole, and no independent set of size  $(1 + \lambda)n$ .

First, we show that  $\chi(G) \geq |S| = \binom{k+1}{2}$ . Indeed, if  $\chi(G) \leq |S| - 1$ , then  $G$  contains an independent set of size

$$\frac{|V(G)|}{|S| - 1} \geq \frac{|S|n - 3h}{|S| - 1} = \left(1 + \frac{1}{|S| - 1}\right)n - \frac{3h}{|S| - 1} > (1 + \lambda)n,$$

contradiction.

It remains to prove that  $\omega(G) = k$ . Clearly,  $\omega(G) \geq k$ , otherwise, by Lemma 8, we would have  $\chi(G) \leq \binom{k}{2}$ , contradicting the last paragraph. Thus, we have to show that  $G$  has no clique of size  $k + 1$ . For this, we need the following observation.

▷ **Claim 18.** Let  $K$  be a subset of  $S$  that does not contain three points  $(a_1, b_1), (a_2, b_2), (a_3, b_3)$  such that  $a_1 < a_2 \leq a_3$  and  $b_2 \leq b_1$  and  $b_2 < b_3$ . Then we have  $|K| \leq k$ .

*Proof.* We call  $(a_1, b_1), (a_2, b_2), (a_3, b_3)$  a *bad triple*, if  $a_1 < a_2 \leq a_3$  and  $b_2 \leq b_1$  and  $b_2 < b_3$ .

Let  $S = S_k$ . We prove the claim by induction on  $k$ . For  $k = 1$ , the claim is trivial. Suppose that  $k \geq 2$  and that the statement has already been verified for  $k - 1$ . We distinguish two cases.

*Case 1:*  $K$  contains at most 1 element from the column  $\{(k, b) : b \in [k]\}$ . Let

$$K' = \{(a, b) : (a, b + 1) \in K \text{ and } a < k\}.$$

Then  $|K'| \geq |K| - 1$  and  $K' \subset S_{k-1}$  does not contain a bad triple. Thus, by the induction hypothesis, we have  $|K'| \leq k - 1$ , which implies that  $|K| \leq k$ .



Case 2:  $K$  contains 2 distinct elements of the form  $(k, b)$  and  $(k, b')$ , where  $b < b'$ . Then  $K$  cannot contain  $(a, k)$  for any  $a \in [k - 1]$ , otherwise  $(a, k), (k, b), (k, b')$  would be a bad triple. Thus,  $K$  contains at most one element from the row  $\{(a, k) : a \in [k]\}$  (it might contain  $(k, k)$ ). Let

$$K' = \{(a, b) : (a + 1, b) \in K \text{ and } b \leq k - 1\}.$$

Again,  $|K'| \geq |K| - 1$  and  $K' \subset S_{k-1}$  does not contain a bad triple. By the induction hypothesis, we have  $|K'| \leq k - 1$  and, hence,  $|K| \leq k$ .  $\triangleleft$

Now we are in a position to finish the proof of Theorem 14. Let  $G$  denote the magical graph obtained from  $G'$  by deleting a vertex from each of its holes that form a triangle (see right before Claim 18). Suppose that  $C \subset V$  is a clique in  $G$ . Then  $C$  does not contain a hole and it intersects each  $A_{a,b}$  in at most one vertex. Let  $K = \{(a, b) \in S : A_{a,b} \cap C \neq \emptyset\}$ . The condition that  $C$  does not contain a hole implies that  $K$  does not contain three points  $(a_1, b_1), (a_2, b_2), (a_3, b_3)$  such that  $a_1 < a_2 \leq a_3$  and  $b_2 \leq b_1$  and  $b_2 < b_3$ . Hence, by Claim 18, we have  $|C| = |K| \leq k$ . This completes the proof of Theorem 14 and, hence, the proof of Theorem 2.

#### 4 Bounding function for curves that intersect a vertical line—Proof of Theorem 3

A triple-ordered graph is a graph  $G_{<_1, <_2, <_3}$  with three total orders  $<_1, <_2, <_3$  on its vertex set.

► **Definition 19.** A triple-ordered graph  $G_{<_1, <_2, <_3}$  is called double-magical, if there exist two magical graphs  $G_{<_1, <_2}^1$  and  $G_{<_1, <_3}^2$  on  $V(G)$  such that  $E(G_{<_1, <_2, <_3}) = E(G_{<_1, <_2}^1) \cap E(G_{<_1, <_3}^2)$ . An unordered graph  $G$  is said to be double-magical, if there exist three total orders  $<_1, <_2, <_3$  on  $V(G)$  such that the triple-ordered graph  $G_{<_1, <_2, <_3}$  is double-magical. We say that  $G$  is witnessed by  $(<_1, <_2, <_3)$ .

By Lemmas 12 and 13, it is not hard to characterize disjointness graphs of  $x$ -monotone curves intersected by a vertical line.

► **Lemma 20.** Let  $\mathcal{C}$  be a collection of  $x$ -monotone curves such that each member of  $\mathcal{C}$  intersects the vertical line  $l$ . Then the disjointness graph of  $\mathcal{C}$  is double-magical.

**Proof.** Without loss of generality, let  $l = \{x = 0\}$ . For each  $\gamma \in \mathcal{C}$ , let  $(-x_\gamma^-, y_\gamma^-)$  be the left endpoint of  $\gamma$ , let  $(0, y_\gamma)$  be the intersection point of  $\gamma$  and  $l$ , and let  $(x_\gamma^+, y_\gamma^+)$  be the right endpoint of  $\gamma$ . Also, let  $\gamma^- = \gamma \cap \{x \leq 0\}$  and  $\gamma^+ = \gamma \cap \{x \geq 0\}$ , and let  $\mathcal{C}^- = \{\gamma^- : \gamma \in \mathcal{C}\}$  and  $\mathcal{C}^+ = \{\gamma^+ : \gamma \in \mathcal{C}\}$ . Then  $\mathcal{C}^+$  is a collection of grounded curves, and  $\mathcal{C}^-$  is the reflection of a collection of grounded curves to the line  $l$ .

Let  $G, G^-$  and  $G^+$  be the disjointness graphs of  $\mathcal{C}, \mathcal{C}^-$  and  $\mathcal{C}^+$ , respectively, such that we identify  $\gamma, \gamma^-$  and  $\gamma^+$  as the vertices of these graphs for every  $\gamma \in \mathcal{C}$ . Then  $E(G) = E(G^-) \cap E(G^+)$ . Let  $<_1$  be the total ordering on  $\mathcal{C}$  defined by  $\gamma <_1 \gamma'$  if  $y_\gamma < y_{\gamma'}$ , let  $<_2$  be the ordering defined by  $\gamma <_2 \gamma'$  if  $x_\gamma^- < x_{\gamma'}^-$ , and let  $<_3$  be the ordering defined by  $\gamma <_3 \gamma'$  if  $x_\gamma^+ < x_{\gamma'}^+$ . By Lemma 12,  $G_{<_1, <_2}^-$  and  $G_{<_1, <_3}^+$  are magical, so  $G_{<_1, <_2, <_3}$  is double-magical.  $\blacktriangleleft$

We can just as easily prove the converse of Lemma 20, using Lemma 13.

► **Lemma 21.** Let  $G$  be a double-magical graph. Then there exists a collection of curves  $\mathcal{C}$  such that each member of  $\mathcal{C}$  has a nonempty intersection with the vertical line  $\{x = 0\}$ , and the disjointness graph of  $\mathcal{C}$  is isomorphic to  $G$ .

## 54:12 On the Chromatic Number of Disjointness Graphs of Curves

**Proof.** Let  $(\prec_1, \prec_2, \prec_3)$  be total orders on  $V(G)$  witnessing that  $G$  is double-magical, and let  $G_{\prec_1, \prec_2}^1, G_{\prec_1, \prec_3}^2$  be two magical graphs on  $V(G)$  such that  $E(G) = E(G^1) \cap E(G^2)$ .

Let  $|V(G)| = n$ . By Lemma 13, there exist  $n$  grounded  $x$ -monotone curves  $\gamma_1^+, \dots, \gamma_n^+$  such that  $\gamma_i^+$  is contained in the nonnegative plane  $\{x \geq 0\}$  with one endpoint at  $(0, i)$ , the disjointness graph of  $\{\gamma_1^+, \dots, \gamma_n^+\}$  is  $G^1$ , and  $\gamma_i^+$  corresponds to the  $i$ -th vertex of  $G^1$  in the order  $\prec_1$ . Also, there exist  $n$   $x$ -monotone curves  $\gamma_1^-, \dots, \gamma_n^-$  such that  $\gamma_i^-$  is contained in the nonpositive plane  $\{x \leq 0\}$  with one endpoint at  $(0, i)$ , the disjointness graph of  $\{\gamma_1^-, \dots, \gamma_n^-\}$  is  $G^2$ , and  $\gamma_i^-$  corresponds to the  $i$ -th vertex of  $G^2$  in the order  $\prec_1$ . For  $i = 1, \dots, n$ , set  $\gamma_i = \gamma_i^- \cup \gamma_i^+$ , then the disjointness graph of  $\mathcal{C} = \{\gamma_i : i \in [n]\}$  is isomorphic to  $G$ , and every curve in  $G$  has a nonempty intersection with the vertical line  $\{x = 0\}$ .  $\blacktriangleleft$

For any double-magical graph  $G = G_{\prec_1, \prec_2, \prec_3}$ , define four partial orders  $\prec_1, \prec_2, \prec_3, \prec_4$  on  $V(G)$ , as follows. For  $a, b \in V(G)$ , let

- (i)  $a \prec_1 b$  if  $a \prec_1 b, a \prec_2 b, a \prec_3 b$ , and  $ab \in E(G)$ ;
- (ii)  $a \prec_2 b$  if  $a \prec_1 b, b \prec_2 a, b \prec_3 a$ , and  $ab \in E(G)$ ;
- (iii)  $a \prec_3 b$  if  $a \prec_1 b, a \prec_2 b, b \prec_3 a$ , and  $ab \in E(G)$ ;
- (iv)  $a \prec_4 b$  if  $a \prec_1 b, b \prec_2 a, a \prec_3 b$ , and  $ab \in E(G)$ .

It follows easily from the definition of double-magical graphs that these are indeed partial orders. Moreover, they satisfy the following conditions.

- (1) If  $ab \in E(G)$ , then  $a$  and  $b$  are comparable by precisely one of these 4 partial orders.
- (2) For any  $a, b, c \in V(G)$  and  $i \in [4]$ , if  $a \prec_i b$  and  $b \prec_i c$ , then  $ac \in E(G)$ .
- (3) For any  $a, b, c \in V(G)$  and  $i \in [4]$ , if  $a \prec_i b$  and  $b \prec_{2-i} c$ , then  $ac \in E(G)$ .

► **Theorem 22.** *Let  $G$  be a double-magical graph. If  $\omega(G) = k$ , then  $\chi(G) \leq \frac{k+1}{2} \binom{k+2}{3}$ .*

**Proof.** Let  $\prec_1, \prec_2, \prec_3$  be total orders on  $V(G)$  witnessing  $G$ , and let  $\prec_1, \prec_2, \prec_3, \prec_4$  denote the partial orders defined above. Clearly, there is no chain of length  $k+1$  with respect to any of the partial orders  $\prec_i$ , because that would contradict the assumption  $\omega(G) = k$ .

For  $h = 1, \dots, k$ , let  $S_h$  denote the set of vertices  $v \in V(G)$  for which the size of a longest  $\prec_1$ -chain with maximal element  $v$  is  $k-h+1$ . Then the sets  $S_1, \dots, S_k$  form a partition of  $V(G)$ , where each  $S_h$  is a  $\prec_1$ -antichain that contains no clique of size  $h+1$ . Indeed, suppose that  $C \subset S_h$  induces a clique of size  $h+1$  in  $G$ , and consider the smallest vertex  $v \in C$  with respect to the order  $\prec_1$ . There exists a  $\prec_1$ -chain  $D$  of size  $k-h+1$  ending at  $v$ . This implies that for every  $a \in D$  and  $b \in C$ , we have  $a \prec_1 v$  and  $v \prec_i b$  for some  $i \in \{2, 3, 4\}$ . Then, by (2), we would have  $ab \in E(G)$ . Hence,  $D \cup C$  would induce a clique of size  $k+1$ , contradiction.

For  $h = 1, \dots, k$  and  $m = 1, \dots, h$ , let  $S_{h,m}$  denote the set of vertices in  $S_h$  for which the largest  $\prec_2$ -chain in  $S_h$  with smallest element  $v$  has size  $h-m+1$ . As  $\omega(G[S_h]) \leq h$ , the sets  $S_{h,1}, \dots, S_{h,h}$  are  $\prec_1$ - and  $\prec_2$ -antichains partitioning  $S_h$ . Further,  $S_{h,m}$  contains no clique of size  $m+1$ . Otherwise, if  $C \subset S_{h,m}$  forms a clique of size  $m+1$  in  $G$ , then consider the largest vertex  $v \in C$  with respect to the order  $\prec_1$ . There exists a  $\prec_2$ -chain  $D$  of size  $h-m+1$  whose smallest element is  $v$ . Hence, for every  $a \in C$  and  $b \in D$ , we have  $a \prec_i v$  and  $v \prec_2 b$  for some  $i \in \{3, 4\}$ , which implies, by (3), that  $ab \in E(G)$ . Hence,  $C \cup D$  would induce a clique of size  $h+1$  in  $S_h$ , contradiction.

Thus, we obtained that  $S_{h,m}$  is a  $\prec_1$ - and  $\prec_2$ -antichain, which does not contain a clique of size  $m+1$ . In particular, the size of the longest  $\prec_3$ - and  $\prec_4$ -chains in  $S_{h,m}$  is at most  $m$ . This means that  $G[S_{h,m}]$  can be properly colored with  $m^2$  colors. Indeed, set the color of  $v \in S_{h,m}$  to be  $\phi(v) = (r, q)$ , where  $r$  is the size of the largest  $\prec_3$ -chain with smallest element  $v$ , and  $q$  is the size of the largest  $\prec_4$ -chain with smallest element  $v$ . Then  $\phi : S_{h,m} \rightarrow [m]^2$  is a proper coloring of  $G[S_{h,m}]$ .

As  $S_h = \bigcup_{m=1}^h S_{h,m}$ , we have

$$\chi(G[S_h]) \leq \sum_{m=1}^h \chi(G[S_{h,m}]) \leq \sum_{m=1}^h m^2 = \frac{h(h+1)(2h+1)}{6}.$$

Finally, since  $V(G) = \bigcup_{h=1}^k S_h$ , we obtain

$$\chi(G) \leq \sum_{h=1}^k \chi(G[S_h]) \leq \sum_{h=1}^k \frac{h(h+1)(2h+1)}{6} = \frac{k+1}{2} \binom{k+2}{3}. \quad \blacktriangleleft$$

## 5 Construction of double-magical graphs—Proof of Theorem 4

In view of 21, to prove Theorem 4, it is enough to construct a double-magical graph with the desired clique and chromatic numbers.

► **Theorem 23.** *For every positive integer  $k \geq 2$ , there exists a double-magical graph  $G$  satisfying  $\omega(G) = k$  and  $\chi(G) = \frac{k+1}{2} \binom{k+2}{3}$ .*

In the rest of this section, we prove this theorem. The proof of Lemma 22 reveals a lot about the structure of double-magical graphs satisfying the properties of Theorem 23, if they exist. To construct them, we use reverse engineering.

For any vector  $\mathbf{v} \in \mathbb{R}^d$  and any  $j \in [d]$ , let  $\mathbf{v}(j)$  denote the  $j$ th coordinate of  $\mathbf{v}$ . The *sign vector* of  $\mathbf{v} \in \mathbb{R}^d$  is the  $d$ -dimensional vector  $\text{sg}(\mathbf{v})$  with

$$\text{sg}(\mathbf{v})(i) = \begin{cases} 1 & \text{if } \mathbf{v}(i) > 0, \\ -1 & \text{if } \mathbf{v}(i) < 0, \\ 0 & \text{if } \mathbf{v}(i) = 0. \end{cases}$$

Let  $\mathbf{v}_1 = (1, 1, 1)$ ,  $\mathbf{v}_2 = (1, -1, -1)$ ,  $\mathbf{v}_3 = (1, 1, -1)$  and  $\mathbf{v}_4 = (1, -1, 1)$ . For any  $\mathbf{i} \in [k]^4$ , let

$$P(\mathbf{i}) = k^3 \mathbf{i}(1) \mathbf{v}_1 + k^2 \mathbf{i}(2) \mathbf{v}_2 + k \mathbf{i}(3) \mathbf{v}_3 + \mathbf{i}(4) \mathbf{v}_4.$$

These  $k^4$  points have the useful property that if  $\mathbf{i} \neq \mathbf{i}'$ , then the relative position of  $P(\mathbf{i})$  and  $P(\mathbf{i}')$  depends only on the smallest coordinate in which  $\mathbf{i}$  and  $\mathbf{i}'$  differ. We refer to this property as the *LEX property* (short for “lexicographic”), which is formally defined as follows.

**LEX property.** Let  $\mathbf{i}, \mathbf{i}' \in [k]^4$  be such that  $\mathbf{i} \neq \mathbf{i}'$ , and let  $r$  be the smallest index such that  $\mathbf{i}(r) \neq \mathbf{i}'(r)$ . If  $\mathbf{i}(r) > \mathbf{i}'(r)$ , then

$$\text{sg}(P(\mathbf{i}) - P(\mathbf{i}')) = \mathbf{v}_r.$$

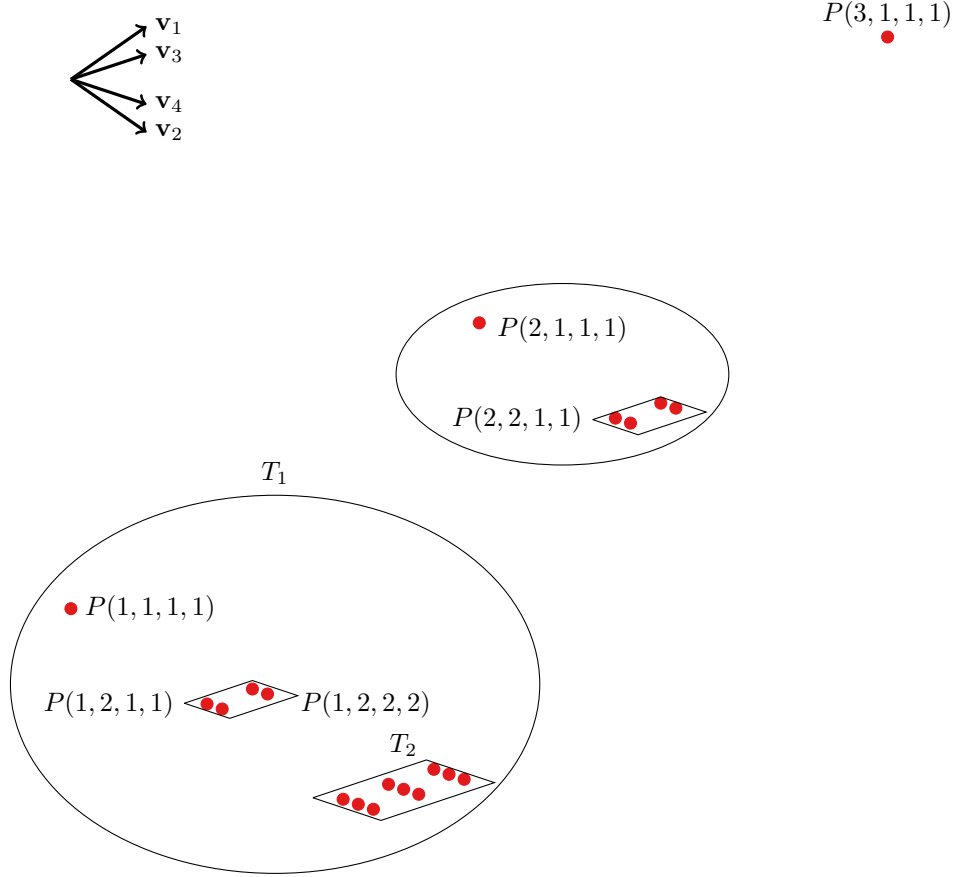
Let

$$S = \{\mathbf{i} \in [k]^4 : \mathbf{i}(1) + \mathbf{i}(2) \leq k + 1, \mathbf{i}(2) \geq \mathbf{i}(3), \text{ and } \mathbf{i}(2) \geq \mathbf{i}(4)\},$$

so that we have

$$|S| = \sum_{i=1}^k (k+1-i)^2 = \frac{k+1}{2} \binom{k+2}{3}.$$

An ordered triple of points  $(u, v, w) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^3$  is called a *hole* if  $u(1) < v(1) < w(1)$ , and either  $v(2) < \min\{u(2), w(2)\}$ , or  $v(3) < \min\{u(3), w(3)\}$ .



■ **Figure 4** An illustration of the points  $P(\mathbf{i})$  for  $\mathbf{i} \in S$ ,  $k = 3$ .

▷ **Claim 24.** Let  $H \subset S$ . If the set  $\{P(\mathbf{i}) : \mathbf{i} \in H\}$  does not contain a hole, then  $|H| \leq k$ .

*Proof.* Let  $S = S_k$ . We prove this claim by induction on  $k$ . If  $k = 1$ ,  $S$  contains one element, so there is nothing to prove.

Suppose that  $k \geq 2$ . Let  $T_1 = \{\mathbf{i} \in S : \mathbf{i}(1) = 1\}$  and  $H_1 = H \cap T_1$ . (See Figure 4 for an illustration.) We distinguish two cases.

*Case 1:*  $|H_1| \leq 1$ . Define

$$H' = \{(i_1 - 1, i_2, i_3, i_4) : (i_1, i_2, i_3, i_4) \in H \setminus H_1\}.$$

Then  $H' \subset S_{k-1}$  and  $H'$  does not contain a hole. Hence, we obtain  $|H'| \leq k - 1$ , by the induction hypothesis. On the other hand,  $|H'| \geq |H| - 1$ , which yields that  $|H| \leq k$ .

*Case 2:*  $|H_1| \geq 2$ . In this case, we must have  $H = H_1$ . Otherwise, choose  $\mathbf{i}, \mathbf{i}' \in H_1$ ,  $\mathbf{j} \in H \setminus H_1$ , and let  $u = P(\mathbf{i}), v = P(\mathbf{i}')$ , and  $w = P(\mathbf{j})$ . Assume without loss of generality that  $u(1) < v(1)$ . Then  $u(1) < v(1) < w(1)$ , and by the LEX property we have  $w(2) > \max\{u(2), v(2)\}$  and  $w(3) > \max\{u(3), v(3)\}$ . Therefore, if  $(u, v, w)$  is not a hole, then we must have  $u(2) < v(2) < w(2)$  and  $u(3) < v(3) < w(3)$ . However, this means that  $\text{sg}(v - u) = (1, 1, 1) = \mathbf{v}_1$ , which contradicts the LEX property, as  $\mathbf{i}(1) = \mathbf{i}'(1)$ .

Hence, we can assume that  $H = H_1 \subset T_1$ . Let  $T_2 = \{\mathbf{i} \in S : \mathbf{i}(1) = 1, \mathbf{i}(2) = k\} \subset T_1$  and  $H_2 = H \cap T_2$ . Again, we distinguish two subcases.

*Subcase 1:*  $|H_2| \leq 1$ . Define  $H' = H \setminus H_2$ . Then  $H' \subset S_{k-1}$  and  $H'$  does not contain a hole, which yields, by the induction hypothesis, that  $|H'| \leq k-1$ . On the other hand,  $|H'| \geq |H| - 1$ , so  $|H| \leq k$ .

*Subcase 2:*  $|H_2| \geq 2$ . In this case, we show that  $H = H_2$ . Otherwise, let  $\mathbf{i}, \mathbf{i}' \in H_2$ ,  $\mathbf{j} \in H \setminus H_2$ , and  $u = P(\mathbf{j}), v = P(\mathbf{i})$  and  $w = P(\mathbf{i}')$ . Assume without loss of generality that  $v(1) < w(1)$ . Then  $u(1) < v(1) < w(1)$ ,  $u(2) > \max\{v(2), w(2)\}$ , and  $u(3) > \max\{v(3), w(3)\}$ , by the LEX property. Thus,  $(u, v, w)$  is a hole, unless  $u(2) > v(2) > w(2)$  and  $u(3) > v(3) > w(3)$ , which would mean that the  $\text{sg}(w-v) = (1, -1, -1) = \mathbf{v}_2$ . However, this contradicts the LEX property, because  $\mathbf{i}(2) = \mathbf{i}'(2)$ . Hence, we can suppose that  $H = H_2 \subset T_2$ . Here,  $T_2$  is partitioned into  $k$  sets  $U_1, \dots, U_k$ , where  $U_l = \{(1, k, l, m) : m = 1, \dots, k\}$  for  $l = 1, \dots, k$ . Note that  $|U_l| = k$ . We show that  $H$  is either completely contained in one of the sets  $U_l$ , or  $H$  intersects each of  $U_1, \dots, U_k$  in at most one element. In either case, we get  $|H| \leq k$ . Suppose to the contrary that there exists  $l \neq l'$  and three elements  $\mathbf{i}, \mathbf{i}' \in U_l \cap H$ ,  $\mathbf{j} \in U_{l'} \cap H$ . Let  $u = P(\mathbf{i})$ ,  $v = P(\mathbf{i}')$ , and  $w = P(\mathbf{j})$ . Without loss of generality, assume that  $u(1) < v(1)$ . Now there are two cases depending on the order of  $l$  and  $l'$ . If  $l < l'$ , then by the LEX property  $u(1) < v(1) < w(1)$ ,  $v(2) < u(2) < w(2)$ , and  $w(3) < u(3) < v(3)$ , so  $(u, v, w)$  is a hole. If  $l' < l$ , then  $w(1) < u(1) < v(1)$ ,  $w(2) < v(2) < u(2)$ , and  $u(3) < v(3) < w(3)$ , so  $(w, u, v)$  is a hole.  $\triangleleft$

The rest of the proof of Theorem 23 is very similar to that of the proof of Theorem 14. We omit its proof here.

## 6 Concluding remarks

We proved that the best  $\chi$ -bounding function for the family of disjointness graphs of  $x$ -monotone curves satisfies  $f(k) = \Theta(k^4)$ . What can we say about the chromatic number of  $K_k$ -free families of disjointness graphs of segments or convex sets? For both of these families, the best known upper bound on the chromatic number is also  $O(k^4)$ , but there are no matching lower bounds.

---

### References

- 1 Pankaj K Agarwal and Nabil H Mustafa. Independent set of intersection graphs of convex objects in 2D. *Computational Geometry*, 34(2):83–95, 2006.
- 2 Edgar Asplund and Branko Grünbaum. On a coloring problem. *Mathematica Scandinavica*, 8(1):181–188, 1960.
- 3 Peter Brass, William OJ Moser, and János Pach. *Research problems in discrete geometry*. Springer Science & Business Media, 2006.
- 4 James Perkins Burling. On Coloring Problems of Families of Polytopes, 1965.
- 5 Sergio Cabello, Jean Cardinal, and Stefan Langerman. The clique problem in ray intersection graphs. *Discrete & computational geometry*, 50(3):771–783, 2013.
- 6 Márcia R Cerioli, Luerbio Faria, Talita O Ferreira, and Fábio Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18:73–79, 2004.
- 7 Robert P Dilworth. A decomposition theorem for partially ordered sets. *Annals of mathematics*, pages 161–166, 1950.
- 8 Adrian Dumitrescu and János Pach. Minimum clique partition in unit disk graphs. In *Graphs and Combinatorics*, volume 27(3), pages 399–411. Springer, 2011.

- 9 Stephan Eidenbenz and Christoph Stamm. Maximum clique and minimum clique partition in visibility graphs. In *IFIP International Conference on Theoretical Computer Science*, pages 200–212. Springer, 2000.
- 10 Jacob Fox and János Pach. Computing the independence number of intersection graphs. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, pages 1161–1165. Society for Industrial and Applied Mathematics, 2011.
- 11 Zoltán Füredi. The maximum number of unit distances in a convex  $n$ -gon. *J. Comb. Theory, Ser. A*, 55(2):316–320, 1990.
- 12 Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.
- 13 Fanika Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.
- 14 András Gyárfás. On the chromatic number of multiple interval graphs and overlap graphs. *Discrete mathematics*, 55(2):161–166, 1985.
- 15 András Gyárfás. Problems from the world surrounding perfect graphs. *Applicationes Mathematicae*, 19(3-4):413–441, 1987.
- 16 J Mark Keil and Lorna Stewart. Approximating the minimum clique cover and other hard problems in subtree filament graphs. *Discrete Applied Mathematics*, 154(14):1983–1995, 2006.
- 17 Chaya Keller, Shakhar Smorodinsky, and Gábor Tardos. On Max-Clique for intersection graphs of sets and the Hadwiger-Debrunner numbers. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2254–2263. SIAM, 2017.
- 18 Dániel Korándi and István Tomon. Improved Ramsey-type results in comparability graphs. *arXiv preprint*, 2018. [arXiv:1810.00588](https://arxiv.org/abs/1810.00588).
- 19 Alexandr Kostochka. Coloring intersection graphs of geometric figures with a given clique number. *Contemporary mathematics*, 342:127–138, 2004.
- 20 Alexandr Kostochka and Jan Kratochvíl. Covering and coloring polygon-circle graphs. *Discrete Mathematics*, 163(1-3):299–305, 1997.
- 21 Alexandr Kostochka and Jaroslav Nešetřil. Chromatic number of geometric intersection graphs. In *1995 Prague Midsummer Combinatorial Workshop*, volume 95(309), pages 43–45. KAM Series, 1995.
- 22 Alexandr V Kostochka. Upper bounds on the chromatic number of graphs. *Trudy Inst. Mat*, 10:204–226, 1988.
- 23 Jan Kratochvíl and Jirí Matousek. Intersection graphs of segments. *Journal of Combinatorial Theory, Series B*, 62(2):289–315, 1994.
- 24 Jan Kratochvíl and Jaroslav Nešetřil. Independent set and clique problems in intersection-defined classes of graphs. *Commentationes Mathematicae Universitatis Carolinae*, 31(1):85–93, 1990.
- 25 Jan Kynčl. Ramsey-type constructions for arrangements of segments. *European Journal of Combinatorics*, 33(3):336–339, 2012.
- 26 David Larman, Jirí Matoušek, János Pach, and Jenő Töröcsik. A Ramsey-Type Result for Convex Sets. *Bulletin of the London Mathematical Society*, 26(2):132–136, 1994.
- 27 Michał Lasoń, Piotr Micek, Arkadiusz Pawlik, and Bartosz Walczak. Coloring intersection graphs of arc-connected sets in the plane. *Discrete & Computational Geometry*, 52(2):399–415, 2014.
- 28 Sean McGuinness. Colouring arcwise connected sets in the plane I. *Graphs and Combinatorics*, 16(4):429–439, 2000.
- 29 Leon Mirsky. A dual of Dilworth’s decomposition theorem. *The American Mathematical Monthly*, 78(8):876–877, 1971.
- 30 Torsten Mütze, Bartosz Walczak, and Veit Wiechert. Realization of shift graphs as disjointness graphs of 1-intersecting curves in the plane. *arXiv preprint*, 2018. [arXiv:1802.09969](https://arxiv.org/abs/1802.09969).
- 31 János Pach and Gábor Tardos. Forbidden paths and cycles in ordered graphs and matrices. *Israel Journal of Mathematics*, 155(1):359–380, 2006.

- 32 János Pach, Gábor Tardos, and Géza Tóth. Disjointness graphs of segments. *arXiv preprint*, 2017. [arXiv:1704.01892](https://arxiv.org/abs/1704.01892).
- 33 János Pach and Jenő Törőcsik. Some geometric applications of Dilworth's theorem. *Discrete & Computational Geometry*, 12(1):1–7, 1994.
- 34 Arkadiusz Pawlik, Jakub Kozik, Tomasz Krawczyk, Michał Lasoń, Piotr Micek, William T Trotter, and Bartosz Walczak. Triangle-free intersection graphs of line segments with large chromatic number. *Journal of Combinatorial Theory, Series B*, 105:6–10, 2014.
- 35 Alexandre Rok and Bartosz Walczak. Outerstring graphs are  $\chi$ -bounded. In *Symposium on Computational Geometry*, page 136, 2014.
- 36 Alexandre Rok and Bartosz Walczak. Coloring curves that cross a fixed curve. *Discrete & Computational Geometry*, pages 1–22, 2017.
- 37 Andrew Suk. Coloring intersection graphs of  $x$ -monotone curves in the plane. *Combinatorica*, 34(4):487–505, 2014.
- 38 Kenneth Jay Supowit. TOPICS IN COMPUTATIONAL GEOMETRY. *UIUCDCS-R*, 1982.





# Computing Persistent Homology of Flag Complexes via Strong Collapses

Jean-Daniel Boissonnat

Université Côte d’Azur, INRIA, Sophia Antipolis, France

Jean-Daniel.Boissonnat@inria.fr

Siddharth Pritam

Université Côte d’Azur, INRIA, Sophia Antipolis, France

siddharth.pritam@inria.fr

---

## Abstract

In this article, we focus on the problem of computing Persistent Homology of a flag tower, i.e. a sequence of flag complexes connected by simplicial maps. We show that if we restrict the class of simplicial complexes to flag complexes, we can achieve decisive improvement in terms of time and space complexities with respect to previous work. We show that strong collapses of flag complexes can be computed in time  $O(k^2v^2)$  where  $v$  is the number of vertices of the complex and  $k$  is the maximal degree of its graph. Moreover we can strong collapse a flag complex knowing only its 1-skeleton and the resulting complex is also a flag complex. When we strong collapse the complexes in a flag tower, we obtain a reduced sequence that is also a flag tower we call the core flag tower. We then convert the core flag tower to an equivalent filtration to compute its PH. Here again, we only use the 1-skeletons of the complexes. The resulting method is simple and extremely efficient.

**2012 ACM Subject Classification** Mathematics of computing; Theory of computation → Computational geometry; Mathematics of computing → Algebraic topology

**Keywords and phrases** Computational Topology, Topological Data Analysis, Strong Collapse, Persistent homology

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.55

**Funding** This research has received funding from the European Research Council (ERC) under the European Union’s Seventh Framework Programme (FP/2007- 2013) / ERC Grant Agreement No. 339025 GUDHI (Algorithmic Foundations of Geometry Understanding in Higher Dimensions).

## 1 Introduction

This paper is a continuation of the research reported in [8] on the usage of strong collapses to accelerate the computation of the Persistent Homology (PH) of a sequence of simplicial complexes. Computing PH efficiently is a very well studied problem in Computational Topology and Topological Data Analysis. The time complexity to compute persistent homology is  $O(n^\omega)$ , where  $n$  is the total number of simplices and  $\omega \leq 2.4$  is the matrix multiplication exponent [38, 31]. In practice, when we encounter massive and high-dimensional datasets,  $n$  may be very large  $n$  (of order of billions) and computing PH is then slow and memory intensive. This especially occurs for flag complexes since the simplices of a flag complex are the cliques of the 1-skeleton of the complex.

There has been a lot of progress in the recent years along two directions. On one hand, efficient implementations and optimizations have led to a new generation of software for PH computation [32, 6, 4, 40]. Another complementary direction has been explored to reduce the size of the complexes in the sequence while preserving (or approximating in a controlled way) the persistent homology of the sequence [39, 26, 14, 9, 45, 36, 16, 23]. Most of the methods are for general simplicial complexes except [4, 45] which focus on the Vietoris-Rips complex, an example of a flag complex. In this paper, we build on the initial success of



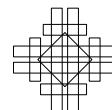
© Jean-Daniel Boissonnat and Siddharth Pritam;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 55; pp. 55:1–55:15

Leibniz International Proceedings in Informatics



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



[8] and show that further decisive progress can be obtained if one restricts the family of simplicial complexes to flag complexes. Flag complexes are fully characterized by their graph (or 1-skeleton), the other faces being obtained by computing the cliques of the graph. Hence, a flag complex can be represented by its 1-skeleton, which is a very compact representation. Flag complexes are very popular and, in particular, Vietoris-Rips complexes are widely used in Topological Data Analysis.

It has been shown in [8] that the persistent homology of a sequence of simplicial complexes can be computed very efficiently using *strong collapses*. The basic idea is to simplify the complexes of the input sequence by using strong collapses, as introduced by J. Barmak and E. Miniam [3], and to compute the PH of an induced sequence of reduced simplicial complexes that has the same PH as the initial one. This reduced sequence is unique up to isomorphism and is called the *core sequence*. A crucial advantage of the method is that it only needs to store the *maximal simplices* of the complex, not the full set of the simplices of all dimensions, which saves space and time by a factor that can be exponential in the dimension of the complex. Still, in the case of a flag complex and, in particular, in the case of the widely used Vietoris-Rips complex, the maximal simplices are the maximal cliques of the 1-skeleton of the complex and their number can be very large (exponential in the dimension of the complex). As a result, the method of [8] devoted most of the time to compute the maximal faces of the complexes prior to their strong collapse.

In this paper, we avoid computing maximal cliques and show that we can strong collapse any flag complex using only the *1-skeleton* or graph of the complex. Another crucial observation is that the reduced complex obtained by strong collapsing a flag complex is itself a flag complex.

Furthermore, if we consider a sequence of flag complexes connected by simplicial maps (a flag tower), we can strong collapse all the complexes of the sequence. The obtained core sequence is also a flag tower with smaller complexes that are connected by simplicial maps that are induced from the maps of the original sequence and the strong collapses. In the general case where the core sequence is not a filtration (which usually happens even if the original sequence is a filtration), we need to convert the flag tower to an equivalent filtration to compute its PH using known algorithms. To do so, we build on the work of [22, 35] that we restrict to flag complexes and strong collapses. This allows us to convert a flag tower to an equivalent flag filtration using again only the 1-skeleton.

The major advantages of our approach are:

- We can compute PH for large complexes of high dimensions as we don't need to compute the maximal simplices.
- The dimension of the original sequence is in fact irrelevant and what matters is the dimension of the core sequence, which is usually quite small.
- Instead of computing the exact PH, we can compute an approximate PH which is substantially faster at a very minimal cost. We will explain more about the approximation scheme in Sections 2 and 5.

The resulting method is simple and extremely efficient. On the theory side, we show that strong collapses can be computed in time  $O(k^2v^2)$  where  $v$  is the number of vertices of the complex and  $k$  the maximal degree of its graph. The algorithm described in this paper has been implemented. Numerous experiments show that the computation of the persistent homology of flag complexes can be obtained much faster than with previous methods, e.g. Ripser [4]. The code will be soon released in the Gudhi library [32].

## 2 Preliminaries

In this section, we provide a brief review of the notions of simplicial complex and strong collapse as introduced in [3] and recalled in [8]. We use the same notations and conventions as in [8] and just recall, in addition, some basic facts about *Flag complexes*. Readers can refer to [33] for a comprehensive introduction to these topics.

**Simplex, simplicial complex and simplicial map.** An abstract simplicial complex  $K$  is a collection of subsets of a non-empty finite set  $X$ , such that for every subset  $A$  in  $K$ , all the subsets of  $A$  are in  $K$ . From now on we will call an *abstract simplicial complex* simply a *simplicial complex* or just a *complex*. An element of  $K$  is called a **simplex**. An element of cardinality  $k + 1$  is called a  $k$ -simplex and  $k$  is called its **dimension**. A simplex is called **maximal** if it is not a proper subset of any other simplex in  $K$ . A sub-collection  $L$  of  $K$  is called a **subcomplex**, if it is a simplicial complex itself.

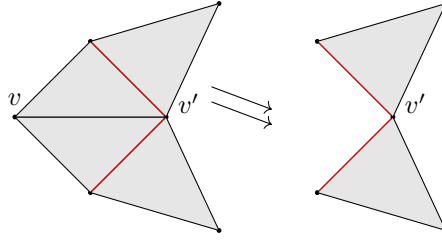
A map  $\psi : K \rightarrow L$  between two simplicial complexes is called a **simplicial map**, if it always maps a simplex in  $K$  to a simplex in  $L$ . Simplicial maps are induced by vertex-to-vertex maps. In particular, there is a finite number of simplicial maps between two given finite simplicial complexes. Simplicial maps induce continuous maps between the underlying geometric realisations of the simplicial complexes. Any general simplicial map can be decomposed into more elementary simplicial maps, namely **elementary inclusions** (i.e., inclusions of a single simplex) and **elementary contractions**  $\{\{u, v\} \mapsto u\}$  (where a vertex is mapped onto another vertex). Two simplicial maps  $\phi : K \rightarrow L$  and  $\psi : K \rightarrow L$  are **contiguous** if, for all  $\sigma \in K$ ,  $\phi(\sigma) \cup \psi(\sigma) \in L$ . Two contiguous maps are known to be homotopic [41, Theorem 12.5].

**Flag complex.** A complex  $K$  is a flag or a clique complex if, when a subset of its vertices has pairwise edges between them, they span a simplex. It follows that the full structure of  $K$  is determined by its 1-skeleton we denote by  $G$ . For a vertex  $v$  in  $G$ , the **open neighborhood**  $N_G(v)$  of  $v$  in  $G$  is defined as  $N_G(v) := \{u \in G \mid [uv] \in E\}$ . The **closed neighborhood**  $N_G[v]$  is  $N_G[v] := N_G(v) \cup \{v\}$ . We further define the relative closed neighborhood of  $u$  by  $v$  in  $G$  as the set of vertices in  $N_G[u]$  that are not in  $N_G[v]$ . We denote it by  $N_G[u \setminus v]$ .

**Dominated vertex.** Let  $\sigma$  be a simplex of a simplicial complex  $K$ , the **closed star** of  $\sigma$  in  $K$ ,  $st_K(\sigma)$  is a subcomplex of  $K$  which is defined as follows,  $st_K(\sigma) := \{\tau \in K \mid \tau \cup \sigma \in K\}$ . The **link** of  $\sigma$  in  $K$ ,  $lk_K(\sigma)$  is defined as the set of simplices in  $st_K(\sigma)$  which do not intersect with  $\sigma$ ,  $lk_K(\sigma) := \{\tau \in st_K(\sigma) \mid \tau \cap \sigma = \emptyset\}$ .

Taking a join with a vertex transforms a simplicial complex into a **simplicial cone**. Formally if  $L$  is a simplicial complex and  $a$  is a vertex not in  $L$  then the simplicial cone  $aL$  is defined as  $aL := \{a, \tau \mid \tau \in L \text{ or } \tau = \sigma \cup a; \text{ where } \sigma \in L\}$ . A vertex  $v$  in  $K$  is called a **dominated vertex** if the link of  $v$  in  $K$ ,  $lk_K(v)$  is a simplicial cone, that is, there exists a vertex  $v' \neq v$  and a subcomplex  $L$  in  $K$ , such that  $lk_K(v) = v'L$ . We say that the vertex  $v'$  is dominating  $v$  and  $v$  is dominated by  $v'$ . The symbol  $K \setminus v$  (deletion of  $v$  from  $K$ ) refers to the subcomplex of  $K$  which has all simplices of  $K$  except the ones containing  $v$ . Below is an important remark from [3, Remark 2.2], which proposes an alternative definition of dominated vertices.

► **Remark 1.** A vertex  $v \in K$  is dominated by another vertex  $v' \in K$ , if and only if all the maximal simplices of  $K$  that contain  $v$  also contain  $v'$  [3].



■ **Figure 1** Illustration of an *elementary strong collapse*. In the complex on the left,  $v$  is dominated by  $v'$ . The link of  $v$  is highlighted in red. Removing  $v$  leads to the complex on the right.

**Strong collapse.** An **elementary strong collapse** is the deletion of a dominated vertex  $v$  from  $K$ , which we denote with  $K \searrow\searrow K \setminus v$ . Figure 1 illustrates an easy case of an elementary strong collapse. There is a **strong collapse** from a simplicial complex  $K$  to its subcomplex  $L$ , if there exists a series of elementary strong collapses from  $K$  to  $L$ , denoted as  $K \searrow\searrow L$ . The inverse of a strong collapse is called a **strong expansion**. If there exists a combination of strong collapses and/or strong expansions from  $K$  to  $L$ , then  $K$  and  $L$  are said to have the same **strong homotopy type**.

The notion of strong homotopy type is stronger than the notion of simple homotopy type in the sense that if  $K$  and  $L$  have the same strong homotopy type, then they have the same simple homotopy type, and therefore the same homotopy type [3]. There are examples of contractible or simply collapsible simplicial complexes that are not strong collapsible.

A complex without any dominated vertex will be called a **minimal complex**. A **core** of a complex  $K$  is a minimal subcomplex  $K^c \subseteq K$ , such that  $K \searrow\searrow K^c$ . *Every simplicial complex has a unique core up to isomorphism. The core decides the strong homotopy type of the complex, and two simplicial complexes have the same strong homotopy type if and only if they have isomorphic cores* [3, Theorem 2.11].

**Retraction map.** If a vertex  $v \in K$  is dominated by another vertex  $v' \in K$ , the vertex map  $r : K \rightarrow K \setminus v$  defined as:  $r(w) = w$  if  $w \neq v$  and  $r(v) = v'$ , induces a simplicial map that is a *retraction map*. The homotopy between  $r$  and the identity  $i_{K \setminus v}$  over  $K \setminus v$  is in fact a strong deformation retract. Furthermore, the composition  $(i_{K \setminus v})r$  is contiguous to the identity  $i_K$  over  $K$  [3, Proposition 2.9].

**Sequences of complexes.** A **sequence** of simplicial complexes  $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} K_3 \xrightarrow{f_3} \dots \xrightarrow{f_{(m-1)}} K_m\}$ , connected through simplicial maps  $f_i$  is called a **simplicial tower** or simply a *tower*. We call a tower a **flag tower** if all the simplicial complexes  $K_i$  are flag complexes. When all the simplicial maps  $f_i$ s are inclusions, then the tower is called a *filtration* and a flag tower is called a **flag filtration**.

**Persistent homology.** If we compute the homology classes of all the  $K_i$ , we get the sequence  $\mathcal{P}(\mathcal{T}) : \{H_p(K_1) \xrightarrow{f_1^*} H_p(K_2) \xrightarrow{f_2^*} H_p(K_3) \xrightarrow{f_3^*} \dots \xrightarrow{f_{(m-1)}^*} H_p(K_m)\}$ . Here  $H_p()$  denotes the homology class of dimension  $p$  with coefficients from a field  $\mathbb{F}$  and  $*$  denotes an induced homomorphism.  $\mathcal{P}(\mathcal{T})$  is a sequence of vector spaces connected through homomorphisms, called a **persistence module**. More formally, a *persistence module*  $\mathbb{V}$  is a sequence of vector spaces  $\{V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow \dots \rightarrow V_m\}$  connected with homomorphisms  $\{\rightarrow\}$  between them. A persistence module arising from a sequence of simplicial complexes captures the evolution of the topology of the sequence.

Any persistence module can be *decomposed* into a collection of intervals of the form  $[i, j)$  [10]. The multiset of all the intervals  $[i, j)$  in this decomposition is called the **persistence diagram** of the persistence module. An interval of the form  $[i, j)$  in the persistence diagram of  $\mathcal{P}(\mathcal{T})$  corresponds to a homological feature (a ‘cycle’) which appeared at  $i$  and disappeared at  $j$ . The persistence diagram (PD) completely characterizes the persistence module, that is, there is a bijective correspondence between the PD and the equivalence class of the persistence module [10, 51].

Two different persistence modules  $\mathbb{V} : \{V_1 \rightarrow V_2 \rightarrow \dots \rightarrow V_m\}$  and  $\mathbb{W} : \{W_1 \rightarrow W_2 \rightarrow \dots \rightarrow W_m\}$ , connected through a set of homomorphisms  $\phi_i : V_i \rightarrow W_i$  are **equivalent** if the  $\phi_i$  are isomorphisms and the following diagram commutes [10, 20].

$$\begin{array}{ccccccc}
 V_1 & \longrightarrow & V_2 & & \dots & & \longrightarrow & V_{m-1} & \longrightarrow & V_m \\
 \downarrow \phi_1 & & \downarrow \phi_2 & & & & & \downarrow \phi_{m-1} & & \downarrow \phi_m \\
 W_1 & \longrightarrow & W_2 & & \dots & & \longrightarrow & W_{m-1} & \longrightarrow & W_m
 \end{array}$$

The equivalent persistence modules will have the same interval decomposition, therefore the same diagram.

**Overview of the algorithm and approximation scheme.** The algorithm presented in this paper adopts the same strategy as the algorithm reported in [8]. However we just focus on *flag towers* instead of general sequences.

Let  $\mathcal{T} : \{K_1 \xrightarrow{f_1} K_2 \xrightarrow{f_2} \dots \xrightarrow{f_{(m-1)}} K_m\}$  be a flag tower of which we want to compute the persistence diagram. We first strong collapse the various complexes  $K_i, i = 1, \dots, m$  as suggested in [8]. Since each  $K_i$  is a flag complex, the computation of its core can be computed much more efficiently using only its 1-skeleton. The new algorithm is discussed in detail in Section 3.

We then compute a *core tower* that connects the  $K_i$ s through induced simplicial maps. This is again an adaptation of what has been done in [8] for general simplicial complexes to the case of flag complexes. Lastly, we compute a flag *filtration* with the same PH as the core tower. This is similar to what has been done in [22, 35], and is detailed in Section 4. This filtration can then be sent to any algorithm that computes the persistence homology of a flag filtration [40, 4, 6, 32].

It is important to note that the algorithm computes the exact PH of the input flag tower  $\mathcal{T}$ . However, instead of considering all the  $K_i$ s and the associated simplicial maps, we can select some of the  $K_i$ s we rename  $K'_1, \dots, K'_q$ , and compose the original maps  $f_1, \dots, f_{m-1}$  to obtain simplicial maps  $f'_1, \dots, f'_{q-1}$  connecting the selected complexes. We thus obtain a sub-tower  $\mathcal{T}'$  and the algorithm will then compute the exact PH of  $\mathcal{T}'$ . An application of this idea will be presented for Vietoris-Rips filtrations in Section 5. By rounding the values of the threshold parameter to a given number of snapshots of the threshold value, we will be able to approximate in a controlled way the PH of the initial filtration  $\mathcal{T}$ .

### 3 Strong Collapse of a Flag complex

In this section, we show that the core of a flag complex  $K$  is itself a flag complex whose graph is called the *core graph* of  $K$ . The core graph of  $K$  can be computed from the 1-skeleton  $G$  of  $K$  in time  $\mathcal{O}(v^2k^2)$ , where  $v$  is the number of vertices in  $K$  and  $k$  is an upper bound on the degree of  $G$  (i.e. the number of edges that are incident on a vertex in  $K$ ).

Although this change wrt the algorithm in [8] might look minor, it is crucial in practice as the time to compute all the maximal simplices of a flag complex from its graph is exponential in the number of its vertices. We thus reduce immensely the time and space complexity of the general algorithm of [8] whose complexity is  $\mathcal{O}(v^2\Gamma_0d + m^2\Gamma_0d)$ , where  $\Gamma_0$  is an upper bound on the number of *maximal simplices* incident to a vertex.

In the following lemma, we describe a condition in terms of the closed neighborhood  $N_G[v]$  of a vertex  $v$  of a flag complex  $K$  under which  $v$  will be dominated by another vertex  $v'$  of  $K$ . This result has been studied in another context in [30, Lemma 4.1].

► **Lemma 2.** *Let  $K$  be a flag complex. A vertex  $v \in K$  is dominated by  $v'$  iff  $N_G[v] \subseteq N_G[v']$ .*

**Proof.** If  $v$  is dominated by  $v'$ , then, according to Remark 1, the set of maximal simplices that contain  $v$  is a subset of the set of maximal simplices that contain  $v'$ . It follows that  $N_G[v] \subseteq N_G[v']$ .

Now we prove the other direction. Let  $\sigma$  be a maximal simplex of  $K$  containing  $v$ . Any other vertex  $x$  of  $\sigma$  is joined to  $v$  by an edge  $[x, v] \in \sigma$ . Moreover, since  $N_G[v] \subseteq N_G[v']$ ,  $[v, v']$  and  $[x, v']$  are in  $K$ . It follows that every vertex in  $\sigma$  has an edge with both  $v$  and  $v'$  and, since  $K$  is a flag complex and  $\sigma$  is maximal,  $v'$  must be in  $\sigma$ . This implies that all the maximal simplices that contain  $v$  also contain  $v'$ . Hence  $v$  is dominated by  $v'$ . ◀

As mentioned before, an elementary strong collapse consists in removing a dominated vertex, and it can be easily observed that removing a vertex does not affect the ‘flagness’ of the residual complex  $K \setminus v$ . In other words, if  $\sigma$  is a maximal clique with vertex  $v$ , the resultant clique  $\sigma \setminus v$  is still a maximal clique in  $K \setminus v$ . Moreover, all the other cliques that do not contain  $v$  still span the complete simplices. This implies that the core  $K^c$  of a flag complex  $K$  with graph  $G$  is a flag complex of a sub-graph  $G^c$  of  $G$ .

In what follows next, we describe an algorithm to compute the core graph  $G^c \subseteq G$  whose flag complex is the core  $K^c$  of  $K$ .

**Data structure.** We represent  $G$  with its adjacency matrix  $M$ , where the rows and the columns of  $M$  represent the vertices of  $G$ . An entry  $M[v_i][v_j]$  associated with vertices  $v_i$  and  $v_j$  is set to 1 if either the edge  $[v_i, v_j] \in G$  or  $i = j$ , and to 0 otherwise. Note that we set  $M[v_i][v_j] = 1$  for  $i = j$  to be able to consider closed neighborhood. We will say that a row  $v$  is contained in another row  $v'$  if the set of column indices of the non-zero entries of  $v$  is a subset of the indices of the non-zero entries of  $v'$ . It is clear that if a row  $v$  is contained in another row  $v'$ , we have  $N_G[v] \subseteq N_G[v']$  and therefore the vertex  $v$  is dominated by the vertex  $v'$ .

**Core graph algorithm.** Given the adjacency matrix  $M$  of  $G$ , we compute the adjacency matrix  $C$  of the core graph  $G^c$ . In view of Lemma 2, we can easily compute  $C$  from  $M$  using basic row removal operations. Loosely speaking, we remove the rows of  $M$  that are contained in another row. After removing the row associated to  $v$ , we simultaneously update the matrix by removing the column associated to  $v$ . The process is iterated as long as the matrix can be reduced. Upon termination, we output the reduced matrix  $C$ , which is the adjacency matrix of the core graph  $G^c$  of  $K$ . Since the core of a complex is always unique, the order in which vertices are removed does not matter [3].

**Retraction map computation.** We can easily compute the retraction map  $r$  defined in Section 2 using the above core graph algorithm. A row  $v$  being removed in  $M$  corresponds to a dominated vertex in  $K$  and the row which contains  $v$  corresponds to a dominating vertex. Therefore we map the dominated vertex to the dominating vertex.



**Domination tests optimization.** Let us observe that, to check if a row  $v$  is dominated by some other row  $v'$ , it is sufficient to compare  $v$  with its neighbors, which are at most  $k$  in number, if  $k$  denotes the maximum degree of the vertices in  $G$ .

We define a row  $v$  to be a **candidate row** for the next iteration if at least one of its neighbors has been removed in a previous row removal iteration. We observe that the candidate rows are the only rows that need to be considered in the domination tests of the algorithm. Indeed, a row  $w$  of  $M$  whose set of neighbors has not been modified at the previous *iteration* cannot be dominated by another row  $v'$  of  $M$ , as  $w$  was not dominated in the previous iteration and all other vertices can only lose neighbors. This ensures that  $w$  will still remain un-dominated.

We maintain a *queue*, for the candidate rows (rowQueue) which is implemented as a First in First out (FIFO) queue. At each iteration, we *pop out* a candidate row from rowQueue for domination test. After each successful domination test, we push the new candidate rows in the queue in preparation for the subsequent iteration. In the first iteration, we push all the rows in rowQueue. Algorithm 1 gives the pseudo code of our algorithm.

---

**Algorithm 1** Core graph algorithm.

---

```

1: procedure CORE( $M$ )
2:   input : the adjacency matrix  $M$  of the graph of a flag complex  $K$ 
3:    $rowQueue \leftarrow$  push all rows of  $M$  (all vertices of  $K$ )
4:   while rowQueue is not empty do
5:      $v \leftarrow pop(rowQueue)$ 
6:      $N_G[v] \leftarrow$  the non-zero columns of  $v$ 
7:     for  $w$  in  $N_G[v]$  do
8:       if  $N_G[v] \subseteq N_G[w]$  then
9:         Remove from  $M$  the column and the row associated to  $v$ 
10:        push all the entries of  $N_G(v)$  to  $rowQueue$  if not pushed before
11:        break
12:      end if
13:    end for
14:  end while
15:  return  $M$  ▷  $M$  is now the adjacency matrix of the core of  $K$ 
16: end procedure

```

---

**Time Complexity.** Let us start by analyzing the most basic operation in our algorithm which is to determine if a row is dominated by another row. We store the rows of the matrix as sorted lists. Deciding if a sorted list is included in another sorted list (Line 8) can be done in time  $\mathcal{O}(l)$ , where  $l$  is the size of the longer list. In our case, the length of a row list is at most  $k + 1$  where  $k$  denotes as before the maximal degree of any vertex. Hence lines 8-12 takes  $\mathcal{O}(k)$  time.

As explained in the paragraph *Domination tests optimization*, each row is checked against at most  $k$  other rows. Hence the for loop (Lines 7-13) is executed at most  $k$  times. Moreover, since at each iteration we ought to remove at least one row, the total number of iterations on the rows, i.e. the number of times the while loop is executed, is at most  $\mathcal{O}(v^2)$ , where  $v$  is the total number of vertices of the complex  $K$ . It follows that the worst-case time complexity of our algorithm is  $\mathcal{O}(v^2k^2)$ .

## 4 From a Flag Tower to a Flag Filtration

In this section, we show that, thanks to the notion of strong collapses, we can efficiently turn a flag *tower* into a flag *filtration* using only edge inclusions over the 1-skeletons of the complexes.

### 4.1 Previous work

It is known that any general simplicial map can be decomposed into elementary inclusions and elementary contractions. Hence if we can replace an elementary contraction  $\{\{u, v\} \mapsto u\}$  by an equivalent (not necessarily elementary) inclusion, we transform a tower into an equivalent filtration. This was the philosophy introduced by Dey et al. [22]. This idea has been further refined by Kerber and Schreiber [35] who introduced a slightly different approach based on coning. They provided theoretical bounds on the size and time to construct the final equivalent filtration. Specifically, they proved that the size of the equivalent filtration is  $\mathcal{O}(d * n * \log n_0)$ , where  $d$  is the maximal dimension of the complexes in the input tower and  $n$  (resp.  $n_0$ ) the total number of elementary inclusions (resp. vertex inclusions) in the input tower [35, Theorem 2].

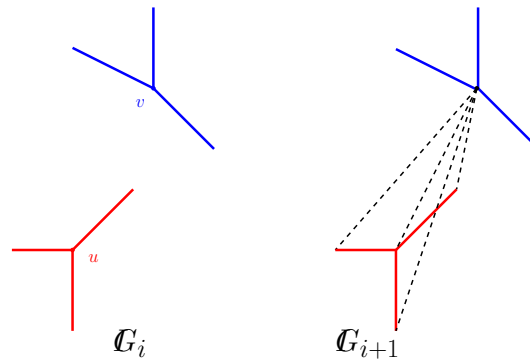
### 4.2 A new construction

We now present an algorithm that turns a flag tower into a flag filtration with the same PH. Our work builds upon the above mentioned previous works [22, 35]. The difference is that we use strong expansion which is the inverse operation of a strong collapse. The main advantage of strong expansions is that, when the input is a flag tower, we can use the domination criterion of Lemma 2. This leads to a simple algorithm that only deals with edges. The output filtration is a flag filtration, which can be represented very compactly. Moreover, since a strong expansion is a coning, we will be able to use the theoretical results of [35]. Now we describe our construction.

Let  $K_i$  be a flag complex and  $G_i$  be its 1-skeleton. We associate to  $K_i$  an augmented complex  $\mathbb{K}_i \supseteq K_i$ . As will be seen below,  $\mathbb{K}_i$  is also a flag complex whose 1-skeleton will be denoted by  $\mathbb{G}_i$ . Following the terminology of [35], we call a vertex  $v \in \mathbb{K}_i$  to be **active** if it is currently *not dominated*. The active closed neighborhood  $ActN_{G_i}[v]$  is then defined as the set of all active vertices in  $N_{G_i}[v]$ . Similarly,  $ActN_{G_i}[v \setminus u]$  denotes the set of active vertices in the closed neighborhood  $N_{G_i}[v]$  of  $v$  that are not in  $N_{G_i}[u]$ . Finally, let  $\{[u, ActN_{G_i}[v \setminus u]]\}$  denote the set of edges between  $u$  and  $ActN_{G_i}[v \setminus u]$ .

Using the notions defined above, we now explain how to inductively construct a filtration associated to a given flag tower. The construction operates in a streaming fashion on the 1-skeleton. We distinguish two kinds of inputs: elementary inclusions (of a vertex or an edge) and elementary contractions. For  $i = 0$ , we set  $\mathbb{G}_0 = \emptyset$ . We then define  $\mathbb{G}_i$  as follows.

1. if  $G_i \xrightarrow{\cup \sigma} G_{i+1}$  is an elementary *inclusion* where  $\sigma$  is either a vertex or an edge, we set  $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \sigma$ .
2. if  $G_i \xrightarrow{\{u, v\} \mapsto u} G_{i+1}$  is an elementary *contraction*
  - 2.1. if  $|ActN_{G_i}[v \setminus u]| \leq |ActN_{G_i}[u \setminus v]|$ , we set  $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[u, ActN_{G_i}[v \setminus u]]\}$  and  $v$  as contracted
  - 2.2. otherwise, we set  $\mathbb{G}_{i+1} := \mathbb{G}_i \cup \{[v, ActN_{G_i}[u \setminus v]]\}$  and  $u$  as contracted.



Note that  $G_i \subseteq G_{i+1}$  and thus  $K_i \subseteq K_{i+1}$ . We continue the construction until the end of our input tower.

**Complexity Analysis.** The vertices marked *contracted* during our construction are exactly the same as the inactive vertices defined in [35]. By construction, any contracted vertex will be dominated permanently in the filtration. Since such a vertex stops existing in the tower later on, its neighborhood stays the same and the vertex remains dominated. Therefore, at any point in our construction, the number of active vertices is less than the number of active vertices that are used in [35]. Moreover, since a strong expansion is a coning, the size of the final filtration in our construction is at most that obtained by the construction prescribed in [35]. Moreover, since we are working with 1-skeletons only, the space and time complexity of our method is much lower than that of [35].

Let us now analyze the time complexity of the algorithm. Notice that there are two basic operations on the 1-skeleton, elementary inclusions in Line 1, and the computation and comparison of  $ActN_{G_i}[v \setminus u]$  and  $ActN_{G_i}[u \setminus v]$  in Lines 2.1 and 2.2. Elementary inclusions can be performed in constant time  $\mathcal{O}(1)$ . To compute  $ActN_{G_i}[v \setminus u]$  we need to compute  $N_{G_i}[v]$ ,  $N_{G_i}[u]$  and the subset of vertices that are dominated (inactive) in  $N_{G_i}[v \setminus u]$ . We can access  $N_{G_i}[v]$  and  $N_{G_i}[u]$  in constant time  $\mathcal{O}(1)$  and compute the set-difference  $N_{G_i}[v \setminus u]$  in  $\mathcal{O}(k \log k)$  time, where  $k$  is the maximum degree of a vertex. Finally computing the dominated vertices in  $N_{G_i}[v \setminus u]$  can be done in  $\mathcal{O}(k^3)$  time as  $|N_{G_i}[v \setminus u]| \leq k$ . Therefore computing  $ActN_{G_i}[v \setminus u]$  and  $ActN_{G_i}[u \setminus v]$  takes  $\mathcal{O}(k^3)$  time. Since the size of active relative closed neighborhoods are bounded by  $k$ , for each elementary contraction, we include at most  $k$  edges in  $\mathcal{O}(k)$  time. It follows that the worst-case time complexity for each elementary contraction is  $\mathcal{O}(k^3)$ .

We conclude that the time complexity of the algorithm is  $\mathcal{O}(|G_m| + n_c * k^3)$  where  $n_c$  is the number of elementary contractions in the input tower and  $|G_m|$  is the size of the 1-skeleton of the output flag filtration. The space complexity of our construction is  $\mathcal{O}(n_0 * k)$  which is the size of a sparse adjacency matrix of a flag complex with  $n_0$  vertices.

**Correctness.** Finally we prove few lemmas and state our main result Theorem 7 to certify the correctness of the output of our construction.

► **Lemma 3.** Let  $f_i : K_i \xrightarrow{\{u,v\} \mapsto u} K_{i+1}$  be the first elementary contraction in the tower  $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$ . Then the complex  $K_{i+1}$  is a subcomplex of  $K_{i+1}$  and  $K_{i+1} \searrow \searrow K_{i+1}$ .

**Proof.** We prove the second part  $K_{i+1} \searrow \searrow K_{i+1}$  of the statement which then implies the first part  $K_{i+1} \subset K_{i+1}$ . Since  $f_i$  is the first contraction  $K_i = K_i$  and  $G_i = G_i$ . Let  $G_{i+1} := G_i \cup \{u, ActN_{G_i}[v \setminus u]\}$  be the graph defined in the construction Line 2.1. By construction, contracting  $v$  to  $u$  in both graphs  $G_i$  and  $G_{i+1}$  yields the same graph  $G_{i+1}$ .

## 55:10 Computing Persistent Homology of Flag Complexes via Strong Collapses

Let  $x' \in \text{Act}N_{G_i}[v \setminus u]$ . We observe that adding the edge  $[ux']$  to  $G_i$  does not change the fact that all  $x \in \{N_{G_i}[v \setminus u] \setminus \text{Act}N_{G_i}[v \setminus u]\}$  are dominated since the addition of  $[ux']$  only adds neighbors to  $N_{G_i}[x']$  and  $N_{G_i}[u]$ . Removing all the dominated vertices in  $N_{G_i}[v \setminus u]$  thus provides a sequence of elementary strong collapses. By performing all such elementary strong collapses,  $\mathbb{K}_{i+1}$  is eventually transformed into a complex  $K_{i+1}^0$ . By doing so, we have removed all the dominated vertices from  $N_{G_i}[v \setminus u]$  and added edges between  $u$  and the non dominated vertices that are in  $N_{G_i}[v \setminus u]$ . This implies that  $v$  is dominated by  $u$  in  $K_{i+1}^0$ . The elementary strong collapse of  $v$  onto  $u$ , implies  $K_{i+1}^0 \searrow \searrow K_{i+1}$  and therefore  $\mathbb{K}_{i+1} \searrow \searrow K_{i+1}$ .  $\blacktriangleleft$

► **Lemma 4.** Let  $f_i : K_i \xrightarrow{\{u,v\} \mapsto u} K_{i+1}$  be the first elementary contraction in the tower  $\mathbb{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$ . Then the following diagram commutes

$$\begin{array}{ccc} H_p(K_i) & \xrightarrow{f_i^*} & H_p(K_{i+1}) \\ & \searrow i^* & \downarrow (i')^* \\ & & H_p(\mathbb{K}_{i+1}) \end{array}$$

Here  $i' : K_{i+1} \hookrightarrow \mathbb{K}_{i+1}$  is the inclusion induced by the strong collapse.  $i^*$  and  $(i')^*$  are homomorphisms induced by the inclusion maps.

**Proof.** Using the fact that  $f_i$  is the first contraction, we have the inclusion  $\mathbb{K}_i = K_i \subseteq \mathbb{K}_{i+1}$ . Let  $K_{i+1}^0$  be the complex as defined in the proof of Lemma 3. Consider the following diagram of simplicial complexes, and note that  $i' = i_1 \circ i_0$  where  $i_0$  and  $i_1$  are both inclusions induced by the respective strong collapses.

$$\begin{array}{ccc} K_i & \xrightarrow{f_i} & K_{i+1} \\ \downarrow i & & \downarrow i_0 \\ \mathbb{K}_{i+1} & \xleftarrow{i_1} & K_{i+1}^0 \end{array}$$

We claim that the maps  $i' \circ f_i$  and  $i$  are contiguous, which we denote  $i' \circ f_i \sim i$ . Indeed, let  $\sigma$  be any simplex in  $K_i$ . Since  $i$  is an inclusion,  $i(\sigma) = \sigma$ .

**Case 1.** If  $v \notin \sigma$ , then  $i' \circ f_i(\sigma) = \sigma = i(\sigma)$ .

**Case 2.** If  $v \in \sigma$ ,  $f_i(\sigma)$  is a simplex  $\gamma \in K_{i+1}$  that contains  $u$  and, since  $i_0$  is an inclusion,  $i_0 \circ f_i(\sigma) = \gamma$ . Observe that, in the retraction map associated to the strong collapse  $r_1 : \mathbb{K}_{i+1} \searrow \searrow K_{i+1}^0$ ,  $v$  is not contracted (by construction of  $K_{i+1}^0$ ). Therefore  $r_1 \circ i(\sigma)$  is a simplex  $\gamma' \in K_{i+1}^0$  that contains  $v$ .

Now, as mentioned in the proof of Lemma 3,  $u$  dominates  $v$  in  $K_{i+1}^0$ . Therefore all the maximal simplices in  $K_{i+1}^0$  that contain  $v$  also contain  $u$  (Remark 1). Therefore,  $\gamma'$  is a face of a maximal simplex  $\tau \in K_{i+1}^0$  that contains  $u$  (in addition to  $v$ ).

Since  $\gamma$  is obtained by contracting  $v$  to  $u$ ,  $\gamma$  must be a face of  $\tau$  that contains both  $u$  and  $v$ . This implies that  $\gamma' \cup \gamma \subseteq \tau$ , which in turn implies that  $r_1 \circ i(\sigma)$  is contiguous to  $i_0 \circ f_i(\sigma)$ . After composing both sides with  $i_1$ , we get  $i_1 \circ r_1 \circ i(\sigma) \sim i_1 \circ i_0 \circ f_i(\sigma)$ . Now since  $\mathbb{K}_{i+1} \searrow \searrow K_{i+1}^0$ ,  $i_1 \circ r_1 \sim 1_{\mathbb{K}_{i+1}}$  [3], where  $1_{\mathbb{K}_{i+1}}$  is the identity over  $\mathbb{K}_{i+1}$ . As  $i_1 \circ i_0 = i'$ , we have  $i' \circ f_i(\sigma) \sim i(\sigma)$ .

Combining both the cases we conclude  $i' \circ f_i \sim i$ .

Since contiguous maps are homotopic at the level of geometric realizations, the diagram in the lemma commutes.  $\blacktriangleleft$

Proceeding by induction, Lemma 3 then immediately implies the following result.

► **Lemma 5.** *Given a tower  $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$ . For each  $0 \leq i \leq m$ ,  $\mathbb{K}_i \searrow \searrow K_i$ .*

Again, using an inductive argument along with Lemmas 4 and 5, we can deduce the following result.

► **Theorem 6.** *The following diagram commutes and all the vertical maps  $\phi_i^*$  are isomorphisms. As a consequence, the tower  $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$  and the constructed filtration  $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow \dots \hookrightarrow \mathbb{K}_m$  have the same persistence diagram.*

$$\begin{array}{ccccccc}
 H_p(\mathbb{K}_1) & \xleftarrow{*} & H_p(\mathbb{K}_2) & \xleftarrow{*} & \dots & \longrightarrow & H_p(\mathbb{K}_{m-1}) & \xleftarrow{*} & H_p(\mathbb{K}_m) \\
 \downarrow \phi_1^* & & \downarrow \phi_2^* & & & & \downarrow \phi_{m-1}^* & & \downarrow \phi_m^* \\
 H_p(K_1) & \xrightarrow{f_1^*} & H_p(K_2) & \xrightarrow{f_2^*} & \dots & \longrightarrow & H_p(K_{m-1}) & \xrightarrow{f_{m-1}^*} & H_p(K_m)
 \end{array}$$

Here  $\phi_i$  is a strong collapse for each  $i \in \{0, \dots, m\}$  and  $*$  indicates the induced homomorphisms.

We summarize our result in the following theorem. We write  $\mathcal{T} : K_0 \xrightarrow{f_0} K_1 \xrightarrow{f_1} \dots \xrightarrow{f_{m-1}} K_m$  for the given flag tower where, w.l.o.g.,  $K_0 = \emptyset$ . and each  $f_i$  is either an inclusion or an elementary contraction. The inclusions are not necessarily elementary but corresponds to an elementary inclusion on the graphs  $G_i$ . We denote by  $d$  the maximal dimension of the  $K_i$ s in  $\mathcal{T}$ , and by  $n$  the total number of elementary inclusions of simplices in  $\mathcal{T}$ , by  $n_c$  the total number of elementary contractions and by  $n_0$  the number of vertex inclusions in  $\mathcal{T}$ .

► **Theorem 7.** *There exists a filtration  $\mathcal{F} : \mathbb{K}_0 \hookrightarrow \mathbb{K}_1 \hookrightarrow \dots \hookrightarrow \mathbb{K}_m$ , where the inclusions are not necessarily elementary, such that  $\mathcal{T}$  and  $\mathcal{F}$  have the same persistence diagram and the size of the filtration  $|\mathbb{K}_m|$  is at most  $\mathcal{O}(d * n * \log n_0)$ . Moreover,  $\mathcal{F}$  is a flag filtration which can be computed from  $\mathcal{T}$  using only the 1-skeletons  $G_i$ s of the  $K_i$ s. The time complexity of the algorithm is  $\mathcal{O}(|\mathbb{G}_m| + n_c * k^3)$  time and its space complexity  $\mathcal{O}(n_0 * k)$ , where  $\mathbb{G}_m$  denotes the 1-skeleton of  $\mathbb{K}_m$  and  $k$  is an upper bound on the degree of the vertices in  $\mathbb{G}_m$ .*

## 5 Computational experiments

Our algorithm has been implemented for Vietoris-Rips (VR) filtrations as a C++ module named RipsCollapser. Recall that, for VR-filtrations, the filtration value of a simplex is the length of the longest edge of the simplex. RipsCollapser takes as input a VR filtration and returns the reduced flag filtration as shown above. RipsCollapser can be used in two modes: in the exact mode, the output filtration has the same PD as the input filtration while, in the approximate mode to be described below, a certified approximation is returned. The output filtration can then be sent to any software that computes the PD of a VR-filtration such as the Gudhi library (the one we chose for our experiments) [32] or Ripser [4]. Therefore RipsCollapser is to be considered as an ad-on to software computing PD. RipsCollapser will be available as an open-source package of a next release of the Gudhi library.

**Approximate persistence diagram.** Given a VR filtration, one can choose to collapse the original complexes after each edge inclusion. However, as mentioned in Section 2, we can also choose to strong collapse the complexes less often, i.e. after several edge inclusions rather than just one. This will result in a faster algorithm but comes with a cost: the computed PD

is then only approximate. We call **snapshots** the values of the scale parameter at which we choose to strong collapse the complex. The difference between two consecutive snapshots is called a **step**. We approximate the filtration value of a simplex as the value of the snapshot at which it first appears. We can observe that our algorithm will report all persistence pairs that are separated by at least one snapshot. Hence if all steps are equal to some  $\epsilon > 0$ , we will compute all the persistence pairs whose lengths are at least  $\epsilon$ . It follows that the  $l_\infty$ -bottleneck distance between the computed PD and the exact one is at most  $\epsilon$ . If instead the ratio between any two consecutive steps is taken to a constant  $\rho > 1$ , the  $l_\infty$ -bottleneck distance will be at most  $\log \rho$  after reparameterizing the filtrations on a log-log-scale [45].

**Experimental setup.** We present results on five datasets **netw-sc**, **senate**, **eleg**, **HIV** and **drag 1** that are publicly available [18]. Each dataset is given as the interpoint distance matrix. The reported time includes the time of RipsCollapser and the time to compute the persistent diagram (PD). The time of RipsCollapser includes: 1. The time taken to compute the largest 1-skeleton associated to the maximum threshold value, 2. The time taken to collapse all the sub-skeletons and assemble their cores. 3. To transform them into an equivalent flag-filtration.

The code has been compiled using the compiler ‘clang-900.0.38’ and all computations were performed on a ‘2.8 GHz Intel Core i5’ machine with 16 GB of available RAM. We took all steps to be equal. Parameter *Step* controls the quality of the approximation : if *Step* = 0, we obtain the exact PD, otherwise *Step* is an upper bound on the  $l_\infty$ -bottleneck distance between the output diagram and the exact one. RipsCollapser works irrespective of the dimension of the input complexes. However, the size of the complexes in the reduced filtration, even if much smaller than in the original filtration, might exceed the capacities of the PD computation algorithm. For this reason, we introduced a parameter *dim* and restricts PD computation to dimension at most *dim*.

Some experimental results are reported in Table 1. We first observe that the reduction done by RipsCollapser is enormous. The reduced complexes are small and of low dimension (column Size/Dim) compared to the input VR-complexes which are of dimensions respectively 57, 54 and 105 for the first three datasets **netw-sc**, **senate** and **eleg**. We also observe that, while the time taken by RipsCollapser is large for exact persistence computation, very good approximations can be obtained fast. Moreover the computing time mildly increases with the number of snapshots. This suggests that implementing the collapses in parallel would lead to further substantial improvement.

**Comparison with Ripser.** Ripser [4] is the state of the art software to compute persistent diagrams of VR filtrations. It computes the *exact* PD associated to the input filtration up to dimension *dim*. Although RipsCollapser is more complementary to Ripser than a competitor, we run Ripser<sup>1</sup> on the same datasets as in Table 1 to demonstrate the benefit of using RipsCollapser.

Results are presented in Table 2. The main observation is that Ripser performs quite well in low dimensions but its ability to handle higher dimensions is limited.

---

<sup>1</sup> We used the command

```
<./ripser inputData -format distances -threshold inputTh -dim inputDim >.
```

■ **Table 1** The columns are, from left to right: dataset (Data), number of points (Pnt), maximum value of the scale parameter (Thrsld), number of simplices (Size) and dimension of the final filtration (Dim), parameter (*dim*), time (in seconds) taken by RipsCollapser, total time (in seconds) including PD computation (Tot-Time), parameter *Step* and the number of snapshots used (Snaps).

| Data    | Pnt  | Thrsld | RipsCollapser +PD |            |          |          |             |       |
|---------|------|--------|-------------------|------------|----------|----------|-------------|-------|
|         |      |        | Size/Dim          | <i>dim</i> | Pre-Time | Tot-Time | <i>Step</i> | Snaps |
| netw-sc | 379  | 5.5    | 155/3             | $\infty$   | 7.28     | 7.38     | 0.02        | 263   |
| "       | "    | "      | 155/3             | $\infty$   | 13.93    | 14.03    | 0.01        | 531   |
| "       | "    | "      | 175/3             | $\infty$   | 366.46   | 366.56   | 0           | 8420  |
| senate  | 103  | 0.415  | 405/4             | $\infty$   | 2.53     | 2.54     | 0.001       | 403   |
| "       | "    | "      | 417/4             | $\infty$   | 15.96    | 15.98    | 0           | 2728  |
| eleg    | 297  | 0.3    | 577K/15           | $\infty$   | 11.65    | 26.02    | 0.001       | 284   |
| "       | "    | "      | 835K/16           | $\infty$   | 518.36   | 540.40   | 0           | 9850  |
| HIV     | 1088 | 1050   | 127.3M/?          | 4          | 660      | 3,955    | 4           | 184   |
| drag1   | 1000 | 0.05   | 478.3M/?          | 4          | 687      | 14,170   | 0.0002      | 249   |

■ **Table 2** Time is the total time (in seconds) taken by Ripser.  $\infty$  means that the experiment ran longer than 12 hours or crashed due to memory overload.

| Data    | Pnt  | Threshold | Val        |       | Val        |          | Val        |          |
|---------|------|-----------|------------|-------|------------|----------|------------|----------|
|         |      |           | <i>dim</i> | Time  | <i>dim</i> | Time     | <i>dim</i> | Time     |
| netw-sc | 379  | 5.5       | 4          | 25.3  | 5          | 231.2    | 6          | $\infty$ |
| senate  | 103  | 0.415     | 3          | 0.52  | 4          | 5.9      | 5          | 52.3     |
| "       | "    | "         | 6          | 406.8 | 7          | $\infty$ |            |          |
| eleg    | 297  | 0.3       | 3          | 8.9   | 4          | 217      | 5          | $\infty$ |
| HIV     | 1088 | 1050      | 2          | 31.35 | 3          | $\infty$ |            |          |
| drag1   | 1000 | 0.05      | 3          | 249   | 4          | $\infty$ |            |          |

References

- 1 M. Adamaszek and J. Stacho. Complexity of simplicial homology and independence complexes of chordal graphs. *Computational Geometry: Theory and Applications*, 57:8–18, 2016.
- 2 D. Attali, A. Lieutier, and D. Salinas. Efficient data structure for representing and simplifying simplicial complexes in high dimensions. *International Journal of Computational Geometry and Applications (IJCGA)*, 22:279–303, 2012.
- 3 J. A. Barmak and E. G. Minian. Strong homotopy types, nerves and collapses. *Discrete and Computational Geometry*, 47:301–328, 2012.
- 4 U. Bauer. Ripser. URL: <https://github.com/Ripser/ripser>.
- 5 U. Bauer, M. Kerber, and J. Reininghaus. Clear and compress: Computing persistent homology in chunks. In *Topological Methods in Data Analysis and Visualization III, Mathematics and Visualization*, pages 103–117. Springer, 2014.
- 6 U. Bauer, M. Kerber, J. Reininghaus, and H. Wagner. PHAT – persistent homology algorithms toolbox. *Journal of Symbolic Computation*, 78, 2017.
- 7 J-D. Boissonnat and C. S. Karthik. An Efficient Representation for Filtrations of Simplicial Complexes. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017.
- 8 J-D. Boissonnat, S.Pritam, and D. Pareek. Strong Collapse for Persistence. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, 2018.



- 9 M. Botnan and G. Spreemann. Approximating persistent homology in Euclidean space through collapses. *Applicable Algebra in Engineering, Communication and Computing*, 26:73–101, 2015.
- 10 G. Carlsson and V. de Silva. Zigzag Persistence. *Found Comput Math*, 10, 2010.
- 11 G. Carlsson, V. de Silva, and D. Morozov. Zigzag persistent homology and real-valued functions. *SOCG*, pages 247–256, 2009.
- 12 G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the Local Behavior of Spaces of Natural Images. In: *International Journal of Computer Vision*, 76:1–12, 2008.
- 13 J. M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. In: *Proceedings of the National Academy of Sciences*, 110:18566–18571, 2013.
- 14 F. Chazal and S. Oudot. Towards Persistence-Based Reconstruction in Euclidean Spaces. *SOCG*, 2008.
- 15 C. Chen and M. Kerber. Persistent homology computation with a twist. In *European Workshop on Computational Geometry (EuroCG)*, pages 197–200, 2011.
- 16 Aruni Choudhary, Michael Kerber, and Sharath Raghvendra. Polynomial-Sized Topological Approximations Using the Permutahedron. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: <http://drops.dagstuhl.de/opus/volltexte/2016/5923>, doi:10.4230/LIPIcs.SoCG.2016.31.
- 17 H. Edelsbrunner D. Cohen-Steiner and J. Harer. Stability of Persistence Diagrams. *Discrete and Computatational Geometry*, 37:103–120, 2007.
- 18 Datasets. URL: <https://github.com/n-otter/PH-roadmap/> .
- 19 V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. In: *Algebraic and Geometric Topology*, 7:339 – 358, 2007.
- 20 H. Derksen and J. Weyman. Quiver representations. *Notices of the American Mathematical Society*, 52(2):200–206, February 2005.
- 21 T. K. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. *Publications de l’Institut Mathematique (Beograd)*, 60:23–45, 1999.
- 22 T. K. Dey, F. Fan, and Y. Wang. Computing Topological Persistence for Simplicial Maps. In *Symposium on Computational Geometry (SoCG)*, pages 345–354, 2014.
- 23 T. K. Dey, D. Shi, and Y. Wang. *SimBa: An efficient tool for approximating Rips-filtration persistence via Simplicial Batch-collapse*. In European Symp. on Algorithms (ESA), pages 35:1–35:16, 2016.
- 24 T. K. Dey and R. Slechta. Filtration Simplification for Persistent Homology via Edge Contraction. *International Conference on Discrete Geometry for Computer Imagery*, 2019.
- 25 C. H. Dowker. Homology groups of relations. *The Annals of Mathematics*, 56:84–95, 1952.
- 26 P. Dłotko and H. Wagner. Simplification of complexes for persistent homology computations. *Homology, Homotopy and Applications*, 16:49–63, 2014.
- 27 H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.
- 28 H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28:511–533, 2002.
- 29 B. T. Fasy, J. Kim, F. Lecci, and C. Maria:. Introduction to the R package TDA. *CoRR abs/1411.1830*, 2014.
- 30 E. Fieux and J. Lacaze. Foldings in graphs and relations with simplicial complexes and posets. *Discrete Mathematics*, 312(17):2639–2651, 2012.
- 31 F. Le Gall. Powers of tensors and fast matrix multiplication. *ISSAC ’*, 14:296–303, 2014.
- 32 Gudhi: Geometry understanding in Higher Dimensions. URL: <http://gudhi.gforge.inria.fr/>.
- 33 A. Hatcher. *Algebraic Topology*. Univ. Press Cambridge, 2001.
- 34 C. S. Karthik J-D. Boissonnat and S. Tavenas. Building Efficient and Compact Data Structures for Simplicial Complexes. *Algorithmica*, 79:530–567, 2017.

- 35 M. Kerber and H. Schreiber. Barcodes of Towers and a Streaming Algorithm for Persistent Homology. *33rd International Symposium on Computational Geometry*, 2017. arXiv:1701.02208.
- 36 M. Kerber and R. Sharathkumar. Approximate Čech Complex in Low and High Dimensions. In *Algorithms and Computation*, pages 666–676. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science, 2013.
- 37 C. Maria and S. Oudot. Zigzag Persistence via Reflections and Transpositions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)* pp. 181–199, January 2015.
- 38 N. Milosavljevic, D. Morozov, and P. Skraba. Zigzag persistent homology in matrix multiplication time. In *Symposium on Computational Geometry (SoCG)*, 2011.
- 39 K. Mischaikow and V. Nanda. Morse Theory for Filtrations and Efficient Computation of Persistent Homology. *Discrete and Computational Geometry*, 50:330–353, September 2013.
- 40 D. Mozozov. Dionysus. URL: <http://www.mrzv.org/software/dionysus/>.
- 41 J. Munkres. *Elements of Algebraic Topology*. Perseus Publishing, 1984.
- 42 N. Otter, M. Porter, U. Tillmann, P. Grindrod, and H. Harrington. A roadmap for the computation of persistent homology. *EPJ Data Science, Springer Nature*, page 6:17, 2017.
- 43 J. Perea and G. Carlsson. A Klein-Bottle-Based Dictionary for Texture Representation. In: *International Journal of Computer Vision*, 107:75–97, 2014.
- 44 H. Schreiber. Sophia. URL: <https://bitbucket.org/schreiberh/sophia/>.
- 45 D. Sheehy. Linear-Size Approximations to the Vietoris–Rips Filtration. *Discrete and Computational Geometry*, 49:778–796, 2013.
- 46 M. Tancer. Recognition of collapsible complexes is NP-complete. *Discrete and Computational Geometry*, 55:21–38, 2016.
- 47 J. H. C Whitehead. Simplicial spaces nuclei and m-groups. *Proc. London Math. Soc.*, 45:243–327, 1939.
- 48 A. C. Wilkerson, H. Chintakunta, and H. Krim. Computing persistent features in big data: A distributed dimension reduction approach. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2014.
- 49 A. C. Wilkerson, T. J. Moore, A. Swami, and A. H. Krim. Simplifying the homology of networks via strong collapses. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 11–15, 2013.
- 50 A. Zomorodian. The Tidy Set: A Minimal Simplicial Set for Computing Homology of Clique Complexes. In *Proceedings of the Twenty-sixth Annual Symposium on Computational Geometry*, pages 257–266, 2010.
- 51 A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 2005.



# Ham-Sandwich Cuts and Center Transversals in Subspaces

Patrick Schneider

Department of Computer Science, ETH Zürich

<http://people.inf.ethz.ch/schnpatr/>

patrick.schneider@inf.ethz.ch

---

## Abstract

The Ham-Sandwich theorem is a well-known result in geometry. It states that any  $d$  mass distributions in  $\mathbb{R}^d$  can be simultaneously bisected by a hyperplane. The result is tight, that is, there are examples of  $d + 1$  mass distributions that cannot be simultaneously bisected by a single hyperplane. In this abstract we will study the following question: given a continuous assignment of mass distributions to certain subsets of  $\mathbb{R}^d$ , is there a subset on which we can bisect more masses than what is guaranteed by the Ham-Sandwich theorem?

We investigate two types of subsets. The first type are linear subspaces of  $\mathbb{R}^d$ , i.e.,  $k$ -dimensional flats containing the origin. We show that for any continuous assignment of  $d$  mass distributions to the  $k$ -dimensional linear subspaces of  $\mathbb{R}^d$ , there is always a subspace on which we can simultaneously bisect the images of all  $d$  assignments. We extend this result to center transversals, a generalization of Ham-Sandwich cuts. As for Ham-Sandwich cuts, we further show that for  $d - k + 2$  masses, we can choose  $k - 1$  of the vectors defining the  $k$ -dimensional subspace in which the solution lies.

The second type of subsets we consider are subsets that are determined by families of  $n$  hyperplanes in  $\mathbb{R}^d$ . Also in this case, we find a Ham-Sandwich-type result. In an attempt to solve a conjecture by Langerman about bisections with several cuts, we show that our underlying topological result can be used to prove this conjecture in a relaxed setting.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Ham-Sandwich cuts, center transversal, topological methods

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.56

**Related Version** <https://arxiv.org/abs/1903.12516>

## 1 Introduction

The famous Ham-Sandwich theorem (see e.g. [15, 19], Chapter 21 in [20]) is a central result in geometry that initiated a significant amount of research on several ways to partition mass distributions. It states that any  $d$  mass distributions in  $\mathbb{R}^d$  can be simultaneously bisected by a hyperplane. A ( $d$ -dimensional) *mass distribution*  $\mu$  on  $\mathbb{R}^d$  is a measure on  $\mathbb{R}^d$  such that all open subsets of  $\mathbb{R}^d$  are measurable,  $0 < \mu(\mathbb{R}^d) < \infty$  and  $\mu(S) = 0$  for every lower-dimensional subset  $S$  of  $\mathbb{R}^d$ . An intuitive example of a mass distribution is, for example, the volume of some full-dimensional geometric object in  $\mathbb{R}^d$ . The Ham-Sandwich theorem has been generalized in several ways. One famous generalization is the polynomial Ham-Sandwich theorem, which states that any  $\binom{n+d}{d} - 1$  mass distributions in  $\mathbb{R}^d$  can be simultaneously bisected by an algebraic surface of degree  $n$  [19]. Another extension is the center transversal theorem, which generalizes the result to flats of lower dimensions:

► **Theorem 1** (Center transversal theorem [8, 22]). *Let  $\mu_1, \dots, \mu_k$  be  $k$  mass distributions in  $\mathbb{R}^d$ , where  $k \leq d$ . Then there is a  $(k - 1)$ -dimensional affine subspace  $g$  such that every halfspace containing  $g$  contains at least a  $\frac{1}{d-k+2}$ -fraction of each mass.*

We call such an affine subspace a  $(k - 1, d)$ -center transversal. For  $k = d$ , we get the statement of the Ham-Sandwich theorem. Further, for  $k = 1$ , we get another well-known result in geometry, the so called *Centerpoint theorem* [17].



© Patrick Schneider;

licensed under Creative Commons License CC-BY

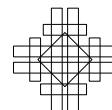
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 56; pp. 56:1–56:15

Leibniz International Proceedings in Informatics



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



In this work we will consider two different generalizations of the Ham-Sandwich theorem. The first one is about Ham-Sandwich cuts in linear subspaces. More precisely, we define a *mass assignment* on  $G_k(\mathbb{R}^d)$  as a continuous assignment  $\mu : G_k(\mathbb{R}^d) \rightarrow M_k$ , where  $G_k(\mathbb{R}^d)$  denotes the *Grassmann manifold* consisting of all  $k$ -dimensional linear subspaces of  $\mathbb{R}^d$  and  $M_k$  denotes the space of all  $k$ -dimensional mass distributions. In other words,  $\mu$  continuously assigns a mass distribution  $\mu^h := \mu(h)$  to each  $k$ -dimensional linear subspace  $h$  of  $\mathbb{R}^d$ . Examples of mass assignments include projections of higher dimensional mass distributions to  $h$  or the volume of intersections of  $h$  with (sufficiently smooth) higher dimensional geometric objects. Also, mass distributions in  $\mathbb{R}^d$  can be viewed as mass assignments on  $G_d(\mathbb{R}^d)$ . In fact, in this paper, we will use the letter  $\mu$  both for mass distributions as well as for mass assignments. The Ham-Sandwich theorem says that on every subspace we can simultaneously bisect the images of  $k$  mass assignments. But as there are many degrees of freedom in choosing subspaces, it is conceivable that there is some subspace on which we can simultaneously bisect more than  $k$  images of mass assignments. We will show that this is indeed the case, even for the more general notion of center transversals:

► **Theorem 2.** *Let  $\mu_1, \dots, \mu_{n+d-k}$  be mass assignments on  $G_k(\mathbb{R}^d)$ , where  $n \leq k \leq d$ . Then there exists a  $k$ -dimensional linear subspace  $h$  such that  $\mu_1^h, \dots, \mu_{n+d-k}^h$  have a common  $(n-1, k)$ -center transversal.*

In particular, for  $k = n$  we get that there is always a subspace on which we can simultaneously bisect  $d$  images of mass assignments. This result will only be proved in Section 4. First we will look at a conjecture by Barba [2] which motivated this generalization: Let  $\ell$  and  $\ell'$  be two lines in  $\mathbb{R}^3$  in general position. We say that  $\ell$  is above  $\ell'$  if the unique vertical line that intersects both  $\ell$  and  $\ell'$  visits first  $\ell$  and then  $\ell'$  when traversed from top to bottom.

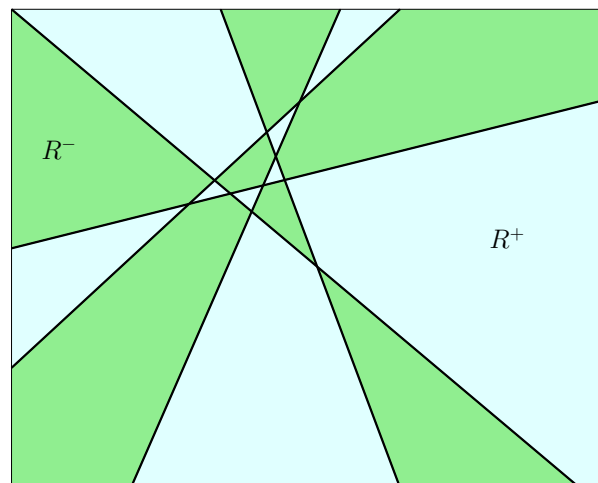
► **Conjecture 3.** *Given three sets  $R, B$  and  $G$  of lines in  $\mathbb{R}^3$  in general position, each with an even number of lines, there is a line  $\ell$  in  $\mathbb{R}^3$  such that  $\ell$  lies below exactly  $|R|/2$  lines of  $R$ ,  $|B|/2$  lines of  $B$  and  $|G|/2$  lines of  $G$ . That is, there is some Ham-Sandwich line that simultaneously bisects (with respect to above-below relation) the lines of  $R, B$  and  $G$ .*

It should be mentioned that Barba et al. have shown that the analogous statement for four sets of lines is false [2]. The conjecture can also be phrased in a slightly different terminology: Given three sets  $R, B$  and  $G$  of lines in  $\mathbb{R}^3$  in general position, each with an even number of lines, there is a vertical plane  $h$  such that  $R \cap h, B \cap h$  and  $G \cap h$  can be simultaneously bisected by a line in  $h$ . Here,  $h$  is not restricted to contain the origin, but it is restricted to be vertical, i.e., it has to be parallel to the  $z$ -axis. We will prove a stronger statement of this conjecture by showing that  $h$  can always be chosen to contain the origin.

More generally, in the setting of mass assignments, we show that at the cost of some masses, we can always fix  $k-1$  vectors in the considered subspaces. Without loss of generality, we assume that these vectors are vectors of the standard basis of  $\mathbb{R}^d$ . We say that a linear subspace of  $\mathbb{R}^d$  is  $m$ -horizontal, if it contains  $e_1, \dots, e_m$ , where  $e_i$  denotes the  $i$ 'th unit vector of  $\mathbb{R}^d$ , and we denote the space of all  $m$ -horizontal,  $k$ -dimensional subspaces of  $\mathbb{R}^d$  by  $Hor_k^m(\mathbb{R}^d)$ .

► **Theorem 4.** *Let  $\mu_1, \dots, \mu_{d-k+2}$  be mass assignments on  $Hor_k^{k-1}(\mathbb{R}^d)$ , where  $2 \leq k \leq d$ . Then there exists a  $k$ -dimensional  $(k-1)$ -horizontal linear subspace  $h$  where  $\mu_1^h, \dots, \mu_{d-k+2}^h$  have a common Ham-Sandwich cut.*

This result will be proved in Section 2. The proof of Conjecture 3 follows, after some steps to turn the lines into mass assignments, from the case  $d = 3$  and  $k = 2$ . This will be made explicit in Section 3.



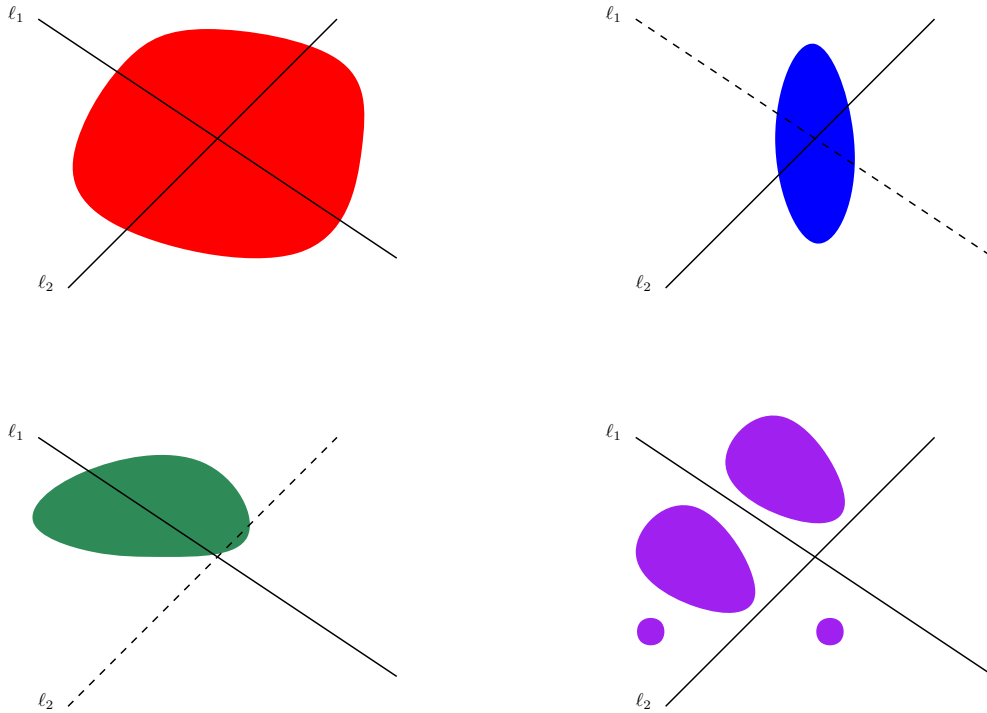
■ **Figure 1** The regions  $R^+$  (light blue) and  $R^-$  (green).

The second generalization of the Ham-Sandwich theorem that we investigate in this paper considers bisections with several cuts, where the masses are distributed into two parts according to a natural 2-coloring of the induced arrangement. More precisely, let  $\mathcal{L}$  be a set of oriented hyperplanes. For each  $\ell \in \mathcal{L}$ , let  $\ell^+$  and  $\ell^-$  denote the positive and negative side of  $\ell$ , respectively (we consider the sign resulting from the evaluation of a point in these sets into the linear equation defining  $\ell$ ). For every point  $p \in \mathbb{R}^d$ , define  $\lambda(p) := |\{\ell \in \mathcal{L} \mid p \in \ell^+\}|$  as the number of hyperplanes that have  $p$  in their positive side. Let  $R^+ := \{p \in \mathbb{R}^d \mid \lambda(p) \text{ is even}\}$  and  $R^- := \{p \in \mathbb{R}^d \mid \lambda(p) \text{ is odd}\}$ . More intuitively, this definition can also be understood the following way: if  $C$  is a cell in the hyperplane arrangement induced by  $\mathcal{L}$ , and  $C'$  is another cell sharing a facet with  $C$ , then  $C$  is a part of  $R^+$  if and only if  $C'$  is a part of  $R^-$ . See Figure 1 for an example. A similar setting, where the directions of the hyperplanes are somewhat restricted, has been studied by several authors [1, 5, 13].

We say that  $\mathcal{L}$  *bisects* a mass distribution  $\mu$  if  $\mu(R^+) = \mu(R^-)$ . Note that reorienting one hyperplane just maps  $R^+$  to  $R^-$  and vice versa. In particular, if a set  $\mathcal{L}$  of oriented hyperplanes simultaneously bisects a family of mass distributions  $\mu_1, \dots, \mu_k$ , then so does any set  $\mathcal{L}'$  of the same hyperplanes with possibly different orientations. Thus we can ignore the orientations and say that a set  $\mathcal{L}$  of (undirected) hyperplanes simultaneously bisects a family of mass distributions if some orientation of the hyperplanes does. Langerman [14] conjectured the following:

► **Conjecture 5.** *Any  $dn$  mass distributions in  $\mathbb{R}^d$  can be simultaneously bisected by  $n$  hyperplanes.*

For  $n = 1$ , this is again the Ham-Sandwich theorem. For  $d = 1$ , this conjecture is also true, this result is known as the *Necklace splitting theorem* [11, 15]. Recently, the conjecture has been proven for several values of  $n$  and  $d$  [3, 4, 6, 12], but it is still open in its full generality. In this work, we will not prove this conjecture, but we will consider a relaxed version of it: We say that  $\mathcal{L}$  *almost bisects*  $\mu$  if there is an  $\ell \in \mathcal{L}$  such that  $\mathcal{L} \setminus \{\ell\}$  bisects  $\mu$ . For a family of mass distributions  $\mu_1, \dots, \mu_k$  we say that  $\mathcal{L}$  *almost simultaneously bisects*  $\mu_1, \dots, \mu_k$  if for every  $i \in \{1, \dots, k\}$   $\mathcal{L}$  either bisects or almost bisects  $\mu_i$ . See Figure 2 for an illustration. In this relaxed setting, we are able to prove the following:



■ **Figure 2** The lines  $\ell_1$  and  $\ell_2$  almost simultaneously bisect four masses.

► **Theorem 6.** *Let  $\mu_1, \dots, \mu_{dn}$  be  $dn$  mass distributions in  $\mathbb{R}^d$ . Then there are  $n$  hyperplanes that almost simultaneously bisect  $\mu_1, \dots, \mu_{dn}$ .*

We hope that our methods might extend to a proof of Conjecture 5. We will first prove a similar result where we enforce that all bisecting hyperplanes contain the origin. The general version then follows from lifting the problem one dimension higher. The proof is based on the following idea: for each mass,  $n - 1$  of the hyperplanes define two regions, one we take with positive sign, the other with negative sign. This defines a so called *charge* (a mass distribution, which unfortunately is locally negative, which is why we will need the relaxed setting). The  $n$ 'th hyperplane should now bisect this new mass distribution. However, this  $n$ 'th hyperplane now again changes the other mass distributions, so in the end we want to guarantee that there are  $n$  hyperplanes such that all of them correctly bisect the masses. More precisely, let  $G_{d-1}(\mathbb{R}^d)^n$  be the space of all sets of  $n$  hyperplanes containing the origin (i.e., linear subspaces) in  $\mathbb{R}^d$ . Similar to before, we define a mass assignment  $\mu$  on  $G_{d-1}(\mathbb{R}^d)^n$  as a continuous assignment  $G_{d-1}(\mathbb{R}^d)^n \rightarrow M_d$ , where  $M_d$  again denotes the space of all  $d$ -dimensional mass distributions. In other words,  $\mu$  continuously assigns a mass distribution  $\mu^p := \mu(p)$  to  $\mathbb{R}^d$  for each  $p = (h_1, \dots, h_n) \in Gr_{d-1}(\mathbb{R}^d)^n$ . An example of such mass assignments could be the intersection of a fixed  $d$ -dimensional mass distribution with the Minkowski sum of the hyperplanes with a unit ball. In Section 5, we will prove the following:

► **Theorem 7.** *Let  $\mu_1, \dots, \mu_{(d-1)n}$  be  $(d - 1)n$  mass assignments on  $G_{d-1}(\mathbb{R}^d)^n$ . Then there exists  $p = (h_1, \dots, h_n) \in Gr_{d-1}(\mathbb{R}^d)^n$  such that for every  $i \in \{1, \dots, n\}$ , the hyperplane  $h_i$  simultaneously bisects  $\mu_{(d-1)(i-1)+1}^p, \dots, \mu_{(d-1)i}^p$ .*

We then use the underlying topological result to prove Theorem 6 in Section 6. All the results are proved using topological methods, and the underlying topological results might be of independent interest. For an introduction to topological methods, we refer to the



books by Matoušek [15] and de Longueville [7]. Most of the proofs in this work use so-called Stiefel-Whitney classes of vector bundles. The standard reference for this concept is the classic book by Milnor and Stasheff [16].

## 2 Ham Sandwich Cuts in horizontal subspaces

In order to prove Theorem 4, we establish a few preliminary lemmas. Consider the following space, which we denote by  $F_{hor}$ : the elements of  $F_{hor}$  are pairs  $(h, \vec{\ell})$ , where  $h$  is an (unoriented)  $k$ -dimensional  $(k - 1)$ -horizontal linear subspace of  $\mathbb{R}^d$  and  $\vec{\ell}$  is an oriented 1-dimensional linear subspace of  $h$ , that is, an oriented line in  $h$  through the origin. The space  $F_{hor}$  inherits a topology from the Stiefel manifold. Furthermore, inverting the orientation of  $\vec{\ell}$  is a free  $\mathbb{Z}_2$ -action, giving  $F_{hor}$  the structure of a  $\mathbb{Z}_2$ -space.

We will first give a different description of the space  $F_{hor}$ . Define

$$F' := S^{d-k} \times S^{k-2} \times [0, 1] / (\approx_0, \approx_1),$$

where  $(x, y, 0) \approx_0 (x, y', 0)$  for all  $y, y' \in S^{k-2}$  and  $(x, y, 1) \approx_1 (-x, y, 1)$  for all  $x \in S^{d-k}$ . Further, define a free  $\mathbb{Z}_2$ -action on  $F'$  by  $-(x, y, t) := (-x, -y, t)$ . We claim that the  $\mathbb{Z}_2$ -space  $F'$  is “the same” as  $F_{hor}$ :

► **Lemma 8.** *There is a  $\mathbb{Z}_2$ -equivariant homeomorphism between  $F'$  and  $F_{hor}$ .*

**Proof.** Consider the subspace  $Y \subset \mathbb{R}^d$  spanned by  $e_1, \dots, e_{k-1}$ . The space of unit vectors in  $Y$  is homeomorphic to  $S^{k-2}$ . Similarly let  $X \subset \mathbb{R}^d$  be spanned by  $e_k, \dots, e_d$ . Again, the space of unit vectors in  $X$  is homeomorphic to  $S^{d-k}$ . In a slight abuse of notation, we will write  $y$  and  $x$  both for a unit vector in  $Y$  and  $X$  as well as for the corresponding points in  $S^{k-2}$  and  $S^{d-k}$ , respectively.

We first construct a map  $\varphi$  from  $S^{d-k} \times S^{k-2} \times [0, 1]$  to  $F_{hor}$  as follows: for every  $x \in S^{d-k}$  let  $h(x)$  be the unique  $(k - 1)$ -horizontal subspace spanned by  $x, e_1, \dots, e_{k-1}$ . See Figure 3 for an illustration. Note that  $h(-x) = h(x)$ . Further, define  $v(x, y, t) := (1 - t)x + ty$  and let  $\vec{\ell}(x, y, t)$  be the directed line defined by the vector  $v(x, y, t)$ . Note that  $\vec{\ell}(x, y, t)$  lies in the plane spanned by  $x$  and  $y$  and thus also in  $h(x)$ . Finally, set  $\varphi(x, y, t) := (h(x), \vec{\ell}(x, y, t))$ . Both  $h$  and  $v$  are both open and closed continuous maps, and thus so is  $\varphi$ . Also, we have that  $v(-x - y, t) = -(1 - t)x - ty = -v(x, y, t)$ , so  $\varphi$  is  $\mathbb{Z}_2$ -equivariant.

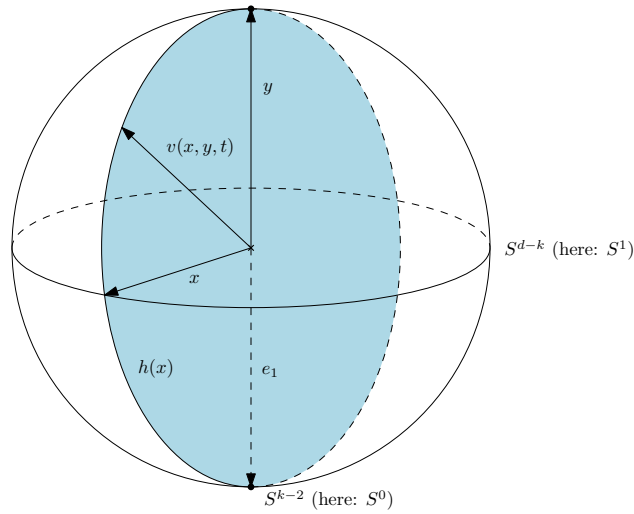
Note that for  $t = 0$  we have  $v(x, y, 0) = x$ , so  $\varphi(x, y, 0)$  does not depend on  $y$ , and in particular  $\varphi(x, y, 0) = \varphi(x, y', 0)$  for all  $y, y' \in S^{k-2}$ . Similarly, for  $t = 1$  we have  $v(x, y, 1) = y$  and  $h(-x) = h(x)$ , and thus  $\varphi(x, y, 1) = \varphi(-x, y, 1)$  for all  $x \in S^{d-k}$ . Hence,  $\varphi$  induces a map  $\varphi'$  from  $F'$  to  $F_{hor}$  which is still open, closed, continuous and  $\mathbb{Z}_2$ -equivariant. Finally, it is easy to see that  $\varphi'$  is bijective. Thus,  $\varphi'$  is a  $\mathbb{Z}_2$ -equivariant homeomorphism between  $F'$  and  $F_{hor}$ , as required. ◀

We now prove a Borsuk-Ulam-type statement for  $F_{hor}$ .

► **Lemma 9.** *There is no  $\mathbb{Z}_2$ -map  $f : F_{hor} \rightarrow S^{d-k}$ .*

**Proof.** Assume for the sake of contradiction that  $f$  exists. Then, by Lemma 8,  $f$  induces a map  $F : S^{d-k} \times S^{k-2} \times [0, 1] \rightarrow S^{d-k}$  with the following properties:

- (1)  $F(-x, -y, t) = -F(x, y, t)$  for all  $t \in (0, 1)$ ;
- (2)  $F(x, y, 0) = F(x, y', 0)$  for all  $y, y' \in S^{k-2}$  and  $F(-x, y, 0) = -F(x, y, 0)$  for all  $x \in S^{d-k}$ ;
- (3)  $F(x, -y, 1) = -F(x, y, 1)$  for all  $y \in S^{k-2}$  and  $F(-x, y, 1) = F(x, y, 1)$  for all  $x \in S^{d-k}$ .



■ **Figure 3** The map  $\varphi$  for  $d = 3$  and  $k = 2$ .

In particular,  $F$  is a homotopy between  $f_0(x, y) := F(x, y, 0)$  and  $f_1(x, y) := F(x, y, 1)$ . Fix some  $y_0 \in S^{k-2}$ . Then  $F$  induces a homotopy between  $g_0(x) := f_0(x, y_0)$  and  $g_1(x) := f_1(x, y_0)$ . Note that  $g_0 : S^{d-k} \rightarrow S^{d-k}$  has odd degree by property (2). On the other hand,  $g_1 : S^{d-k} \rightarrow S^{d-k}$  has even degree by property (3). Thus,  $F$  induces a homotopy between a map of odd degree and a map of even degree, which is a contradiction. ◀

We now have all tools that are necessary to prove Theorem 4.

► **Theorem 4.** *Let  $\mu_1, \dots, \mu_{d-k+2}$  be mass assignments on  $\text{Hor}_k^{k-1}(\mathbb{R}^d)$ , where  $2 \leq k \leq d$ . Then there exists a  $k$ -dimensional  $(k-1)$ -horizontal linear subspace  $h$  where  $\mu_1^h, \dots, \mu_{d-k+2}^h$  have a common Ham-Sandwich cut.*

**Proof.** For each  $\mu_i$  and  $(h, \vec{\ell})$ , consider the point  $v_i$  on  $\vec{\ell}$  for which the orthogonal hyperplane bisects  $\mu_i^h$ . (If  $v_i$  is not unique, the set of all possible such points is an interval, in which case we choose  $v_i$  as the midpoint of this interval.) This induces a continuous  $\mathbb{Z}_2$ -map  $g : F_{hor} \rightarrow \mathbb{R}^{d-k+2}$ . For  $i \in \{1 \dots, d-k+1\}$ , set  $w_i := v_i - v_{d-k+2}$ . The  $w_i$ 's then induce a continuous  $\mathbb{Z}_2$ -map  $f : F_{hor} \rightarrow \mathbb{R}^{d-k+1}$ . We want to show that there exists  $(h, \vec{\ell})$  where  $v_1 = v_2 = \dots = v_{d-k+2}$ , or equivalently,  $w_1 = \dots, w_{d-k+1} = 0$ , i.e.,  $f$  has a zero. Assume that this is not the case. Then normalizing  $f$  induces a  $\mathbb{Z}_2$ -map  $f' : F_{hor} \rightarrow S^{d-k}$ , which is a contradiction to Lemma 9. ◀

Note that the higher  $k$  is chosen, the weaker our result. In fact, for  $k > \frac{d}{2} + 1$ , our result is weaker than what we would get from the Ham-Sandwich theorem. We conjecture that this trade-off is not necessary:

► **Conjecture 10.** *Let  $\mu_1, \dots, \mu_d$  be mass assignments in  $\mathbb{R}^d$  and  $k \geq 2$ . Then there exists a  $k$ -dimensional  $(k-1)$ -horizontal linear subspace  $h$  such that  $\mu_1^h, \dots, \mu_d^h$  have a common Ham-Sandwich cut.*

### 3 Application: bisecting lines in space

Recall the setting of Conjecture 3: Given three sets  $R, B$  and  $G$  of lines in  $\mathbb{R}^3$  in general position, each with an even number of lines, is there a line  $\ell$  in  $\mathbb{R}^3$  such that  $\ell$  lies below exactly  $|R|/2$  lines of  $R$ ,  $|B|/2$  lines of  $B$  and  $|G|/2$  lines of  $G$ ? Here, general position means

that (i) no two lines are parallel, (ii) no line is vertical (i.e., parallel to the  $z$ -axis), (iii) no line intersects the  $z$ -axis and (iv) for any four lines, if there is a line intersecting all of them, the (unique) vertical plane containing this common intersecting line does not go through the origin.

We want to prove that there always is such a line  $\ell$  using Theorem 4. In order to apply Theorem 4, we need to define a mass assignment. To this end, we replace every line  $r$  in  $R$  by a very thin infinite cylinder of radius  $\varepsilon$ , centered at  $r$ . Denote the collection of cylinders obtained this way by  $R^*$ . Define  $B^*$  and  $G^*$  analogously. For each vertical plane  $h$  through the origin, let  $D_K^h$  be a disk in  $h$  centered at the origin, with some (very large) radius  $K$ . Define  $\mu_R^h$  as  $(R^* \cap h) \cap D_K^h$ . It is straightforward to show that  $\mu_R^h$  is a mass assignment. Analogously we can define mass assignments  $\mu_B^h$  and  $\mu_G^h$ . From Theorem 4, where we set  $e_1$  to be the unit vector on the  $z$ -axis, we deduce that there is a vertical plane  $h_0$  and a line  $\ell \in h_0$  such that  $\ell$  simultaneously bisects  $\mu_R^{h_0}$ ,  $\mu_B^{h_0}$  and  $\mu_G^{h_0}$ . We claim that this  $\ell$  gives a solution to Conjecture 3.

To show this, we distinguish two cases: The first case is that all the cylinders in  $R^* \cup B^* \cup G^*$  intersect  $D_K^{h_0}$ . In this case, it is a standard argument to show that  $\ell$  is a Ham-Sandwich cut of the point set  $(R \cup B \cup G) \cap h_0$ . Note that because of general position assumptions (ii) and (iv), at most one triple of points in  $(R \cup B \cup G) \cap h_0$  is collinear. As all three sets have an even number of lines, we thus have that  $\ell$  either contains two points or no point at all. Further, if it contains two points  $p_1$  and  $p_2$ , then they must have the same color. In this case, slightly rotate  $\ell$  such that  $p_1$  lies above  $\ell$  and  $p_2$  lies below  $\ell$ . Now, in any case,  $\ell$  is a Ham-Sandwich cut that contains no points. In particular,  $\ell$  lies below exactly  $|R|/2$  lines of  $R$ ,  $|B|/2$  lines of  $B$  and  $|G|/2$  lines of  $G$ , which is what we required.

The second case is that some cylinders in  $R^* \cup B^* \cup G^*$  do not intersect  $D_K^{h_0}$ . By the general position assumption (i), choosing  $K$  sufficiently large, we can assume that exactly one cylinder  $c^*$  does not intersect  $D_K^{h_0}$ . Without loss of generality, let  $c^* \in R^*$ , defined by some line  $c \in R$ . If  $K$  is chosen sufficiently large, by general position assumption (iii) we can further assume that  $c$  is parallel to  $h_0$ . Thus, similar to above,  $\ell$  is a Ham-Sandwich cut of the point set  $((R \setminus \{c\}) \cup B \cup G) \cap h_0$ . Again, at most one triple of points is collinear. As  $(R \setminus \{c\})$  contains an odd number of lines,  $\ell$  passes through either 1 or 3 points. If  $\ell$  passes through 3 points  $p_1, p_2$  and  $p_3$ , then without loss of generality  $p_1 \in (R \setminus \{c\}) \cap h_0$ . Further,  $p_2$  and  $p_3$  must be induced by the same set of lines, without loss of generality  $B$ . In both cases, we can slightly rotate  $\ell$  such that  $p_1$  is above  $\ell$  and  $p_2$  and  $p_3$  lie on different sides of  $\ell$ . Similarly, if  $\ell$  contains 1 point  $p_1$ , then  $p_1 \in (R \setminus \{c\}) \cap h_0$ , and we can slightly translate  $\ell$  such that  $p_1$  lies above  $\ell$ . Now again,  $\ell$  lies below exactly  $|R|/2$  lines of  $R$ ,  $|B|/2$  lines of  $B$  and  $|G|/2$  lines of  $G$ , which is what we required.

Thus, we have proved the following Theorem:

► **Theorem 11.** *Given three sets  $R, B$  and  $G$  of lines in  $\mathbb{R}^3$  in general position, each with an even number of lines, there is a line  $\ell$  in  $\mathbb{R}^3$  such that  $\ell$  lies below exactly  $|R|/2$  lines of  $R$ ,  $|B|/2$  lines of  $B$  and  $|G|/2$  lines of  $G$ .*

#### 4 Center Transversals in general subspaces

In this section we consider the more general case of assignments of mass distributions to all linear subspaces. The space of all linear subspaces of fixed dimension defines in a natural way a *vector bundle*. Recall the following definition: a vector bundle consists of a base space  $B$ , a total space  $E$ , and a continuous projection map  $\pi : E \mapsto B$ . Furthermore, for each  $b \in B$ , the fiber  $\pi^{-1}(b)$  over  $b$  has the structure of a vector space over the real numbers.

Finally, a vector bundle satisfies the *local triviality condition*, meaning that for each  $b \in B$  there is a neighborhood  $U \subset B$  containing  $p$  such that  $\pi^{-1}(U)$  is homeomorphic to  $U \times \mathbb{R}^d$ . A *section* of a vector bundle is a continuous mapping  $s : B \rightarrow E$  such that  $\pi s$  equals the identity map, i.e.,  $s$  maps each point of  $B$  to its fiber. Recall that we denote by  $G_m(\mathbb{R}^n)$  the Grassmann manifold consisting of all  $m$ -dimensional subspaces of  $\mathbb{R}^n$ . Let  $\gamma_m^d$  be the *canonical bundle* over  $G_m(\mathbb{R}^n)$ . The bundle  $\gamma_m^d$  has a total space  $E$  consisting of all pairs  $(L, v)$ , where  $L$  is an  $m$ -dimensional subspace of  $\mathbb{R}^n$  and  $v$  is a vector in  $L$ , and a projection  $\pi : E \rightarrow G_m(\mathbb{R}^n)$  given by  $\pi((L, v)) = L$ . Another space that we will be working with is the *complete flag manifold*  $\tilde{V}_{n,n}$ : a *flag*  $\mathcal{F}$  in a vector space  $V$  of dimension  $n$  is an increasing sequence of subspaces of the form

$$\mathcal{F} = \{0\} = V_0 \subset V_1 \subset \dots \subset V_k = V.$$

A flag is a *complete flag* if  $\dim V_i = i$  for all  $i$  (and thus  $k = n$ ). The complete flag manifold  $\tilde{V}_{n,n}$  is the manifold of all complete flags of  $\mathbb{R}^n$ . Similar to the Grassmann manifold, we can define a canonical bundle for each  $V_i$ , which we will denote by  $\vartheta_i^n$ . For details on vector bundles and sections, see [16].

► **Lemma 12.** *Let  $s_1, \dots, s_{m+1}$  be  $m + 1$  sections of the canonical bundle  $\vartheta_l^{m+l}$ . Then there is a flag  $\mathcal{F} \in \tilde{V}_{m+l, m+l}$  such that  $s_1(\mathcal{F}) = \dots = s_{m+1}(\mathcal{F})$ .*

This Lemma is a generalization of Proposition 2 in [22] and Lemma 1 in [8]. Our proof follows the proof in [22].

**Proof.** Consider the sections  $q_i := s_{m+1} - s_i$ . We want to show that there exists a flag  $\mathcal{F}$  for which  $q_1(\mathcal{F}) = \dots = q_m(\mathcal{F}) = 0$ . The sections  $q_1, \dots, q_m$  determine a unique section in the  $m$ -fold Whitney sum of  $\vartheta_l^{m+l}$ , which we denote by  $W$ . Note that  $W$  has base  $\tilde{V}_{m+l, m+l}$  and fiber dimension  $ml$ . We will show that  $W$  does not admit a nowhere zero section. For this, it suffices to show that the highest Stiefel-Whitney class  $w_{ml}(W)$  is nonzero (see [16], §4, Proposition 3).

By the Whitney product formula we have  $w_{ml}(W) = w_l(\vartheta_l^{m+l})^m$ . Note that the projection  $f : \tilde{V}_{m+l, m+l} \rightarrow G_l(\mathbb{R}^{m+l})$  which maps  $(V_0, \dots, V_l, \dots, V_{m+l})$  to  $V_l$  induces a bundle map from  $\vartheta_l^{m+l}$  to  $\gamma_l^{m+l}$ . Thus by the naturality of Stiefel-Whitney classes we have  $w_l(\vartheta_l^{m+l})^m = f^*(w_l(\gamma_l^{m+l})^m) = f^*(w_l(\gamma_l^{m+l}))^m$ . Further, we have the following commutative diagram

$$\begin{array}{ccc} \tilde{V}_{m+l, m+l} & \xrightarrow{i} & \tilde{V}_{\infty, m+l} \\ f \downarrow & & \downarrow g \\ G_l(\mathbb{R}^{m+l}) & \xrightarrow{j} & G_l(\mathbb{R}^{\infty}) \end{array} \quad ,$$

where  $i$  and  $j$  are inclusions and  $g$  is the canonical map from  $\tilde{V}_{\infty, m+l}$  to  $G_l(\mathbb{R}^{\infty})$  (see e.g. [10, 22]). In  $\mathbb{Z}_2$ -cohomology, we get the following diagram:

$$\begin{array}{ccc} H^*(\tilde{V}_{m+l, m+l}) & \xleftarrow{i^*} & H^*(\tilde{V}_{\infty, m+l}) \\ f^* \uparrow & & \uparrow g^* \\ H^*(G_l(\mathbb{R}^{m+l})) & \xleftarrow{j^*} & H^*(G_l(\mathbb{R}^{\infty})) \end{array} \quad .$$

It is known that  $H^*(\tilde{V}_{\infty, m+l})$  is a polynomial algebra  $\mathbb{Z}_2[t_1, \dots, t_{m+l}]$  and that  $g^*$  maps  $H^*(G_l(\mathbb{R}^{\infty}))$  injectively onto the algebra  $\mathbb{Z}_2[\sigma_1, \dots, \sigma_l] \subset \mathbb{Z}_2[t_1, \dots, t_l] \subset \mathbb{Z}_2[t_1, \dots, t_{m+l}]$ , where  $\sigma_i$  denotes the  $i$ 'th symmetric polynomial in the variables  $t_1, \dots, t_l$  [16, 10]. Further,

$H^*(\tilde{V}_{m+l,m+l})$  is a polynomial algebra  $\mathbb{Z}_2[t_1, \dots, t_{m+l}] / (\sigma_1, \dots, \sigma_{m+l})$  and  $i^*$  is the corresponding quotient map. Since  $w_l(\gamma_l^{m+l}) = j^*(\sigma_l)$ , we have  $w_l(\vartheta_l^{m+l})^m = f^*(j^*(\sigma_l))^m$  and in particular  $w_l(\vartheta_l^{m+l})^m = 0$  would imply that  $(\sigma_l)^m \in \ker i^*$ , i.e.  $(t_1 \cdots t_l)^m$  is in the ideal  $(\sigma_1, \dots, \sigma_{m+l})$ . But this is a contradiction to Proposition 2.21 in [21]. ◀

Consider now a continuous map  $\mu : \tilde{V}_{m+l,m+l} \rightarrow M_l$ , which assigns an  $l$ -dimensional mass distribution to  $V_l$  for every flag. We call such a map an *l-dimensional mass assignment on  $\tilde{V}_{m+l,m+l}$* .

► **Corollary 13.** *Let  $\mu_1, \dots, \mu_{m+1}$  be  $l$ -dimensional mass assignments on  $\tilde{V}_{m+l,m+l}$ . Then there exists a flag  $\mathcal{F} \ni V_l$  such that some point  $p \in V_l$  is a centerpoint for all  $\mu_1^{\mathcal{F}}, \dots, \mu_{m+1}^{\mathcal{F}}$ .*

**Proof.** For every  $\mu_i$  and every flag  $\mathcal{F}$ , the centerpoint region of  $\mu_i^{\mathcal{F}}$  is a convex compact region in the respective  $V_l$ . In particular, for each  $\mu_i$  we get a multivalued, convex, compact section  $s_i$  in  $\vartheta_l^{m+l}$ . Using Proposition 1 from [22], Lemma 12 implies that there is a Flag in which all  $s_i$ 's have a common point  $p$ . ◀

From this we can now deduce Theorem 2

► **Theorem 2.** *Let  $\mu_1, \dots, \mu_{n+d-k}$  be mass assignments on  $G_k(\mathbb{R}^d)$ , where  $n \leq k \leq d$ . Then there exists a  $k$ -dimensional linear subspace  $h$  such that  $\mu_1^h, \dots, \mu_{n+d-k}^h$  have a common  $(n-1, k)$ -center transversal.*

**Proof.** Note that a  $(n-1, k)$ -center transversal in a  $k$ -dimensional space is a common centerpoint of the projection of the masses to a  $k - (n-1)$ -dimensional subspace. Consider a flag  $\mathcal{F} = (V_0, \dots, V_d)$ . For each mass assignment  $\mu_i$  define  $\mu_i'(\mathcal{F}) := \pi_{k-(n-1)}(\mu_i^{V_k})$ , where  $\pi_{k-(n-1)}$  denotes the projection from  $V_k$  to  $V_{k-(n-1)}$ . Every  $\mu_i'$  is an  $(k - (n-1))$ -dimensional mass assignment on  $\tilde{V}_{d,d}$ . The result now follows from Corollary 13 by setting  $l = k - (n-1)$  and  $m = d - k + n - 1$ . ◀

## 5 Sections in product bundles

Similar to before, we again work with vector bundles, but now over a different space. Recall that a mass assignment  $\mu$  on  $G_{d-1}(\mathbb{R}^d)^n$  assigns a  $d$ -dimensional mass distribution  $\mu^p$  to  $\mathbb{R}^d$  for each  $p = (h_1, \dots, h_n) \in Gr_{d-1}(\mathbb{R}^d)^n$ . We want to show that given  $(d-1)n$  such mass assignments, there is a  $p$  such that each  $h_i$  bisects  $d-1$  of their images. The idea is the following: we assign  $d-1$  masses to each  $h_i$ . For every  $p$ , we now sweep a copy of  $h_i$  along a line  $\ell$  orthogonal to  $h_i$  and for every mass assigned to  $h_i$  we look at the point on  $\ell$  for which the swept copy through that point bisects the mass. We want to show that for some  $p$ , all these points coincide with the origin.

► **Lemma 14.** *Consider the vector bundle  $\xi := (\gamma_m^d)^k$  (the  $k$ -fold Cartesian product of  $\gamma_m^d$ ) over the space  $B := G_m(\mathbb{R}^d)^k$ . Let  $q := d - m$ . Then for any  $q$  sections  $s_1, \dots, s_q$  of  $\xi$  there exists  $b \in B$  such that  $s_1(b) = \dots = s_q(b) = 0$ .*

This Lemma is another generalization of Proposition 2 in [22] and Lemma 1 in [8]. Our proof follows the proof in [8].

**Proof.** The sections  $s_1, \dots, s_q$  determine a unique section in the  $q$ -fold Whitney sum of  $\xi$ , which we denote by  $\xi^q$ .  $\xi^q$  has base  $B$  and fiber dimension  $kqm$ . We want to show that  $\xi^q$  does not allow a nowhere zero section. For this, it is again enough to show that the highest Stiefel-Whitney class  $w_{kqm}(\xi^q)$  does not vanish. Denote by  $\Gamma_m^d$  the  $q$ -fold Whitney sum

of  $\gamma_m^d$  and consider the vector bundle  $\zeta := (\Gamma_m^d)^k$ . Note that  $\zeta$  also has base  $B$  and fiber dimension  $kqm$ . Furthermore, there is a natural bundle map from  $\zeta$  to  $\xi^q$ , and as they have the same base space,  $\zeta$  and  $\xi^q$  are isomorphic (see [16], §3, Lemma 3.1). Thus, it is enough to show that the highest Stiefel-Whitney class  $w_{kqm}(\zeta)$  does not vanish. The Stiefel-Whitney classes of a Cartesian product of vector bundles can be computed as the cross product of the Stiefel-Whitney classes of its components in the following way (see [16], §4, Problem 4-A):

$$w_j(\eta_1 \times \eta_2) = \sum_{i=0}^j w_i(\eta_1) \times w_{j-i}(\eta_2).$$

It was shown by Dol’nikov [8] that  $w_{qm}(\Gamma_m^d) = 1 \in \mathbb{Z}_2 = H^{qm}(G_m(\mathbb{R}^d); \mathbb{Z}_2)$ . By the Künneth theorem and induction it follows that  $w_{kqm}((\Gamma_m^d)^k) = 1 \in \mathbb{Z}_2 = H^{kqm}((G_m(\mathbb{R}^d))^k; \mathbb{Z}_2)$ . ◀

In the following, we will use Lemma 14 only for the case  $m = 1$ , i.e., for products of line bundles. This case could also be proved using a Borsuk-Ulam-type result on product of spheres (Theorem 4.1 in [9], for  $n_1 = \dots = n_r = d - 1$ , see also [18]). Consider now  $B := G_1(\mathbb{R}^d)^n$ , i.e., all  $n$ -tuples of lines in  $\mathbb{R}^d$  through the origin. Further, for every  $i \in \{1, \dots, n\}$  we define  $\xi_i$  as the following vector bundle: the base space is  $B$ , the total space  $E_i$  is the set of all pairs  $(b, v)$ , where  $b = (\ell_1(b), \dots, \ell_n(b))$  is an element of  $B$  and  $v$  is a vector in  $\ell_i(b)$ , and the projection  $\pi$  is given by  $\pi((b, v)) = b$ . It is straightforward to show that this is indeed a vector bundle. In other words, we consider one line to be marked and the fiber over an  $n$ -tuple of lines is the 1-dimensional vector space given by the marked line. We are now ready to prove Theorem 7.

► **Theorem 7.** *Let  $\mu_1, \dots, \mu_{(d-1)n}$  be  $(d - 1)n$  mass assignments on  $G_{d-1}(\mathbb{R}^d)^n$ . Then there exists  $p = (h_1, \dots, h_n) \in Gr_{d-1}(\mathbb{R}^d)^n$  such that for every  $i \in \{1, \dots, n\}$ , the hyperplane  $h_i$  simultaneously bisects  $\mu_{(d-1)(i-1)+1}^p, \dots, \mu_{(d-1)i}^p$ .*

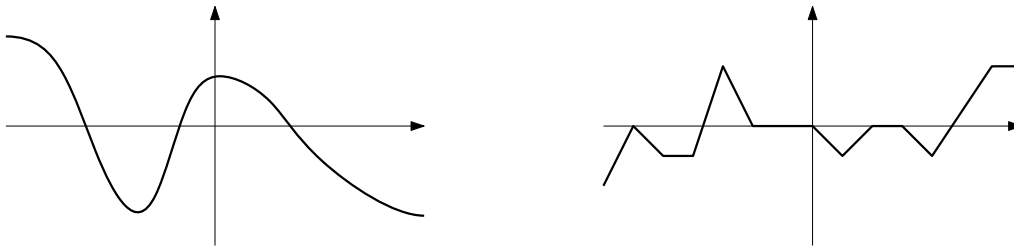
**Proof.** Consider  $\xi = (\gamma_1^d)^n$ . Recall that  $B = G_1(\mathbb{R}^d)^n$ . For an element  $b = (\ell_1(b), \dots, \ell_n(b))$  of  $B$ , consider for every  $i \in \{1, \dots, n\}$  the  $(d - 1)$ -dimensional hyperplane through the origin that is orthogonal to  $\ell_i(b)$  and denote it by  $h_i(b)$ . Similarly, for every  $(b, v) \in E_i$ , let  $g_i(b, v)$  be the hyperplane through  $v$  orthogonal to  $\ell_i(b)$ . Note that  $g_i(b, 0) = h_i(b)$ .

Consider now the mass  $\mu_1$ . The set of all pairs  $(b, v)$  such that  $(g_1(b, v), h_2(b), \dots, h_n(b))$  bisects  $\mu_1$  defines a section  $s_1^1$  in  $\xi_1$ . Analogously, use  $\mu_{(d-1)(j-1)+1}$  to define  $s_j^1$  for all  $j \in \{2, \dots, n\}$ . Then  $s^1 := (s_1^1, \dots, s_n^1)$  is a section in  $(\gamma_1^d)^n$ . Similarly, for  $i \in \{2, \dots, d - 1\}$ , using the masses  $\mu_{(d-1)(j-1)+i}$  for all  $j \in \{2, \dots, n\}$  define a section  $s^i$  in  $(\gamma_1^d)^n$ .

We have thus defined  $d - 1$  sections in  $(\gamma_1^d)^n$ . Hence, by applying Lemma 14, we get that there is a point  $b_0$  in  $B$  such that  $s^1(b_0), \dots, s^{d-1}(b_0) = 0$ . In particular, all orthogonal hyperplanes  $g_i(b, v)$  contain the origin, so their collection is an element of  $G_{d-1}(\mathbb{R}^d)^n$ . Further, it follows from the definition of the sections  $s^i$  that  $h_i$  simultaneously bisects  $\mu_{(d-1)(i-1)+1}^p, \dots, \mu_{(d-1)i}^p$ . ◀

## 6 Application: bisections with several cuts

The objective of this section is to prove Theorem 6. Before we dive into the technicalities, let us briefly discuss the main ideas. We first show that any  $(d - 1)n$  mass distributions in  $\mathbb{R}^d$  can be almost simultaneously bisected by  $n$  hyperplanes through the origin. The idea of this proof is very similar to the proof of Theorem 7: consider some mass  $\mu$  and assume that  $n - 1$  of the hyperplanes are fixed. Sweep the last hyperplane along a line through the origin and stop when the resulting arrangement of  $n$  hyperplanes almost bisects  $\mu$ . We do



■ **Figure 4** Two graphs of limit antipodal functions.

the same for every mass, one hyperplane is swept, the others are considered to be fixed. Each hyperplane is swept for  $(d - 1)$  masses. Using Lemma 12, we want to argue, that there is a solution, such that all the swept hyperplanes are stopped at the origin. The only problem with this approach is, that the points where we can stop the hyperplane are in general not unique. In fact, the region of possible solutions for one sweep can consist of several connected components, so in particular, it is not a section, and we cannot use Lemma 12 directly. We will therefore need another Lemma, that says that we can find a section in this space of possible solutions. This Lemma is actually the only reason why our approach only works for the relaxed setting: we need to sometimes ignore certain hyperplanes to construct such a section. However, constructing a section that lies completely in the space of solutions is stronger than what we would need to use Lemma 12. It would be enough to argue, that assuming no almost simultaneous bisection exists, we could find a nowhere zero section contradicting Lemma 12. It is thus possible that our approach could be strengthened to prove Conjecture 5.

Let us now start by stating the aforementioned result for bisections with hyperplanes containing the origin:

► **Theorem 15.** *Let  $\mu_1, \dots, \mu_{(d-1)n}$  be  $(d - 1)n$  mass distributions in  $\mathbb{R}^d$ . Then there are  $n$  hyperplanes, all containing the origin, that almost simultaneously bisect  $\mu_1, \dots, \mu_{(d-1)n}$ .*

As mentioned, in order to prove this result, we need a few additional observations. In the following, by a *limit antipodal* function we mean a continuous function  $f : \mathbb{R} \mapsto \mathbb{R}$  with the following two properties:

1.  $\lim_{x \rightarrow \infty} f = -\lim_{x \rightarrow -\infty} f$ ,
2. the set of zeroes of  $f$  consists of finitely many connected components.

See Figure 4 for an illustration. Note that these two conditions imply that if  $\lim_{x \rightarrow \infty} f \neq 0$  and if the graph of  $f$  is never tangent to the  $x$ -axis, the zero set consists of an odd number of components. For any subset  $A$  of a vector bundle  $\xi = (E, B, \pi)$ , denote by  $Z(A)$  the set of base points on whose fiber  $A$  contains 0 or  $A$  is unbounded. In particular, for any section  $s$ ,  $Z(s)$  denotes the set of zeroes of the section (as a section is a single point on every fiber, and thus never unbounded).

Consider again  $B := G_1(\mathbb{R}^d)^n$ , i.e., all  $n$ -tuples of lines in  $\mathbb{R}^d$  through the origin and the vector bundles  $\xi_i$ . Note that  $\xi_i$  has a natural orientable cover  $\xi'_i = (E', B', \pi')$  where all the lines are oriented. Denote by  $p$  the covering map from  $\xi'_i$  to  $\xi_i$ .

Assume now that we are given a continuous function  $f : E' \rightarrow \mathbb{R}$  with the following properties:

- (a) for every point  $b' \in B'$ , the restriction of  $f$  to the fiber  $\pi'^{-1}(b')$ , denoted by  $f_{b'}$ , is a limit antipodal function;
- (b) for any point  $b \in B$  and any two lifts  $b'_1, b'_2 \in p^{-1}(b)$  we have either  $f_{b'_1}(x) = f_{b'_2}(x)$  or  $f_{b'_1}(x) = -f_{b'_2}(x)$  or  $f_{b'_1}(x) = f_{b'_2}(-x)$  or  $f_{b'_1}(x) = -f_{b'_2}(-x)$ .

Let  $V'_f := \{e \in E' \mid f(e) = 0\}$  be the zero set of  $f$ . Note that the second condition ensures that  $V'_f$  is the lift of a set  $V_f \subseteq E$ . We call  $V_f$  a *quasi-section* in  $\xi_i$ . Further note that  $Z(V_f)$  consists of the base points where  $f_b(0) = 0$  or  $\lim_{x \rightarrow \infty} f_b = 0$ .

► **Lemma 16.** *Let  $V_f$  be a quasi-section in  $\xi_i$ . Then there is a section  $s$  such that  $Z(s) \subset Z(V_f)$ . In particular, if  $Z(V_f) = \emptyset$ , then  $\xi_i$  allows a nowhere zero section.*

Before proving this lemma, we show how to apply it to prove Theorem 15.

**Proof of Theorem 15.** Define  $h_i(b)$  and  $g_i(b, v)$  as in the proof of Theorem 7.

Consider now the mass  $\mu_1$ . For each  $b \in B$ , choose some orientations of  $h_2(b), \dots, h_n(b)$  and an orientation of  $\ell_1(b)$  arbitrarily. Then for each  $v \in \ell_1(b)$ , we have well-defined regions  $R^+(b, v)$  and  $R^-(b, v)$ . In particular, taking  $\mu_1(R^+(b, v)) - \mu_1(R^-(b, v))$  for all orientations defines a function  $f_1 : E' \rightarrow \mathbb{R}$  which satisfies condition (a) and (b) from above. Let  $V_1$  be the set of all pairs  $(b, v)$  such that  $(g_1(b, v), h_2(b), h_3(b), \dots, h_n(b))$  bisects  $\mu_1$ . As this is exactly the set of pairs  $(b, v)$  for which  $f_1(b, v) = 0$ , it follows that  $V_1$  is a quasi-section.

Let now  $s_1^1$  be a section in  $\xi_1$  with  $Z(s_1) \subset Z(V_1)$ , the existence of which we get from Lemma 16. Analogously, use  $\mu_i$  to define  $V_i$  and  $s_i^1$  for all  $i \in \{2, \dots, n\}$ . Then  $s^1 := (s_1^1, \dots, s_n^1)$  is a section in  $(\gamma_1^d)^n$ . Similarly, for  $k \in 2, \dots, d-1$ , using the masses  $\mu^{(k-1)n+1}, \dots, \mu_{kn}$  define a section  $s^k$  in  $(\gamma_1^d)^n$ .

We have thus defined  $d-1$  sections in  $(\gamma_1^d)^n$ . Hence, by applying Lemma 14, we get that there is a point  $b_0$  in  $B$  such that  $s^1(b_0), \dots, s^{d-1}(b_0) = 0$ . We claim that  $H := (h_1(b_0), \dots, h_n(b_0))$  almost simultaneously bisects  $\mu_1, \dots, \mu_{(d-1)n}$ : without loss of generality, consider the mass  $\mu_1$ . As  $s_1^1(b_0) = 0$ , we know by the definition of  $s_1^1$  that  $(b_0, 0)$  is in  $Z(V_1)$ . By the definition of  $Z(V_1)$  this means that  $V_1 \cap \pi^{-1}(b_0)$  (1) contains  $(b_0, 0)$  or (2) is unbounded.

In case (1), we get that  $(g_1(b_0, 0), h_2(b_0), \dots, h_n(b_0))$  bisects  $\mu_1$ . But since  $g_i(b_0, 0) = h_i(b_0)$ , this set is exactly  $H$ . In case (2), we notice that  $V_1$  is unbounded on  $\pi^{-1}(b_0)$  if and only if  $\lim_{x \rightarrow \infty} f_{1, b_0} = 0$ . But this means that  $(h_2(b_0), \dots, h_n(b_0))$  bisects  $\mu_1$ . Thus,  $H$  indeed almost bisects  $\mu_1$ . ◀

From Theorem 15 we also deduce the main result of this section:

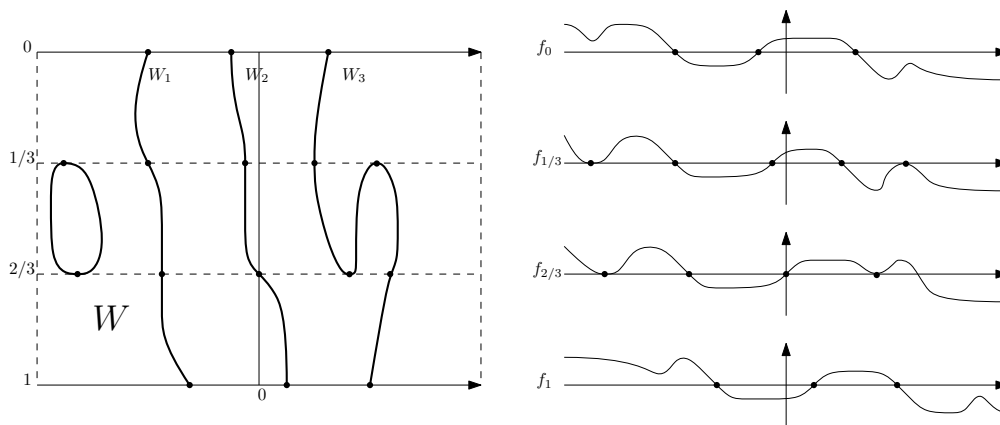
► **Theorem 6.** *Let  $\mu_1, \dots, \mu_{dn}$  be  $dn$  mass distributions in  $\mathbb{R}^d$ . Then there are  $n$  hyperplanes that almost simultaneously bisect  $\mu_1, \dots, \mu_{dn}$ .*

**Proof.** Map  $\mathbb{R}^d$  to the hyperplane  $p : x_{d+1} = 1$  in  $\mathbb{R}^{d+1}$ . This induces an embedding of the masses  $\mu_1, \dots, \mu_{dn}$ . By defining  $\mu'_i(S) = \mu(S \cap p)$  for every full-dimensional open subset of  $\mathbb{R}^{d+1}$ , we get  $dn$  mass distributions  $\mu'_1, \dots, \mu'_{dn}$  in  $\mathbb{R}^{d+1}$ . By Theorem 15, there are  $n$  hyperplanes  $\ell'_1, \dots, \ell'_n$  of dimension  $d$  through the origin that almost simultaneously bisect  $\mu'_1, \dots, \mu'_{dn}$ . Define  $\ell_i := \ell'_i \cap p$ . Note that each  $\ell_i$  is a hyperplane of dimension  $d-1$ . By the definition of  $\mu'_i$ , the hyperplanes  $\ell_1, \dots, \ell_n$  then almost simultaneously bisect  $\mu_1, \dots, \mu_{dn}$ . ◀

It remains to prove Lemma 16.

**Proof of Lemma 16.** Consider again the bundle  $\xi'_i = (E', B', \pi')$ , which is a cover of  $\xi_i$ . The set  $Z(V'_f)$  partitions  $B' \setminus Z(V'_f)$  into connected components. Consider two lifts  $b'_1, b'_2$  of a point  $b \in B$  with the property that the marked line  $\ell_i$  is oriented differently in  $b'_1$  than in  $b'_2$ . We will call a pair of such lifts *antipodal*. We claim that if  $b'_1, b'_2 \notin Z(V'_f)$  then  $b'_1$  and  $b'_2$  are not in the same connected component. If this is true, then we can assign 1 or  $-1$  to each connected component in such a way that for any antipodal pair  $b'_1, b'_2$ , whenever we assign 1 to the connected component containing  $b'_1$  we assign  $-1$  to the connected component containing





■ **Figure 5** The set  $W$  for a family of limit antipodal functions between two antipodal lifts.

$b'_2$ . We define  $s'$  as follows: for every  $b'$ , let  $d(b')$  be the distance to the boundary of its connected component (note that there are several ways to define distance measures on  $B'$ , any of them is sufficient for our purposes). Place a point at distance  $d(b')$  from the origin on the positive side of  $\ell_i$  if the connected component containing  $b'$  was assigned a 1, and on the negative side otherwise. This gives a section on  $\xi'$ . Further, for any two antipodal lifts  $b'_1, b'_2$ , we have  $s(b'_1) = -s(b'_2)$ . Also, for any two lifts  $b'_3, b'_4$ , that are not antipodal, that is,  $\ell_i$  is oriented the same way for both of them, we have  $s(b'_3) = s(b'_4)$ . Thus,  $s'$  projects to a section  $s$  in  $\xi$  with the property that  $s(b) = 0$  only if  $b \in Z(V_f)$ , which is what we want to prove.

Hence, we only need to show that a pair  $b'_1, b'_2$  of antipodal lifts is not in the same connected component. To this end, we will show that every path in  $B'$  from  $b'_1$  to  $b'_2$  crosses  $Z(V_f)$ . Let  $\gamma$  be such a path. Then  $\gamma$  induces a continuous family of limit antipodal functions  $f_t, t \in [0, 1]$ , with  $f_0 = f_{b'_1}$  and  $f_1 = f_{b'_2}$ . Further, as  $b'_1$  and  $b'_2$  are antipodal, we have  $f_0(x) = \pm f_1(-x)$ . If for any  $t$  we have  $\lim_{x \rightarrow \infty} f_t = 0$  we are done, so assume otherwise. Then, it is not possible that  $f_0(x) = f_1(-x)$ , as in this case  $\lim_{x \rightarrow \infty} f_0 = -\lim_{x \rightarrow \infty} f_1$ , so by continuity, there must be a  $t$  with  $\lim_{x \rightarrow \infty} f_t = 0$ . Thus, assume that we have  $f_0(x) = -f_1(x)$ .

The set of zeroes of the  $f_t$  defines a subset of  $\mathbb{R} \times [0, 1]$ , which we denote by  $W$ . See Figure 5 for an illustration. In general  $W$  is not connected, but has finitely many connected components, as by the second condition for limit antipodality each  $f_t$  has finitely many connected components of zeroes. We say that a connected component  $W_i$  of  $W$  has *full support* if for every  $t \in [0, 1]$ ,  $f_t$  has a zero in  $W_i$ . It can be deduced from the limit antipodality of the  $f_t$ 's that  $W$  has an odd number of connected components with full support, denoted by  $W_1, \dots, W_{2k+1}$ . Consider the median component  $W_{k+1}$ . Without loss of generality,  $W_{k+1}$  is a path in  $\mathbb{R} \times [0, 1]$  from  $(x, 0)$  to  $(-x, 1)$ . By a simple continuity argument, we see that  $W_{k+1}$  must cross the line  $(0, t), t \in [0, 1]$ . At this crossing, we are at a base point  $b' \in Z(V_f)$ , which concludes the proof. ◀

In order to prove Conjecture 5, we would like to choose  $Z(V_f)$  as the set of base points where  $f_b(0) = 0$ . Let us briefly give an example where our arguments fail for this definition. Consider  $\mu$  as the area of a unit disk in  $\mathbb{R}^2$ . If we want to simultaneously bisect  $\mu$  with two lines  $\ell_1, \ell_2$  through the origin, these lines need to be perpendicular. Further, any single line through the origin bisects  $\mu$  into two equal parts. Imagine now the line  $\ell_1$  to be fixed, and consider the limit antipodal function  $f_b$  defined by sweeping  $\ell_2$  along an oriented line

perpendicular to  $\ell_1$ . Without loss of generality, this function can be written as

$$f_b(x) = \begin{cases} 0 & x \in [-\infty, -1] \\ 1 + x & x \in [-1, 0] \\ 1 - x & x \in [0, 1] \\ 0 & x \in [1, \infty]. \end{cases}$$

Note that this holds whenever  $\ell_1$  and the sweep line for  $\ell_2$  are perpendicular, so in particular, continuously rotating the arrangement by  $180^\circ$  induces a path between two antipodal lifts in the cover. Further, along this path we never had  $f_b(0) = 0$ , so the two antipodal lifts would be in the same connected component, which would break the proof of Lemma 16 under this definition of  $Z(V_f)$ . Thus, conjecture 5 remains open for now.

---

### References

- 1 Noga Alon and Douglas B West. The Borsuk-Ulam theorem and bisection of necklaces. *Proceedings of the American Mathematical Society*, 98(4):623–628, 1986.
- 2 Luis Barba. personal communication, 2017.
- 3 Luis Barba and Patrick Schnider. Sharing a pizza: bisecting masses with two cuts. *CCCG 2017*, page 174, 2017.
- 4 Sergey Bereg, Ferran Hurtado, Mikio Kano, Matias Korman, Dolores Lara, Carlos Seara, Rodrigo I. Silveira, Jorge Urrutia, and Kevin Verbeek. Balanced partitions of 3-colored geometric sets in the plane. *Discrete Applied Mathematics*, 181:21–32, 2015. doi:10.1016/j.dam.2014.10.015.
- 5 Pavle Blagojević, Florian Frick, Albert Haase, and Günter Ziegler. Topology of the Grünbaum–Hadwiger–Ramos hyperplane mass partition problem. *Transactions of the American Mathematical Society*, 370(10):6795–6824, 2018.
- 6 Pavle VM Blagojević, Aleksandra Dimitrijević Blagojević, and Roman Karasev. More bisections by hyperplane arrangements. *arXiv preprint*, 2018. arXiv:1809.05364.
- 7 Mark De Longueville. *A course in topological combinatorics*. Springer Science & Business Media, 2012.
- 8 V.L. Dol’nikov. Transversals of families of sets in  $\mathbb{R}^n$  and a connection between the Helly and Borsuk theorems. *Russian Academy of Sciences. Sbornik Mathematics*, 184(5):111–132, 1994. URL: <http://mi.mathnet.ru/msb989>.
- 9 Zdzisław Dzedzej, Adam Idzik, and Marek Izydorek. Borsuk-Ulam type theorems on product spaces II. *Topological Methods in Nonlinear Analysis*, 14(2):345–352, 1999.
- 10 Jean-Claude Hausmann. *Mod Two Homology and Cohomology*, 2016.
- 11 Charles R. Hobby and John R. Rice. A Moment Problem in  $L_1$  Approximation. *Proceedings of the American Mathematical Society*, 16(4):665–670, 1965. URL: <http://www.jstor.org/stable/2033900>.
- 12 Alfredo Hubard and Roman Karasev. Bisecting measures with hyperplane arrangements. *arXiv preprint*, 2018. arXiv:1803.02842.
- 13 Roman N Karasev, Edgardo Roldán-Pensado, and Pablo Soberón. Measure partitions using hyperplanes with fixed directions. *Israel Journal of Mathematics*, 212(2):705–728, 2016.
- 14 Stefan Langerman. personal communication, 2017.
- 15 Jiří Matoušek. *Using the Borsuk-Ulam Theorem: Lectures on Topological Methods in Combinatorics and Geometry*. Springer Publishing Company, Incorporated, 2007.
- 16 John Milnor and James D Stasheff. *Characteristic Classes*, volume 76. Princeton university press, 2016.
- 17 Richard Rado. A Theorem on General Measure. *Journal of the London Mathematical Society*, 21:291–300, 1947.

- 18 Edgar A Ramos. Equipartition of mass distributions by hyperplanes. *Discrete & Computational Geometry*, 15(2):147–167, 1996.
- 19 Arthur H. Stone and John W. Tukey. Generalized “sandwich” theorems. *Duke Math. J.*, 9(2):356–359, June 1942. doi:10.1215/S0012-7094-42-00925-6.
- 20 Csaba D Toth, Joseph O’Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017.
- 21 Rade T. Zivaljević. User’s guide to equivariant methods in combinatorics. II. *Publications de l’Institut Mathématique. Nouvelle Série*, 64(78):107–132, 1998.
- 22 Rade T. Zivaljević and Siniša T Vrećica. An extension of the ham sandwich theorem. *Bulletin of the London Mathematical Society*, 22(2):183–186, 1990.



# Distribution-Sensitive Bounds on Relative Approximations of Geometric Ranges

Yufei Tao

Chinese University of Hong Kong, Hong Kong  
taoyf@cse.cuhk.edu.hk

Yu Wang

Chinese University of Hong Kong, Hong Kong  
yuwang@cse.cuhk.edu.hk

---

## Abstract

---

A family  $\mathcal{R}$  of ranges and a set  $X$  of points, all in  $\mathbb{R}^d$ , together define a range space  $(X, \mathcal{R}|_X)$ , where  $\mathcal{R}|_X = \{X \cap h \mid h \in \mathcal{R}\}$ . We want to find a structure to estimate the quantity  $|X \cap h|/|X|$  for any range  $h \in \mathcal{R}$  with the  $(\rho, \epsilon)$ -guarantee: (i) if  $|X \cap h|/|X| > \rho$ , the estimate must have a relative error  $\epsilon$ ; (ii) otherwise, the estimate must have an absolute error  $\rho\epsilon$ . The objective is to minimize the size of the structure. Currently, the dominant solution is to compute a relative  $(\rho, \epsilon)$ -approximation, which is a subset of  $X$  with  $\tilde{O}(\lambda/(\rho\epsilon^2))$  points, where  $\lambda$  is the VC-dimension of  $(X, \mathcal{R}|_X)$ , and  $\tilde{O}$  hides polylog factors.

This paper shows a more general bound sensitive to the content of  $X$ . We give a structure that stores  $O(\log(1/\rho))$  integers plus  $\tilde{O}(\theta \cdot (\lambda/\epsilon^2))$  points of  $X$ , where  $\theta$  – called the *disagreement coefficient* – measures how much the ranges differ from each other in their intersections with  $X$ . The value of  $\theta$  is between 1 and  $1/\rho$ , such that our space bound is never worse than that of relative  $(\rho, \epsilon)$ -approximations, but we improve the latter’s  $1/\rho$  term whenever  $\theta = o(\frac{1}{\rho \log(1/\rho)})$ . We also prove that, in the worst case, summaries with the  $(\rho, 1/2)$ -guarantee must consume  $\Omega(\theta)$  words even for  $d = 2$  and  $\lambda \leq 3$ .

We then constrain  $\mathcal{R}$  to be the set of halfspaces in  $\mathbb{R}^d$  for a constant  $d$ , and prove the existence of structures with  $o(1/(\rho\epsilon^2))$  size offering  $(\rho, \epsilon)$ -guarantees, when  $X$  is generated from various stochastic distributions. This is the first formal justification on why the term  $1/\rho$  is not compulsory for “realistic” inputs.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Relative Approximation, Disagreement Coefficient, Data Summary

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.57

**Related Version** A full version of this paper is available at <http://arxiv.org/abs/1903.06617>.

**Funding** This work was partially supported by a direct grant (Project Number: 4055079) from CUHK and by a Faculty Research Award from Google.

## 1 Introduction

A (data) *summary*, in general, refers to a structure that captures certain information up to a specified precision about a set of objects, but using space significantly smaller than the size of the set. These summaries have become important tools in algorithm design, especially in distributed/parallel computing where the main performance goal is to minimize the communication across different servers.

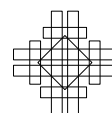
In this paper, we revisit the problem of finding a small-space summary to perform range estimation in  $\mathbb{R}^d$  with relative-error guarantees. Let  $\mathcal{R}$  be a family of geometric ranges in  $\mathbb{R}^d$  (e.g., a “halfspace family”  $\mathcal{R}$  is the set of all halfspaces in  $\mathbb{R}^d$ ), and  $X$  be a set of points in  $\mathbb{R}^d$ .  $\mathcal{R}$  and  $X$  together define a range space  $(X, \mathcal{R}|_X)$ , where  $\mathcal{R}|_X = \{X \cap h \mid h \in \mathcal{R}\}$ . Denote by  $\lambda$  the VC-dimension of  $(X, \mathcal{R}|_X)$ .



© Yufei Tao and Yu Wang;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 57; pp. 57:1–57:14



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



Following the notations of [6, 11], define

$$\overline{X}(h) = |X \cap h|/|X|$$

for each  $h \in \mathcal{R}$ , namely,  $\overline{X}(h)$  is the fraction of points in  $X$  that are covered by  $h$ . Given real-valued parameters  $0 < \rho, \epsilon < 1$ , we need to produce a structure – called a  $(\rho, \epsilon)$ -summary henceforth – that allows us to obtain, for every range  $h \in \mathcal{R}$ , a real-valued estimate  $\tau$  satisfying the following  $(\rho, \epsilon)$ -guarantee:

$$|\overline{X}(h) - \tau| \leq \epsilon \cdot \max\{\rho, \overline{X}(h)\}. \quad (1)$$

Phrased differently, the guarantee says that (i) if  $\overline{X}(h) > \rho$ ,  $\tau$  must have a relative error at most  $\epsilon$ ; (ii) otherwise,  $\tau$  must have an absolute error at most  $\rho\epsilon$ . The main challenge is to minimize the size of the structure.

## 1.1 Previous results

Throughout the paper, all logarithms have base 2 by default.

### 1.1.1 Sample-Based $(\rho, \epsilon)$ -Summaries

We say that a  $(\rho, \epsilon)$ -summary of  $(X, \mathcal{R}|_X)$  is *sample-based* if it meets the requirements below: it stores a subset  $Z \subseteq X$  such that, for any range  $h \in \mathcal{R}$  with  $Z \cap h = \emptyset$ , it returns an estimate 0 for  $\overline{X}(h)$ .

A *relative  $(\rho, \epsilon)$ -approximation* [11, 17] is a subset  $Z \subseteq X$  such that  $|\overline{X}(h) - \overline{Z}(h)| \leq \epsilon \cdot \max\{\rho, \overline{X}(h)\}$  holds for all ranges  $h \in \mathcal{R}$ . Hence, the  $(\rho, \epsilon)$ -guarantee can be fulfilled by simply setting  $\tau$  to  $\overline{Z}(h)$ , rendering  $Z$  a legal (sample-based)  $(\rho, \epsilon)$ -summary. Strengthening earlier results [3, 12, 21], Li et al. [17] proved that a random sample of  $X$  with size  $O(\frac{1}{\rho} \cdot \frac{1}{\epsilon^2} (\lambda \log \frac{1}{\rho} + \log \frac{1}{\delta}))$  is a relative  $(\rho, \epsilon)$ -approximation with probability at least  $1 - \delta$ . This implies the existence of a  $(\rho, \epsilon)$ -summary of size  $O(\frac{1}{\rho} \cdot \frac{\lambda}{\epsilon^2} \log \frac{1}{\rho})$ .

A range space  $(X, \mathcal{R}|_X)$  of a constant VC-dimension is said to be *well-behaved*, if  $\mathcal{R}|_X$  contains at most  $O(|X|) \cdot k^{O(1)}$  sets of size not exceeding  $k$ , for any integer  $k$  from 1 to  $|X|$ . Ezra [6] showed that such a range space admits a sample-based  $(\rho, \epsilon)$ -summary of size  $O(\frac{1}{\rho} \cdot \frac{1}{\epsilon^2} (\log \frac{1}{\epsilon} + \log \log \frac{1}{\rho}))$ ; note that this is smaller than the corresponding result  $O(\frac{1}{\rho} \cdot \frac{1}{\epsilon^2} \log \frac{1}{\rho})$  of [17] when  $\rho \ll \epsilon$ . It is worth mentioning that, when  $d \leq 3$  and  $\mathcal{R}$  is the halfspace family, any  $(X, \mathcal{R}|_X)$  is well-behaved; this, however, is not true when  $d \geq 4$ .

As opposed to the above “generic” bounds, Har-Peled and Sharir [11] proved specific bounds on the halfspace family  $\mathcal{R}$ . For  $d = 2$ , they showed that any  $(X, \mathcal{R})$  has a relative  $(\rho, \epsilon)$ -approximation of size  $O(\frac{1}{\rho} \cdot \frac{1}{\epsilon^{4/3}} \log^{4/3} \frac{1}{\rho\epsilon})$ ; similarly, for  $d = 3$ , the bound becomes  $O(\frac{1}{\rho} \cdot \frac{1}{\epsilon^{3/2}} \log^{3/2} \frac{1}{\rho\epsilon})$ . Combining these results and those of [6] gives the currently best bounds for these range spaces.

### 1.1.2 A Lower Bound of $\Omega(1/\rho)$

Notice that all the above bounds contain a term  $1/\rho$ . This is not a coincidence, but instead is due to a connection to “ $\epsilon$ -nets”. Given a range space  $(X, \mathcal{R}|_X)$ , an  $\epsilon$ -net [13] is a subset  $Z \subseteq X$  such that  $\overline{Z}(h) > 0$  holds for any range  $h \in \mathcal{R}$  satisfying  $\overline{X}(h) \geq \epsilon$ . As can be verified easily, any sample-based  $(\rho, 1/2)$ -summary of  $(X, \mathcal{R}|_X)$  must also be a  $\rho$ -net. This implies that the smallest size of sample-based  $(\rho, 1/2)$ -summaries must be at least that of  $\rho$ -nets.

Regarding the sizes of  $\epsilon$ -nets, a lower bound of  $\Omega(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$  is known for many range families  $\mathcal{R}$  (see [14, 15, 20] and the references therein). More precisely, this means that, for each such family  $\mathcal{R}$ , one cannot hope to obtain an  $\epsilon$ -net of size  $o(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$  for *every* possible  $X$ . It thus

follows that, for these families  $\mathcal{R}$ ,  $\Omega(\frac{1}{\rho} \log \frac{1}{\rho})$  is a lower bound on the sizes of sample-based  $(\rho, 1/2)$ -summaries. This, in turn, indicates that range spaces  $(X, \mathcal{R}|_X)$  defined by such an  $\mathcal{R}$  cannot always be well-behaved.

Coming back to the sizes of  $\epsilon$ -nets, a weaker lower bound of  $\Omega(1/\epsilon)$  holds quite commonly even on the range families that evade the  $\Omega(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$  bound. Consider, for example, the halfspace family  $\mathcal{R}$  in  $\mathbb{R}^2$ . For any  $X$ , the range space  $(X, \mathcal{R}|_X)$  definitely has an  $\epsilon$ -net of size  $O(1/\epsilon)$  [10, 19]. This is tight: place a set  $X$  of points on the boundary of a circle; and it is easy to show that any  $\epsilon$ -net of  $(X, \mathcal{R}|_X)$  must have a size of at least  $1/\epsilon$ . This means that the size of any sample-based  $(\rho, 1/2)$ -summary of  $(X, \mathcal{R}|_X)$  must be  $\Omega(1/\rho)$ .

## 1.2 Our results

### 1.2.1 On One Input: Moving Beyond $\Omega(1/\rho)$

The  $\Omega(1/\rho)$  lower bound discussed earlier holds only in the *worst case*, i.e., it is determined by the “hardest”  $X$ . For other  $X$ , the range space  $(X, \mathcal{R}|_X)$  may admit much smaller  $(\rho, \epsilon)$ -summaries. For example, let  $\mathcal{R}$  be again the set of halfplanes in  $\mathbb{R}^2$ . When all the points of  $X$  lie on a line,  $(X, \mathcal{R})$  has a sample-based  $(\rho, \epsilon)$ -approximation of size only  $O(\frac{1}{\epsilon} \log \frac{1}{\rho})$ ; also, it would be interesting to note that  $(X, \mathcal{R})$  has an  $\epsilon$ -net that contains only 2 points! In general, the existing bounds on  $(\rho, \epsilon)$ -summaries can be excessively loose on individual inputs  $X$ . This calls for an alternative, *distribution-sensitive*, analytical framework that is able to prove tighter bounds using extra complexity parameters that depend on the *content* of  $X$ .

The first contribution of this paper is to establish such a framework by resorting to the concept of *disagreement coefficient* [8] from active learning. This notion was originally defined in a context different from ours; and we will adapt it to range spaces in the next section. At this moment, it suffices to understand that the disagreement coefficient  $\theta$  is a real value satisfying:  $1 \leq \theta \leq 1/\rho$ . The coefficient quantifies the differences among the sets in  $\mathcal{R}|_X$  (a larger  $\theta$  indicates greater differences). Even under the same  $\mathcal{R}$ ,  $\theta$  may vary considerably depending on  $X$ .

We will show that, for any range space  $(X, \mathcal{R}|_X)$  of VC-dimension  $\lambda$ , there is a  $(\rho, \epsilon)$ -summary that keeps  $O(\log(1/\rho))$  integers plus

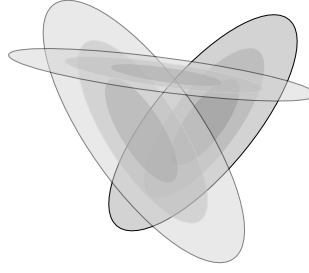
$$O\left(\min\left\{\frac{1}{\rho}, \theta \log \frac{1}{\rho}\right\} \cdot \frac{\lambda}{\epsilon^2} \log \frac{1}{\rho}\right) \quad (2)$$

points of  $X$ . The above is never worse than the general bound  $O(\frac{1}{\rho} \cdot \frac{\lambda}{\epsilon^2} \log \frac{1}{\rho})$  of relative  $(\rho, \epsilon)$ -approximations.

We will also prove that  $\Omega(\theta)$  is a lower bound on the number of words needed to encode a  $(\rho, 1/2)$ -summary even when  $d = 2$  and  $\lambda \leq 3$ . This generalizes the  $\Omega(1/\rho)$  lower bound in Section 1.1 because  $\theta$  is at most, but can reach,  $1/\rho$ . Thus, our result in (2) reflects the hardness of the input, and is tight within polylog factors for constant  $\epsilon$ . Our lower bound is information-theoretic, and does not require the summary to be sample-based.

### 1.2.2 On a Distribution of Inputs: Small Summaries for Halfspaces

Our framework allows us to explain – for the first time we believe – why  $\Omega(1/\rho) \cdot \text{poly}(1/\epsilon)$  is too pessimistic a bound on the sizes of  $(\rho, \epsilon)$ -summaries for inputs encountered in practice. For this purpose, we *must not* allow arbitrary inputs because of the prevalent  $\Omega(1/\rho)$  lower bound; instead, we will examine a class of inputs following a certain distribution.



■ **Figure 1** Mixture of 3 truncated Gaussian distributions in 2D space.

In this paper, we demonstrate the above by concentrating on the family  $\mathcal{R}$  of halfspaces in  $\mathbb{R}^d$  where the dimensionality  $d$  is a constant; this is arguably the “most-studied” family in the literature of relative  $(\rho, \epsilon)$ -approximations. The core of our solutions concerns two stochastic distributions that have drastically different behavior:

- **(Box Uniform)** Suppose that  $X$  is obtained by drawing  $n$  points uniformly at random from the *unit box*  $[0, 1]^d$ . When  $\rho = \Omega(\frac{\log n}{n})$ , we will prove that  $\theta = O(\text{polylog } \frac{1}{\rho})$  with high probability (i.e., at least  $1 - 1/n^2$ ). Accordingly, (2) becomes  $O(\frac{1}{\epsilon^2} \text{polylog } \frac{1}{\rho})$ , improving the general bound of relative  $(\rho, \epsilon)$ -approximations by almost a factor of  $O(1/\rho)$ .
- **(Ball Uniform)** Consider instead that the  $n$  points are drawn uniformly at random from the *unit ball*:  $\{x \in \mathbb{R}^d \mid \sum_{i=1}^d x[i]^2 \leq 1\}$ , where  $x[i]$  represents the  $i$ -th coordinate of point  $x$ . This time, we will prove that  $\theta = O((\frac{1}{\rho})^{\frac{d-1}{d+1}})$  with high probability for  $\rho = \Omega(\frac{\log n}{n})$ . (2) indicates the existence of a  $(\rho, \epsilon)$ -summary with size  $\tilde{O}((\frac{1}{\rho})^{\frac{d-1}{d+1}} \cdot \frac{1}{\epsilon^2})$  for  $\rho = \Omega(\frac{\log n}{n})$ , again circumventing the  $\Omega(1/\rho)$  lower bound.

The very same bounds can also be obtained in *non-uniform* settings. Suppose that  $X$  is obtained by drawing  $n$  points in an *iid* manner, according to a distribution that can be described by a probabilistic density function (pdf)  $\pi(x)$  over  $\mathbb{R}^d$  where  $d = O(1)$ . Define the *support region* of  $\pi$  as  $\text{supp}(\pi) = \{x \in \mathbb{R}^d \mid \pi(x) > 0\}$ . When  $\pi$  satisfies:

- *C1*:  $\text{supp}(\pi)$  is the unit box (or unit ball, resp.);
- *C2*: for every point  $x \in \text{supp}(\pi)$ , it holds that  $\pi(x) = \Omega(1)$ ;

we will show that  $(X, \mathcal{R}|_X)$  has a  $(\rho, \epsilon)$ -summary whose size is asymptotically the same as the aforementioned bound for box uniform (or ball uniform, resp.). Conditions *C1* and *C2* are satisfied by many distributions encountered in practice (e.g., the truncated versions of the Gaussian, Elliptical, and Laplace distributions, etc.), suggesting that real-world datasets may have much smaller  $(\rho, \epsilon)$ -summaries than previously thought.

Even better, the linearity of halfspaces implies that, the same bounds still hold even when the shape of  $\text{supp}(\pi)$  in Condition *C1* is obtained from the unit box/ball by an affine transformation. Call a distribution *atomic* if it satisfies *C1* (perhaps after an affine transformation) and *C2*. Our results hold also on “composite distributions” synthesized from a constant  $z$  number of atomic distributions whose support regions may overlap *arbitrarily*. Specifically, let  $\pi_1, \pi_2, \dots, \pi_z$  be the pdfs of atomic distributions; and define  $\pi(x) = \sum_{i=1}^z \gamma_i \cdot \pi_i(x)$ , for arbitrary positive constants  $\gamma_1, \gamma_2, \dots, \gamma_z$  that sum up to 1; see Figure 1 for an example. Then, the  $(\rho, \epsilon)$ -summary bound on  $\pi$  is asymptotically determined by the highest of the  $(\rho, \epsilon)$ -summary bounds on  $\pi_1, \dots, \pi_z$ .



## 2 Disagreement coefficients

### 2.1 Existing Definitions on Distributions

Disagreement coefficient was introduced by Hanneke [8] to analyze active learning algorithms (although a similar concept had been coined earlier [1] in statistics).

Let  $\mathcal{D}$  be a distribution over  $\mathbb{R}^d$ . For any region  $A \subseteq \mathbb{R}^d$ , we denote by  $\Pr_{\mathcal{D}}[A]$  the probability of  $x \in A$  when  $x$  is drawn from  $\mathcal{D}$ . Let  $\mathcal{R}$  be a family of geometric ranges. Given a subset  $\mathcal{R}' \subseteq \mathcal{R}$ , define the *disagreement region*  $DIS(\mathcal{R}')$  of  $\mathcal{R}'$  as

$$DIS(\mathcal{R}') = \{x \in \mathbb{R}^d \mid \exists h_1, h_2 \in \mathcal{R}' \text{ s.t. } x \in h_1 \text{ and } x \notin h_2\}.$$

That is,  $DIS(\mathcal{R}')$  includes every such point  $x \in \mathbb{R}^d$  that does not fall in all the ranges in  $\mathcal{R}'$ , and in the meantime, does not fall outside all the ranges in  $\mathcal{R}'$ , either. Given a range  $h \in \mathcal{R}$  and a real value  $r > 0$ , define its *r-ball*  $B_{\mathcal{D}}(h, r)$  as the set of all ranges  $h' \in \mathcal{R}$  satisfying  $\Pr_{\mathcal{D}}[DIS(\{h, h'\})] \leq r$ . It is worth mentioning that  $DIS(\{h, h'\})$  is simply the symmetric difference between  $h$  and  $h'$ .

Now, fix a range  $h$ , and consider increasing  $r$  continuously; this can only expand the set  $B_{\mathcal{D}}(h, r)$ , and hence, also  $DIS(B_{\mathcal{D}}(h, r))$ . Interestingly, even though  $\Pr_{\mathcal{D}}[DIS(B_{\mathcal{D}}(h, r))]$  is monotonically increasing, the ratio  $\Pr_{\mathcal{D}}[DIS(B_{\mathcal{D}}(h, r))]/r$  may remain bounded by a certain quantity. Given a real value  $\sigma \geq 0$ , the *disagreement coefficient*  $\theta_{\mathcal{D}}^h(\sigma)$  of  $h$  measures this quantity with respect to all  $r > \sigma$ :

$$\theta_{\mathcal{D}}^h(\sigma) = \max \left\{ 1, \sup_{r > \sigma} \frac{\Pr_{\mathcal{D}}[DIS(B_{\mathcal{D}}(h, r))]}{r} \right\}. \tag{3}$$

The function  $\theta_{\mathcal{D}}^h(\sigma)$  has several useful properties:

1. By definition,  $\theta_{\mathcal{D}}^h(\sigma)$  is between 1 and  $1/\sigma$ , regardless of  $\mathcal{D}$  and  $h$ .
2. The supremum in (3) ensures that  $\theta_{\mathcal{D}}^h(\sigma)$  is monotonically decreasing.
3. For any  $c \geq 1$ , it holds that  $\theta_{\mathcal{D}}^h(\sigma) \leq c \cdot \theta_{\mathcal{D}}^h(c\sigma)$  (see Corollary 7.2 of [9]).

### 2.2 New Definitions on Range Spaces

The above definitions rely on  $\mathcal{D}$ , and are not suitable for our problem settings where the input  $X$  is a finite set. Next, we present a way to adapt the definitions to a range space  $(X, \mathcal{R}|_X)$  for analyzing geometric algorithms.

We impose a uniform distribution over  $X$ : let  $\mathcal{U}(X)$  be the distribution of a random point drawn uniformly from  $X$ . By replacing  $\mathcal{D}$  with  $\mathcal{U}(X)$  in (3), we rewrite (3) into the following for any  $\sigma \geq 0$ :

$$\theta_{\mathcal{U}(X)}^h(\sigma) = \max \left\{ 1, \sup_{r > \sigma} \frac{\Pr_{\mathcal{U}(X)}[DIS(B_{\mathcal{U}(X)}(h, r))]}{r} \right\}. \tag{4}$$

Set

$$\sigma_{min} = \frac{\min_{h \in \mathcal{R}} |X \cap h|}{n} \tag{5}$$

We define the *disagreement coefficient of the range space*  $(X, \mathcal{R}|_X)$  as a function  $\theta_X(\sigma) : [\sigma_{min}, \infty) \rightarrow \mathbb{R}$  where

$$\theta_X(\sigma) = \min_{h \in \mathcal{R} \text{ s.t. } \overline{X}(h) \leq \sigma} \left\{ \theta_{\mathcal{U}(X)}^h(\sigma) \right\}. \tag{6}$$

It is clear from the above discussion that  $1 \leq \theta_X(\sigma) \leq 1/\sigma$  and  $\theta_X(\sigma)$  is monotonically decreasing.

As a remark, the finiteness of  $X$  gives a simpler interpretation of the  $r$ -ball  $B_{\mathcal{U}(X)}(h, r)$ : it is the set of ranges  $h' \in \mathcal{R}$  such that  $DIS(\{h, h'\})$  covers no more than  $r|X|$  points in  $X$ . Also,  $\Pr_{\mathcal{U}(X)}[A]$  for any region  $A \subseteq \mathbb{R}^d$  is simply  $|X \cap A|/|X|$ .

### 3 Small $(\rho, \epsilon)$ -summaries based on disagreement coefficients

Given a range space  $(X, \mathcal{R})$  with VC-dimension  $\lambda$ , we will show how to find a  $(\rho, \epsilon)$ -summary whose size can be bounded using disagreement coefficients. Our algorithm is randomized, and succeeds with probability at least  $1 - \delta$  for a real-valued parameter  $0 < \delta < 1$ . Set  $n = |X|$ . We require that  $\rho \geq \sigma_{min}$ ; otherwise, manually increasing  $\rho$  to  $\sigma_{min}$  achieves the same approximation guarantee.

#### 3.1 Algorithms

##### 3.1.1 Computing a $(\rho, \epsilon)$ -Summary

We will shrink  $\mathcal{R}$  progressively by removing a range  $h$  from  $\mathcal{R}$  once we are sure we can provide an accurate estimate for  $\bar{X}(h)$ . Define  $\mathcal{R}_0 = \mathcal{R}$ . We perform at most  $\lceil \log(1/\rho) \rceil$  rounds. Given  $\mathcal{R}_{i-1}$ , Round  $i \geq 1$  is executed as follows:

1.  $m_i \leftarrow$  the number of points  $x \in X$  such that  $x$  falls in *all* the ranges in  $\mathcal{R}_{i-1}$
2.  $X_i \leftarrow X \cap DIS(\mathcal{R}_{i-1})$
3. draw a set  $S_i$  of points uniformly at random from  $X_i$  with

$$|S_i| = O\left(\frac{|X_i|}{n} \cdot \frac{2^i}{\epsilon^2} \left(\lambda \log \frac{1}{\rho} + \log \frac{\log(1/\rho)}{\delta}\right)\right) \quad (7)$$

4.  $\mathcal{R}_i = \{h \in \mathcal{R}_{i-1} \mid \bar{S}_i(h) \cdot |X_i| + m_i < n/2^i\}$

The algorithm terminates when either  $i = \lceil \log(1/\rho) \rceil$  or  $\mathcal{R}_i = \emptyset$ . Suppose that in total  $t$  rounds are performed. The final  $(\rho, \epsilon)$ -summary consists of sets  $S_1, S_2, \dots, S_t$ , and  $2t + 1$  integers  $n, m_1, m_2, \dots, m_t, |X_1|, |X_2|, \dots, |X_t|$ .

##### 3.1.2 Performing Estimation

Given a range  $h \in \mathcal{R}$ , we deploy the summary to estimate  $\bar{X}(h)$  in two steps:

1.  $j \leftarrow$  the largest  $i \in [1, t]$  such that  $h \in \mathcal{R}_i$
2. return  $\bar{S}_j(h) \cdot \frac{|X_j|}{n} + \frac{m_j}{n}$  as the estimate

Regarding Step 1, whether  $h \in \mathcal{R}_i$  can be detected as follows. First, if  $h \notin \mathcal{R}_{i'}$  for any  $i' < i$ , then immediately  $h \notin \mathcal{R}_i$ . Otherwise, compute  $\bar{S}_i(h)$ , and declare  $h \in \mathcal{R}_i$  if and only if  $\bar{S}_i(h) \cdot |X_i| + m_i < n/2^i$ .

#### 3.2 Analysis

We now proceed to prove the correctness of our algorithms, and bound the size of the produced summary. It suffices to consider  $\epsilon \leq 1/3$  (otherwise, lower  $\epsilon$  to  $1/3$  and then apply the argument below).

The subsequent discussion is carried out under the event that, for every  $i \in [1, t]$ ,  $S_i$  is a relative  $(\rho_i, \epsilon/4)$ -approximation of  $X_i$  with respect to the ranges in  $\mathcal{R}$  where

$$\rho_i = \frac{n(1 + \epsilon)}{2^i \cdot |X_i|}.$$

By the result of [17] (reviewed in Section 1.1), with  $|S_i|$  shown in (7), the event happens with a probability at least  $1 - \delta \cdot \frac{t}{\lceil \log(1/\rho) \rceil} \geq 1 - \delta$ .

### 3.2.1 Correctness

To show that our algorithm indeed outputs a  $(\rho, \epsilon)$ -summary, we prove in the full version:

► **Lemma 1.** *The following are true for all  $i \in [1, t]$ : (i) for every range  $h \in \mathcal{R}_i$ ,  $\bar{X}(h) < (1 + \epsilon)/2^i$ ; (ii) for every range  $h \notin \mathcal{R}_i$ ,  $\bar{X}(h) \geq (1 - \epsilon)/2^i$ .*

Now consider the estimation algorithm in Section 3.1.2. Given the value  $j$  obtained at Step 1 for the input range  $h \in \mathcal{R}$ , the above lemma suggests that

$$(1 - \epsilon)/2^{j+1} \leq \bar{X}(h) < (1 + \epsilon)/2^j.$$

This, together with  $S_j$  being a  $(\rho_j, \epsilon/4)$ -approximation of  $X_j$ , ensures that our estimate satisfies the  $(\rho, \epsilon)$ -guarantee for  $h$ . The details can be found in the full version.

### 3.2.2 Bounding the Size

To bound the size of our  $(\rho, \epsilon)$ -summary, we will focus on bounding  $\sum_{i=1}^t |S_i|$ , because the rest of the summary clearly needs  $O(t) = O(\log(1/\rho))$  extra integers. Let us start with a trivial bound that follows directly from  $|X_i| \leq n$ :

$$\begin{aligned} \sum_{i=1}^t |S_i| &= O\left(\sum_{i=1}^t \frac{2^i}{\epsilon^2} \left(\lambda \log \frac{1}{\rho} + \log \frac{\log(1/\rho)}{\delta}\right)\right) \\ &= O\left(\frac{1}{\rho \epsilon^2} \left(\lambda \log \frac{1}{\rho} + \log \frac{\log(1/\rho)}{\delta}\right)\right). \end{aligned} \tag{8}$$

Next, we use disagreement coefficients to prove a tighter bound. Fix  $h \in \mathcal{R}$  to be an arbitrary range such that  $\bar{X}(h) \leq \rho$  ( $h$  definitely exists because  $\rho \geq \sigma_{min}$ ).

► **Lemma 2.**  $\mathcal{R}_i \subseteq B(h, \rho + (1 + \epsilon)/2^i)$ .

**Proof.** It suffices to prove that, for any  $h' \in \mathcal{R}_i$ ,  $\Pr_{U(X)}[DIS(\{h, h'\})] \leq \rho + (1 + \epsilon)/2^i$ , or equivalently,  $|X \cap DIS(\{h, h'\})| \leq n(\rho + (1 + \epsilon)/2^i)$ .

This holds because  $|X \cap DIS(\{h, h'\})| \leq |(X \cap h) \cup (X \cap h')|$ . By definition of  $h$ , we know  $|X \cap h| \leq n\rho$ , while By Lemma 1, we know  $|X \cap h'| \leq n(1 + \epsilon)/2^i$ . Therefore,  $|X \cap DIS(\{h, h'\})| \leq n(\rho + (1 + \epsilon)/2^i)$ . ◀

► **Lemma 3.**  $|X_i|/n \leq \theta_{U(X)}^h(2\rho) \cdot (\rho + \frac{1+\epsilon}{2^{i-1}})$ .

**Proof.** Lemma 2 tells us that  $DIS(\mathcal{R}_{i-1}) \subseteq DIS(B_{\mathcal{U}(X)}(h, \rho + \frac{1+\epsilon}{2^{i-1}}))$ . Thus:

$$\begin{aligned} |X_i|/n &= \Pr_{\mathcal{U}(X)}(DIS(\mathcal{R}_{i-1})) \\ &\leq \Pr_{\mathcal{U}(X)}\left(DIS\left(B_{\mathcal{U}(X)}\left(h, \rho + \frac{1+\epsilon}{2^{i-1}}\right)\right)\right) \\ \text{(by (4))} &\leq \theta_{\mathcal{U}(X)}^h\left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) \cdot \left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) \end{aligned}$$

By  $1/2^{i-1} > \rho$ , and the fact that  $\theta_{\mathcal{U}(X)}^h$  is monotonically decreasing, the above leads to

$$\begin{aligned} \theta_{\mathcal{U}(X)}^h\left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) \cdot \left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) &\leq \theta_{\mathcal{U}(X)}^h(\rho + \rho) \cdot \left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) \\ &= \theta_{\mathcal{U}(X)}^h(2\rho) \cdot \left(\rho + \frac{1+\epsilon}{2^{i-1}}\right). \end{aligned} \quad \blacktriangleleft$$

Therefore:

$$\begin{aligned} \sum_{i=1}^t \frac{|X_i| \cdot 2^i}{n} &\leq \theta_{\mathcal{U}(X)}^h(2\rho) \cdot \sum_{i=1}^t 2^i \cdot \left(\rho + \frac{1+\epsilon}{2^{i-1}}\right) \\ &= \theta_{\mathcal{U}(X)}^h(2\rho) \cdot \sum_{i=1}^t (2^i \cdot \rho + O(1)) \\ \text{(by } 1/2^i = \Omega(\rho)) &= \theta_{\mathcal{U}(X)}^h(2\rho) \cdot O(t) \\ &= \theta_{\mathcal{U}(X)}^h(2\rho) \cdot O(\log(1/\rho)) \\ &= \theta_{\mathcal{U}(X)}^h(\rho) \cdot O(\log(1/\rho)) \end{aligned} \quad (9)$$

where the last equality used the fact that  $\theta_{\mathcal{U}(X)}^h(2\rho) \leq 2 \cdot \theta_{\mathcal{U}(X)}^h(\rho)$ .

Remember that the above holds for *all*  $h \in \mathcal{R}$  satisfying  $\bar{X}(h) \leq \rho$ . By the definition in (6), we can improve the bound of (9) to

$$\sum_{i=1}^t \frac{|X_i| \cdot 2^i}{n} = \theta_X(\rho) \cdot O(\log(1/\rho)). \quad (10)$$

Combining the above with (7) gives  $\sum_{i=1}^t |S_i| = O(\frac{1}{\epsilon^2} \cdot \theta_X(\rho) \log(1/\rho) \cdot (\lambda \log(1/\rho) + \log \frac{\log(1/\rho)}{\delta}))$ . Putting this together with (8) and setting  $\delta$  to a constant gives:

► **Theorem 4.** *For any  $\rho \geq \sigma_{min}$  and any  $0 < \epsilon < 1$ , a range space  $(X, \mathcal{R}|_X)$  of VC-dimension  $\lambda$  has a  $(\rho, \epsilon)$ -summary which keeps  $O(\log(1/\rho))$  integers and  $O(\min\{\frac{1}{\rho}, \theta_X(\rho) \cdot \log \frac{1}{\rho}\} \cdot \frac{\lambda}{\epsilon^2} \log \frac{1}{\rho})$  points of  $X$ . Here,  $\sigma_{min}$  is defined in (5), and  $\theta_X$  is the disagreement coefficient function defined in (6).*

### 3.2.3 A Remark

Our  $(\rho, \epsilon)$ -summary is currently not sample-based, but this can be fixed by keeping – at Step 1 of the computation algorithm in Section 3.1 – an arbitrary point counted by  $m_i$ .

The  $(\rho, \epsilon)$ -summary after the fix also serves as a  $\rho$ -net. Thus, by setting  $\epsilon$  to a constant in Theorem 4, we know that for any  $\rho \geq \sigma_{min}$ , the range space  $(X, \mathcal{R}|_X)$  in Theorem 4 has an  $\rho$ -net of size  $O(\min\{\frac{1}{\rho}, \theta_X(\rho) \cdot \log \frac{1}{\rho}\} \cdot \lambda \log \frac{1}{\rho})$ . However, it should be pointed out that this bound on  $\rho$ -nets can be slightly improved, as is implied by Theorem 5.1 of [9] and made explicit in [16].

**4 Bridging distribution and finite-set disagreement coefficients**

This section will establish another theorem which will be used together with Theorem 4 to explain why we are able to obtain  $(\rho, \epsilon)$ -summarizes of  $o(1/\rho)$  size on practical datasets. Suppose that the input  $X$  has been generated by taking  $n$  points independently following the same distribution  $\mathcal{D}$  over  $\mathbb{R}^d$ . The learning literature (see, e.g., [9]) has developed a solid understanding on when the quantity  $\theta_{\mathcal{D}}^h(\sigma)$  is small. Unfortunately, those findings can rarely be applied to  $\theta_{\mathcal{U}(X)}^h(\sigma)$  because they are conditioned on requirements that must be met by  $\mathcal{D}$ , e.g., one common requirement is continuity.  $\mathcal{U}(X)$ , due to its discrete nature, seldom meets the requirements.

On the other hand, clearly  $\mathcal{U}(X)$  approximates  $\mathcal{D}$  increasingly better as  $n$  grows. Thus, we ask the question:

How large  $n$  needs to be for  $\theta_{\mathcal{U}(X)}^h(\sigma)$  to be asymptotically the same as  $\theta_{\mathcal{D}}^h(\sigma)$ ?

We partially answer the question in the next theorem:

► **Theorem 5 (The Bridging Theorem).** *Let  $\mathcal{D}$  be a distribution over  $\mathbb{R}^d$ , and  $\mathcal{R}$  be a family of ranges. Denote by  $\lambda$  the VC-dimension of the range space  $(\mathbb{R}^d, \mathcal{R})$ .*

*Fix an arbitrary range  $h \in \mathcal{R}$ , an arbitrary integer  $n$ , a real value  $0 < \delta < 1$ , a real value  $\sigma$  satisfying  $n \geq \frac{c}{\sigma}(\log \frac{n}{\delta} + \lambda \log \frac{1}{\sigma})$  for some universal constant  $c$ . If we draw a set  $X$  of  $n$  points independently from  $\mathcal{D}$ , then with probability at least  $1 - \delta$ , it holds that  $\theta_{\mathcal{U}(X)}^h(\sigma) \leq 8 \cdot \theta_{\mathcal{D}}^h(2\sigma)$ .*

The rest of the section serves as a proof of the theorem. Let us first get rid of two easy cases:

- If  $\sigma \geq 1$ ,  $\theta_{\mathcal{U}(X)}^h(\sigma) = \theta_{\mathcal{D}}^h(2\sigma) = 1$  by definition of (4); and the theorem obviously holds.
- If  $\sigma < 1/n$ , observe that every range  $h' \in B_{\mathcal{U}(X)}(h, \sigma)$  covers exactly the same set of points in  $X$  as  $h$ . Hence,  $\Pr_{\mathcal{U}(X)}[DIS(B_{\mathcal{U}(X)}(h, r))] = 0$ . It follows from (4) that  $\theta_{\mathcal{U}(X)}^h(\sigma) = 1$ . The theorem again obviously holds because  $\theta_{\mathcal{D}}^h(2\sigma) \geq 1$ , by definition.

Hence, it suffices to consider  $1/n \leq \sigma < 1$ . Define  $S = \{i/n \mid i \text{ is an integer in } [\sigma n, n]\}$ . For  $\sigma \geq 1/n$ , (4) implies

$$\theta_{\mathcal{U}(X)}^h(\sigma) \leq \max \left\{ 1, 2 \cdot \max_{r \in S} \frac{\Pr_{\mathcal{U}(X)}[DIS(B_{\mathcal{U}(X)}(h, r))]}{r} \right\}. \tag{11}$$

Consider an arbitrary  $r \in S$ . We will show that, when  $n$  satisfies the condition in the theorem, with probability at least  $1 - \delta/n$ , it holds that

$$\frac{\Pr_{\mathcal{U}(X)}[DIS(B_{\mathcal{U}(X)}(h, r))]}{r} \leq 4 \cdot \theta_{\mathcal{D}}^h(2\sigma). \tag{12}$$

Once this is done, applying the union bound on all the  $r \in S$  will prove that (11) is at most  $8 \cdot \theta_{\mathcal{D}}^h(2\sigma)$  with probability at least  $1 - \delta$ , as claimed in the theorem.

We aim to establish the following equivalent form of (12):

$$\frac{|X \cap DIS(B_{\mathcal{U}(X)}(h, r))|}{nr} \leq 4 \cdot \theta_{\mathcal{D}}^h(2\sigma). \tag{13}$$

For the above purpose, the most crucial step is to prove:

► **Lemma 6.** *When  $n \geq \frac{c_1}{r}(\lambda \log \frac{1}{r} + \log \frac{n}{\delta})$  for some universal constant  $c_1$ , it holds with probability at least  $1 - \delta/(2n)$  that  $B_{\mathcal{U}(X)}(h, r) \subseteq B_{\mathcal{D}}(h, 2r)$ .*

**Proof.** The rationale of our proof is that any  $h' \notin B_{\mathcal{D}}(h, 2r)$  is unlikely to appear in  $B_{\mathcal{U}(X)}(h, r)$  when  $n$  is large. Indeed,  $h' \notin B_{\mathcal{D}}(h, 2r)$  indicates that a point  $x$  drawn from  $D$  has probability over  $2r$  to fall in  $DIS(\{h, h'\})$ . Hence,  $|X \cap DIS(\{h, h'\})|$  should be sharply concentrated around  $2r \cdot n$ , rendering  $h' \notin B_{\mathcal{U}(X)}(h, r)$ . The challenge, however, is that there can be an infinite number of ranges  $h'$  to consider. To tackle the challenge, we need to bring down the number of ranges somehow to  $n^{O(\lambda)}$ . We achieve the purpose by observing that we can define another range space with VC-dimension  $O(\lambda)$  to capture the disagreement regions of range pairs from  $\mathcal{R}$ , as shown below.

Define  $\mathcal{R}^{dis} = \{DIS(\{h, h'\}) \mid h, h' \in \mathcal{R}\}$ . We observe that the range space  $(\mathbb{R}^d, \mathcal{R}^{dis})$  has VC-dimension  $O(\lambda)$ . To explain why, for any  $h \in \mathcal{R}$ , define  $\bar{h} = \mathbb{R}^d \setminus h$ . Accordingly, define  $\bar{\mathcal{R}} = \{\bar{h} \mid h \in \mathcal{R}\}$ . The two range spaces  $(\mathbb{R}^d, \mathcal{R})$  and  $(\mathbb{R}^d, \bar{\mathcal{R}})$  have the same VC-dimension  $\lambda$ . Therefore, the range space  $(\mathbb{R}^d, \mathcal{R} \cup \bar{\mathcal{R}})$  has VC-dimension at most  $2\lambda + 1$ . Now apply a 2-fold intersection on  $(\mathbb{R}^d, \mathcal{R} \cup \bar{\mathcal{R}})$  to create  $(\mathbb{R}^d, \mathcal{R}_1)$  where  $\mathcal{R}_1 = \{h \cap h' \mid h, h' \in \mathcal{R} \cup \bar{\mathcal{R}}\}$ . By a result of [2], the VC dimension of  $(\mathbb{R}^d, \mathcal{R}_1)$  is bounded by  $O(\lambda)$ . Finally, apply a 2-fold union on  $(\mathbb{R}^d, \mathcal{R}_1)$  to create  $(\mathbb{R}^d, \mathcal{R}_2)$  where  $\mathcal{R}_2 = \{h \cup h' \mid h, h' \in \mathcal{R}_1\}$ . By another result of [2], the VC dimension of  $(\mathbb{R}^d, \mathcal{R}_2)$  is bounded by  $O(\lambda)$ . Notice that  $\mathcal{R}^{dis}$  is a subset of  $\mathcal{R}_2$ . It thus follows that the VC-dimension of  $(\mathbb{R}^d, \mathcal{R}^{dis})$  must be  $O(\lambda)$ .

Essentially, now the task is to draw a sufficiently large set  $X$  of points from  $\mathcal{D}$  to guarantee with probability at least  $1 - \delta/(2n)$ : for every range  $h \in \mathcal{R}^{dis}$  with  $\Pr_{\mathcal{D}}(h) > 2r$ , we ensure  $|X \cap h|/|X| > r$ . By applying a result of [17] on general range spaces, we know that  $|X|$  only needs to be  $\frac{c_1}{r}(\lambda \log \frac{1}{r} + \log \frac{n}{\delta})$  for some constant  $c_1$  which does not depend on  $r, \delta$ , and  $n$ .  $\blacktriangleleft$

Set  $r' = \Pr_{\mathcal{D}}(DIS(B_{\mathcal{D}}(h, 2r)))$ ; notice that, by definition of  $\theta_{\mathcal{D}}^h(2r)$ ,  $r' \leq 2r \cdot \theta_{\mathcal{D}}^h(2r)$ . We want to draw a sufficiently large set  $X$  of points from  $\mathcal{D}$  to guarantee, with probability at least  $1 - \delta/(2n)$ ,  $|X \cap DIS(B_{\mathcal{D}}(h, 2r))| \leq 2n \cdot \max\{r, r'\}$ . By Chernoff bounds,  $n$  only needs to be at least  $\frac{c_2}{r} \log \frac{n}{\delta}$  for some universal constant  $c_2$ .

Now, set  $c = \max\{c_1, c_2\}$  and  $n = \frac{c}{r}(\lambda \log \frac{1}{r} + \log \frac{n}{\delta})$ . With probability at least  $1 - \delta/n$ , we can derive (13) from the above discussion as follows:

$$\begin{aligned} \frac{|X \cap DIS(B_{\mathcal{U}(X)}(h, r))|}{nr} &\leq \frac{|X \cap DIS(B_{\mathcal{D}}(h, 2r))|}{nr} && \text{(by Lemma 6)} \\ &\leq \frac{2n \cdot \max\{r, r'\}}{nr} = 2 \cdot \max\{1, r'/r\} \\ &\leq 2 \cdot \max\{1, 2 \cdot \theta_{\mathcal{D}}^h(2r)\} = 4 \cdot \theta_{\mathcal{D}}^h(2r) \leq 4 \cdot \theta_{\mathcal{D}}^h(2\sigma) \end{aligned}$$

where the last inequality used  $r \geq \sigma$  and the fact that  $\theta_{\mathcal{D}}^h$  is monotonically decreasing. This establishes (13) and hence completes the proof of Theorem 5.

## 5 $o(1/\rho)$ -size summaries for halfspace ranges

We are ready to explain why a set of points generated from a stochastic distribution often admits  $(\rho, \epsilon)$ -summaries of  $o(1/\rho)$  size for fixed  $\epsilon$ . This requires specializing  $\mathcal{R}$  into a concrete range family. We will do so by constraining  $\mathcal{R}$  to be the set of halfspaces in  $\mathbb{R}^d$ , because this family has received considerable attention (as reviewed in Section 1.1).

We prove in the full version the next two technical lemmas regarding the disagreement coefficients on box-uniform and ball-uniform distributions:

**► Lemma 7.** *Let  $\mathcal{U}$  be the distribution where a point is drawn uniformly at random from the unit box  $[0, 1]^d$  with  $d = O(1)$ . For any halfspace  $h$  disjoint with the box, it holds that  $\theta_{\mathcal{U}}^h(\sigma) = O(\log^{d-1} \frac{1}{\sigma})$  for all  $\sigma > 0$ .*

► **Lemma 8.** Let  $\mathcal{U}$  be the distribution where a point is drawn uniformly at random from the unit ball  $\{x \in \mathbb{R}^d \mid \sum_{i=1}^d x[i]^2 \leq 1\}$  with  $d = O(1)$ . For any halfspace  $h$  disjoint with the ball, it holds that  $\theta_{\mathcal{U}}^h(\sigma) = O((\frac{1}{\sigma})^{\frac{d-1}{d+1}})$  for all  $\sigma > 0$ .

Next, we establish our main result for *non-uniform* distributions:

► **Theorem 9.** Let  $\mathcal{R}$  be the family of halfspaces in  $\mathbb{R}^d$  with a constant dimensionality  $d$ . Let  $\mathcal{D}$  be a distribution over  $\mathbb{R}^d$  such that the pdf  $\pi$  of  $\mathcal{D}$  satisfies Conditions C1 and C2 as prescribed in Section 1.2. Suppose that we draw a set  $X$  of  $n$  points independently from  $\mathcal{D}$ . Both of the following hold with probability at least  $1 - 1/n^2$ :

- When  $\text{supp}(\pi)$  is the unit box, for any  $0 < \epsilon < 1$  and any  $\rho \geq \frac{c \log n}{n}$  where  $c > 0$  is a constant,  $X$  has a  $(\rho, \epsilon)$ -summary that keeps  $O(\log(1/\rho))$  integers and  $O(\frac{1}{\epsilon^2} \log^{d+1} \frac{1}{\rho})$  points of  $X$ .
- When  $\text{supp}(\pi)$  is the unit ball, for any  $0 < \epsilon < 1$  and any  $\rho \geq \frac{c \log n}{n}$  where  $c > 0$  is a constant,  $X$  has a  $(\rho, \epsilon)$ -summary that keeps  $O(\log(1/\rho))$  integers and  $O(\frac{1}{\epsilon^2} \cdot (\frac{1}{\rho})^{\frac{d-1}{d+2}} \cdot \log^2 \frac{1}{\rho})$  points of  $X$ .

The constant  $c$  in the above does not depend on  $\mathcal{D}$ ,  $n$ ,  $\rho$ , and  $\epsilon$ .

**Proof.** We will prove only the case where  $\text{supp}(\pi)$  is the unit box because the unit-ball case is similar. Set  $\sigma^* = \frac{c \log n}{n}$  where  $c$  is some constant to be determined later. Thanks to Theorem 4, it suffices to prove that with probability at least  $1 - 1/n^2$ ,  $\theta_X(\rho) = O(\log^{d-1} \frac{1}{\rho})$  at every  $\rho \geq \sigma^*$ . We will argue that, with probability at least  $1 - 1/n^2$ , there exists a halfspace  $h \in \mathcal{R}$  such that  $\bar{X}(h) \leq \rho$  and  $\theta_{\mathcal{U}(X)}^h(\rho) = O(\log^{d-1} \frac{1}{\rho})$ . Once this is done, we know  $\theta_X(\rho) = O(\log^{d-1} \frac{1}{\rho})$  from (6).

Condition C<sub>2</sub> says that the pdf  $\pi$  satisfies  $\pi(x) \geq \gamma$  for any point  $x$  in  $\text{supp}(\pi)$  (i.e., the unit box), where  $\gamma$  is a positive constant. Remember that, by definition of  $\text{supp}(\pi)$ ,  $\pi(x) = 0$  for any  $x$  outside  $\text{supp}(\pi)$ .

Simply set  $h$  to a halfspace as stated in Lemma 7, i.e.,  $\theta_{\mathcal{U}}^h(\sigma) = O(\log^{d-1} \frac{1}{\sigma})$ . Let  $\pi_{\mathcal{U}}$  be the pdf of  $\mathcal{U}$ :  $\pi_{\mathcal{U}}(x)$  equals 1 if  $x \in [0, 1]^d$ , or 0 otherwise. Define  $\alpha$  as any constant such that  $\alpha \leq \gamma$ . We have  $\alpha \cdot \pi_{\mathcal{U}}(x) \leq \pi(x) \leq 1 \leq \frac{1}{\alpha} \cdot \pi_{\mathcal{U}}(x)$  for all  $x \in \mathbb{R}^d$ . Given this, Theorem 7.6 of [9] tells us that  $\theta_{\mathcal{D}}^h(\sigma) = O(\theta_{\mathcal{U}}^h(\sigma/\alpha))$ . It thus follows that  $\theta_{\mathcal{D}}^h(\sigma) = O(\log^{d-1} \frac{1}{\sigma})$  for all  $\sigma > 0$ .

Now, apply Theorem 5 on  $h$  by setting  $\delta = 1/n^2$  and  $\lambda = O(1)$ . The theorem shows that, when  $n \geq \frac{\beta \log n}{\rho}$  for some constant  $\beta$ ,  $\theta_{\mathcal{U}(X)}^h(\rho) \leq 8 \cdot \theta_{\mathcal{D}}^h(\rho) = O(\log^{d-1} \frac{1}{\rho})$  with probability at least  $1 - 1/n^2$ . We set  $c \geq \beta$  to ensure  $n \geq \frac{\alpha \log n}{\rho}$ . Note also that the choice of  $h$  guarantees  $\bar{X}(h) = 0 < \rho$ . This makes  $h$  a halfspace we are looking for, and concludes the proof. ◀

Some remarks are in order:

- (Composite Distributions) Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two distributions over  $\mathbb{R}^d$  with pdfs  $\pi_1$  and  $\pi_2$ , respectively (the support regions of  $\pi_1$  and  $\pi_2$  may overlap). Define a distribution  $\mathcal{D}$  with pdf  $\pi(x) = \gamma \cdot \pi_1(x) + (1 - \gamma) \cdot \pi_2(x)$ , for some constant  $0 < \gamma < 1$ . Theorem 7.7 of [9] tells us that, for any halfspace  $h \in \mathcal{R}$  and any  $\sigma > 0$ ,  $\theta_{\mathcal{D}}^h(\sigma) \leq \theta_{\mathcal{D}_1}^h(\frac{\sigma}{\gamma}) + \theta_{\mathcal{D}_2}^h(\frac{\sigma}{1-\gamma})$ . It thus follows from Lemma 7 that, when  $\mathcal{D}_1$  and  $\mathcal{D}_2$  are atomic distributions with support regions obtainable from the unit box through affine transformations,  $\theta_{\mathcal{D}}^h(\sigma) = O(\log^{d-2} \frac{1}{\sigma})$  for any  $h$  disjoint with  $\text{supp}(\mathcal{D}_1) \cup \text{supp}(\mathcal{D}_2)$ . The unit-box result of Theorem 9 can be easily shown to hold on this  $\mathcal{D}$  as well, by adapting the proof in a straightforward manner. The same is true for the unit-ball result of Theorem 9. All these results can now be extended to a composite distribution synthesized from a constant number of atomic distributions (see Section 1.2).



- (More Distributions with Near-Constant  $\theta$ ) What is given in Lemma 7 is only one scenario where  $\theta_{\mathcal{D}}^h(\sigma)$  is nearly a constant. There are other combinations of  $\mathcal{D}$  and  $\mathcal{R}$  where  $\theta_{\mathcal{D}}^h(\sigma) = \tilde{O}(1)$  for all  $h \in \mathcal{R}$ ; see [5, 7, 8, 9, 22] (in some of those combinations,  $\mathcal{R}$  may not contain all the halfspaces in  $\mathbb{R}^d$ ; e.g., a result of [8] concerns only the halfspaces whose boundary planes pass the origin). The proof of Theorem 9 can be adapted to show that  $X$  has a  $(\rho, \epsilon)$ -summary of size  $\tilde{O}(1/\epsilon^2)$  with high probability when  $\rho = \Omega(\max\{\frac{\log n}{n}, \min_{h \in \mathcal{R}} \Pr_{\mathcal{D}}(h)\})$ .
- (Time Complexity) In general, for any  $X$ , a  $(\rho, \epsilon)$ -summary (for the halfspace family  $\mathcal{R}$  in constant-dimensional space) can be found in polynomial time even by implementing the algorithm of Section 3.1 naively. The time can be improved to  $O(n \text{ polylog } n) + n^{1-\Omega(1)} \cdot s^{O(1)}$ , where  $s$  is the size of the returned summary, by utilizing specialized data structures [4, 18].

## 6 A lower bound with disagreement coefficients

In this section, we will prove a lower bound on the sizes of  $(\rho, 1/2)$ -summaries in relation to disagreement coefficients. Our core result is:

► **Theorem 10.** *Let  $\mathcal{R}$  be the family of all halfplanes in  $\mathbb{R}^2$ . Fix any integer  $w$  as the number of bits in a word. Choose arbitrary integers  $\eta, q$ , and  $k$  such that  $\eta \geq 4$ ,  $q$  is a multiple of  $\eta$ , and  $1 \leq k \leq q/(4\eta)$ . There must exist a set  $\mathcal{C}$  of range spaces  $(X, \mathcal{R}|_X)$ , each satisfying the following conditions:*

- $X$  is a set of  $q + k$  points in  $\mathbb{R}^2$ .
- The disagreement coefficient of  $(X, \mathcal{R}|_X)$  satisfies  $\theta_X(\frac{k}{q+k}) = \frac{k+q}{k+q/\eta}$ .
- Any encoding, which encodes a  $(\frac{k}{q+k}, 1/2)$ -summary for each range space in  $\mathcal{C}$ , must use at least  $\eta \cdot w$  bits on at least one range space in  $\mathcal{C}$ .

Therefore, for  $\rho = \frac{k}{q+k}$ , if one wishes to store a  $(\rho, 1/2)$ -summary for each range space in  $\mathcal{C}$ , at least  $\eta \cdot w$  bits (namely,  $\eta$  words) are needed on at least one range space. Since  $\theta_X(\rho) = \frac{k+q}{k+q/\eta} \leq \eta$ , this establishes  $\theta_X(\rho)$  as a space lower bound for  $(\rho, 1/2)$ -summaries. In the theorem, any  $X$  has dimensionality  $d = 2$  and any  $(X, \mathcal{R})$  has VC-dimension at most 3; hence, Theorem 4 is tight up to polylog factors on constant  $\lambda$  and  $\epsilon$ .

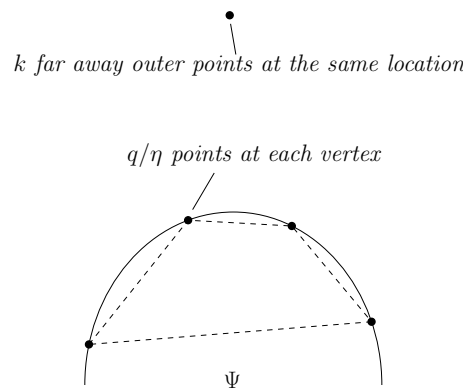
The flexibility of  $\eta, q$ , and  $k$  allows the lower bound to hold in a variety of more concrete settings. For example, by adjusting  $k$  and  $q$ , one sees that  $\theta_X(\rho)$  is a lower bound for the whole range of  $\rho \in (0, O(1)]$ . On the other hand, by focusing on any specific  $\rho \in (0, O(1)]$  but adjusting  $\eta$ , one sees that  $\theta_X(\rho)$  remains as a lower bound when  $\theta_X(\rho)$  goes from  $O(1)$  to  $\Omega(1/\rho)$ .<sup>1</sup>

**Proving Theorem 10.** Fix integers  $\eta, q$ , and  $k$  as stated in Theorem 10. Define  $n = q + k$ . Next, we construct a class  $\mathcal{X}$  of point sets, each consisting of  $n$  points in  $\mathbb{R}^2$ . First, place  $k$  points at coordinates  $(0, \infty)$ . Call them the *outer* points; they belong to all the sets in  $\mathcal{X}$ .

For each set  $X \in \mathcal{X}$ , we generate  $q$  extra *inner* points. For this purpose, place an arbitrary polygon  $\Psi$  with  $\eta$  vertices, making sure that all the vertices fall on the upper arc of the unit circle (i.e., the arc is  $\{(x[1], x[2]) \mid x[1]^2 + x[2]^2 = 1 \text{ and } x[2] \geq 0\}$ ). Then, given each vertex

<sup>1</sup> This rules out, for example, a claim of the form: “when  $\theta_X(\rho) \geq \sqrt{1/\rho}$ , there is a  $(\rho, 1/2)$ -summary of size  $O(\sqrt{\theta_X(\rho)})$ ”.





■ **Figure 2** A set of points in  $\mathcal{X}$  ( $\eta = 4$ ).

of  $\Psi$ , we add  $q/\eta$  inner points to  $X$ , all of which are located at that vertex. See Figure 2 for an example with  $\eta = 4$ . This finishes the construction of  $X$ . It is important to note that a different  $\Psi$  is used for each  $X$ . Thus,  $\mathcal{X}$  includes an infinite number of inputs, each corresponding to a possible  $\Psi$ . Our construction ensures a nice property:

► **Lemma 11.** *Fix any  $X \in \mathcal{X}$ . Given any  $(k/n, 1/2)$ -summary of  $X$ , we are able to infer all the vertices of  $\Psi$  used to construct  $X$ .*

**Proof.** We say that a halfplane in  $\mathcal{R}$  is *upward* if it covers the point  $(0, \infty)$ . Our aim is to prove that, the summary allows us to determine whether an arbitrary upward halfplane covers any inner point of  $X$ . This implies that we can reconstruct all the vertices of  $\Psi$  using the summary.<sup>2</sup>

Given an upward halfplane  $h$ , we use the summary to obtain an estimate – denoted as  $\tau$  – of  $\overline{X}(h)$ . If  $\tau \geq 2k/n$ , we return “yes” (i.e.,  $h$  covers at least one inner point); otherwise, we return “no”. To see that this is correct, first note that  $\overline{X}(h)$  must be at least  $k/n$ , and hence  $0.5\overline{X}(h) \leq \tau \leq 1.5\overline{X}(h)$ . Therefore, if  $h$  covers no inner points,  $\overline{X}(h) = k/n$ , indicating  $\tau < 1.5k/n$ . Otherwise,  $\overline{X}(h) \geq \frac{k+q/\eta}{n} \geq 5k/n$ , indicating  $\tau \geq 2.5k/n$ . ◀

We prove in the full version:

► **Lemma 12.** *For each  $X \in \mathcal{X}$ , the disagreement coefficient of  $(X, \mathcal{R}|_X)$  satisfies  $\theta_X(k/n) = \frac{n}{k+q/\eta}$ .*

The set  $\mathcal{C}$  is simply the set  $\{(X, \mathcal{R}) \mid X \in \mathcal{X}\}$ . Recall that each  $X \in \mathcal{X}$  corresponds to a distinct  $\eta$ -vertex polygon  $\Psi$ . Hence, by Lemma 11, the  $(k/n, 1/2)$ -summaries associated with the range spaces in  $\mathcal{C}$  serve as an encoding of all such  $\Psi$ 's.

So far the number of  $\Psi$ 's is infinite, which does not fit the purpose of arguing for a space lower bound in RAM with a finite word length  $w$ . This can be easily fixed by creating  $2^w$  choices for each vertex of  $\Psi$ , such that each of the  $\eta$  vertices can independently take a choice of its own. This generates  $2^{\eta w}$  polygons for  $\Psi$ , and hence, the same number of inputs in  $\mathcal{C}$ . Lemmas 11 and 12 are still valid. Therefore, any encoding, which encodes a  $(k/n, 1/2)$ -summary for each range space in  $\mathcal{C}$ , can be used to distinguish all those  $2^{\eta w}$  choices of  $\Psi$ . The encoding, therefore, must use  $\eta \cdot w$  bits for at least one range space. This completes the proof of Theorem 10. ◀

<sup>2</sup> To see this, consider any vertex  $v$  of  $\Psi$ , and use the summary to distinguish the line  $\ell$  tangent to the arc at  $v$  and a line that is parallel to  $\ell$ , but moves slightly away from the arc.

---

**References**

---

- 1 Kenneth S. Alexander. Rates of Growth and Sample Moduli for Weighted Empirical Processes Indexed by Sets. *Probability Theory and Related Fields*, 75:379–423, 1987.
- 2 Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.
- 3 Hervé Brönnimann, Bernard Chazelle, and Jiří Matoušek. Product Range Spaces, Sensitive Sampling, and Derandomization. *SIAM Journal on Computing*, 28(5):1552–1575, 1999.
- 4 Timothy M. Chan. Optimal Partition Trees. *Discrete & Computational Geometry*, 47(4):661–690, 2012.
- 5 Ran El-Yaniv and Yair Wiener. Active Learning via Perfect Selective Classification. *Journal of Machine Learning Research*, 13:255–279, 2012.
- 6 Esther Ezra. Small-size relative  $(p, \epsilon)$ -approximations for well-behaved range spaces. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 233–242, 2013.
- 7 Wayne A. Fuller. *Sampling Statistics*. Wiley, 2009.
- 8 Steve Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 353–360, 2007.
- 9 Steve Hanneke. Theory of Active Learning, 2014. Manuscript downloadable at <http://www.stevethanneke.com>.
- 10 Sariel Har-Peled, Haim Kaplan, Micha Sharir, and Shakhar Smorodinsky. Epsilon-Nets for Halfspaces Revisited. *CoRR*, abs/1410.3154, 2014. [arXiv:1410.3154](https://arxiv.org/abs/1410.3154).
- 11 Sariel Har-Peled and Micha Sharir. Relative  $(p, \epsilon)$ -Approximations in Geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.
- 12 David Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Inf. Comput.*, 100(1):78–150, 1992.
- 13 David Haussler and Emo Welzl. Epsilon-Nets and Simplex Range Queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
- 14 János Komlós, János Pach, and Gerhard J. Woeginger. Almost Tight Bounds for epsilon-Nets. *Discrete & Computational Geometry*, 7:163–173, 1992.
- 15 Andrey Kupavskii, Nabil H. Mustafa, and János Pach. New Lower Bounds for epsilon-Nets. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 54:1–54:16, 2016.
- 16 Andrey Kupavskii and Nikita Zhivotovskiy. When are epsilon-nets small? *CoRR*, abs/1711.10414, 2017. [arXiv:1711.10414](https://arxiv.org/abs/1711.10414).
- 17 Yi Li, Philip M. Long, and Aravind Srinivasan. Improved Bounds on the Sample Complexity of Learning. *Journal of Computer and System Sciences (JCSS)*, 62(3):516–527, 2001.
- 18 Jiří Matoušek. Efficient Partition Trees. *Discrete & Computational Geometry*, 8:315–334, 1992.
- 19 Jiří Matoušek, Raimund Seidel, and Emo Welzl. How to Net a Lot with Little: Small epsilon-Nets for Disks and Halfspaces. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 16–22, 1990.
- 20 János Pach and Gábor Tardos. Tight lower bounds for the size of epsilon-nets. In *Proceedings of Symposium on Computational Geometry (SoCG)*, pages 458–463, 2011.
- 21 D. Pollard. Rates of uniform almost-sure convergence for empirical processes indexed by unbounded classes of functions. *Manuscript*, 1986.
- 22 Liwei Wang. Smoothness, Disagreement Coefficient, and the Label Complexity of Agnostic Active Learning. *Journal of Machine Learning Research*, 12:2269–2292, 2011.

# DTM-Based Filtrations

## Hirokazu Anai

Fujitsu Laboratories, AI Lab, Kawasaki, Japan  
anai@jp.fujitsu.com

## Frédéric Chazal

Datashape, Inria Paris-Saclay, France  
frederic.chazal@inria.fr

## Marc Glisse

Datashape, Inria Paris-Saclay, France  
marc.glisse@inria.fr

## Yuichi Ike

Fujitsu Laboratories, AI Lab, Kawasaki, Japan  
ike.yuichi@jp.fujitsu.com

## Hiroya Inakoshi

Fujitsu Laboratories, AI Lab, Kawasaki, Japan  
hiroya.inakoshi@jp.fujitsu.com

## Raphaël Tinarrage

Datashape, Inria Paris-Saclay, France  
raphael.tinarrage@inria.fr

## Yuhei Umeda

Fujitsu Laboratories, AI Lab, Kawasaki, Japan  
umeda.yuhei@jp.fujitsu.com

---

### Abstract

Despite strong stability properties, the persistent homology of filtrations classically used in Topological Data Analysis, such as, e.g. the Čech or Vietoris–Rips filtrations, are very sensitive to the presence of outliers in the data from which they are computed. In this paper, we introduce and study a new family of filtrations, the DTM-filtrations, built on top of point clouds in the Euclidean space which are more robust to noise and outliers. The approach adopted in this work relies on the notion of distance-to-measure functions and extends some previous work on the approximation of such functions.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Topological Data Analysis, Persistent homology

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.58

**Related Version** The complete version of the paper, including proofs and additional comments, can be found at <https://arxiv.org/abs/1811.04757>.

**Funding** This work was partially supported by a collaborative research agreement between Inria and Fujitsu, and the Advanced Grant of the European Research Council GUDHI (Geometric Understanding in Higher Dimensions).

## 1 Introduction

The inference of relevant topological properties of data represented as point clouds in Euclidean spaces is a central challenge in Topological Data Analysis (TDA).

Given a (finite) set of points  $X$  in  $\mathbb{R}^d$ , persistent homology provides a now classical and powerful tool to construct persistence diagrams whose points can be interpreted as



© Hirokazu Anai, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hiroya Inakoshi, Raphaël Tinarrage, and Yuhei Umeda; licensed under Creative Commons License CC-BY

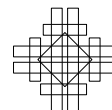
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 58; pp. 58:1–58:15

Leibniz International Proceedings in Informatics



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



homological features of  $X$  at different scales. These persistence diagrams are obtained from *filtrations*, i.e. nested families of subspaces or simplicial complexes, built on top of  $X$ . Among the many filtrations available to the user, unions of growing balls  $\cup_{x \in X} \overline{B}(x, t)$  (sublevel sets of distance functions),  $t \in \mathbb{R}^+$ , and their nerves, the Čech complex filtration, or its usually easier to compute variation, the Vietoris-Rips filtration, are widely used. The main theoretical advantage of these filtrations is that they have been shown to produce persistence diagrams that are stable with respect to perturbations of  $X$  in the Hausdorff metric [6].

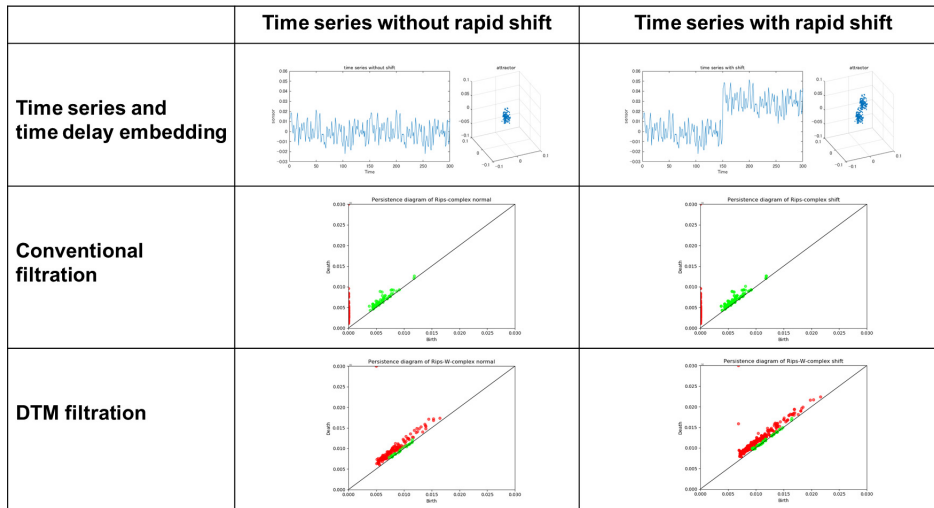
Unfortunately, the Hausdorff distance turns out to be very sensitive to noise and outliers, preventing the direct use of distance functions and classical Čech or Vietoris-Rips filtrations to infer relevant topological properties from real noisy data. Several attempts have been made in the recent years to overcome this issue. Among them, the filtration defined by the sublevel sets of the distance-to-measure (DTM) function introduced in [4], and some of its variants [10], have been proven to provide relevant information about the geometric structure underlying the data. Unfortunately, from a practical perspective, the exact computation of the sublevel sets filtration of the DTM, that boils down to the computation of a  $k$ -th order Voronoi diagram, and its persistent homology turn out to be far too expensive in most cases. To address this problem, [8] introduces a variant of the DTM function, the witnessed  $k$ -distance, whose persistence is easier to compute and proves that the witnessed  $k$ -distance approximates the DTM persistence up to a fixed additive constant. In [3, 2], a weighted version of the Vietoris-Rips complex filtration is introduced to approximate the persistence of the DTM function, and several stability and approximation results, comparable to the ones of [8], are established. Another kind of weighted Vietoris-Rips complex is presented in [1].

**Contributions.** In this paper, we introduce and study a new family of filtrations based on the notion of DTM. Our contributions are the following:

- Given a set  $X \subset \mathbb{R}^d$ , a weight function  $f$  defined on  $X$  and  $p \in [1, +\infty]$ , we introduce the weighted Čech and Rips filtrations that extend the notion of sublevel set filtration of power distances of [3]. Using classical results, we show that these filtrations are stable with respect to perturbations of  $X$  in the Hausdorff metric and perturbations of  $f$  with respect to the sup norm (Propositions 3 and 4).
- For a general function  $f$ , the stability results of the weighted Čech and Rips filtrations are not suited to deal with noisy data or data containing outliers. We consider the case where  $f$  is the empirical DTM-function associated to the input point cloud. In this case, we show an outliers-robust stability result: given two point clouds  $X, Y \subseteq \mathbb{R}^d$ , the closeness between the persistence diagrams of the resulting filtrations relies on the existence of a subset of  $X$  which is both close to  $X$  and  $Y$  in the Wasserstein metric (Theorems 15 and 20).

**Practical motivations.** Even though this aspect is not considered in this paper, it is interesting to mention that the DTM filtration was first experimented in the setting of an industrial research project whose goal was to address an anomaly detection problem from inertial sensor data in bridge and building monitoring [9]. In this problem, the input data comes as time series measuring the acceleration of devices attached to the monitored bridge/building. Using sliding windows and time-delay embedding, these time series are converted into a series of fixed size point clouds in  $\mathbb{R}^d$ . Filtrations are then built on top of these point clouds and their persistence is computed, giving rise to a time-dependent sequence of persistence diagrams that are then used to detect anomalies or specific features occurring along the time [11, 13]. In this practical setting it turned out that the DTM

filtrations reveal to be not only more resilient to noise but also able to better highlight topological features in the data than the standard Vietoris-Rips filtrations, as illustrated on a basic synthetic example on Figure 1. One of the goals of the present work is to provide theoretical foundations to these promising experimental results by studying the stability properties of the DTM filtrations.



**Figure 1** A synthetic example comparing Vietoris-Rips filtration to DTM filtration. The first row represents two time series with very different behavior and their embedding into  $\mathbb{R}^3$  (here a series  $(x_1, x_2, \dots, x_n)$  is converted in the 3D point cloud  $\{(x_1, x_2, x_3), (x_2, x_3, x_4), \dots, (x_{n-2}, x_{n-1}, x_n)\}$ ). The second row shows the persistence diagrams of the Vietoris-Rips filtration built on top of the two point clouds (red and green points represent respectively the 0-dimensional 1-dimensional diagrams); one observes that the diagrams do not clearly ‘detect’ the different behavior of the time series. The third row shows the persistence diagrams of the DTM filtration built on top of the two point clouds; a red point clearly appears away from the diagonal in the second diagram that highlights the rapid shift occurring in the second time series.

**Organisation of the paper.** Preliminary definitions, notations, and basic notions on filtrations and persistence modules are recalled in Section 2. The weighted Čech and Vietoris-Rips filtrations are introduced in Section 3, where their stability properties are established. The DTM-filtrations are introduced in Section 4. Their main stability properties are established in Theorems 15 and 20, and their relation with the sublevel set filtration of the DTM-functions is established in Proposition 16.

The various illustrations and experiments of this paper have been computed with the GUDHI library on Python [14].

For the complete version of this paper, including proofs and additional comments, see the online version at <https://arxiv.org/abs/1811.04757>.

## 2 Filtrations and interleaving distance

In the sequel, we consider interleavings of filtrations, interleavings of persistence modules and their associated pseudo-distances. Their definitions, restricted to the setting of the paper, are briefly recalled in this section.

Let  $T = \mathbb{R}^+$  and  $E = \mathbb{R}^d$  endowed with the standard Euclidean norm.

**Filtrations of sets and simplicial complexes.** A family of subsets  $(V^t)_{t \in T}$  of  $E = \mathbb{R}^d$  is a *filtration* if it is non-decreasing for the inclusion, i.e. for any  $s, t \in T$ , if  $s \leq t$  then  $V^s \subseteq V^t$ . Given  $\epsilon \geq 0$ , two filtrations  $(V^t)_{t \in T}$  and  $(W^t)_{t \in T}$  of  $E$  are  $\epsilon$ -*interleaved* if, for every  $t \in T$ ,  $V^t \subseteq W^{t+\epsilon}$  and  $W^t \subseteq V^{t+\epsilon}$ . The interleaving pseudo-distance between  $(V^t)_{t \in T}$  and  $(W^t)_{t \in T}$  is defined as the infimum of such  $\epsilon$ :

$$d_i((V^t)_{t \in T}, (W^t)_{t \in T}) = \inf\{\epsilon : (V^t) \text{ and } (W^t) \text{ are } \epsilon\text{-interleaved}\}.$$

Filtrations of simplicial complexes and their interleaving distance are similarly defined: given a set  $X$  and an abstract simplex  $S$  with vertex set  $X$ , a *filtration of  $S$*  is a non-decreasing family  $(S^t)_{t \in T}$  of subcomplexes of  $S$ . The interleaving pseudo-distance between two filtrations  $(S_1^t)_{t \in T}$  and  $(S_2^t)_{t \in T}$  of  $S$  is the infimum of the  $\epsilon \geq 0$  such that they are  $\epsilon$ -interleaved, i.e. for any  $t \in T$ ,  $S_1^t \subseteq S_2^{t+\epsilon}$  and  $S_2^t \subseteq S_1^{t+\epsilon}$ .

Notice that the interleaving distance is only a pseudo-distance, as two distinct filtrations may have zero interleaving distance.

**Persistence modules.** Let  $k$  be a field. A *persistence module*  $\mathbb{V}$  over  $T = \mathbb{R}^+$  is a pair  $\mathbb{V} = ((\mathbb{V}^t)_{t \in T}, (v_s^t)_{s \leq t \in T})$  where  $(\mathbb{V}^t)_{t \in T}$  is a family of  $k$ -vector spaces, and  $(v_s^t : \mathbb{V}^s \rightarrow \mathbb{V}^t)_{s \leq t \in T}$  a family of linear maps such that:

- for every  $t \in T$ ,  $v_t^t : \mathbb{V}^t \rightarrow \mathbb{V}^t$  is the identity map,
- for every  $r, s, t \in T$  such that  $r \leq s \leq t$ ,  $v_s^t \circ v_r^s = v_r^t$ .

Given  $\epsilon \geq 0$ , an  $\epsilon$ -*morphism* between two persistence modules  $\mathbb{V}$  and  $\mathbb{W}$  is a family of linear maps  $(\phi_t : \mathbb{V}^t \rightarrow \mathbb{W}^{t+\epsilon})_{t \in T}$  such that the following diagrams commute for every  $s \leq t \in T$ :

$$\begin{array}{ccc} \mathbb{V}^s & \xrightarrow{v_s^t} & \mathbb{V}^t \\ \downarrow \phi_s & & \downarrow \phi_t \\ \mathbb{W}^{s+\epsilon} & \xrightarrow{w_{s+\epsilon}^{t+\epsilon}} & \mathbb{W}^{t+\epsilon} \end{array}$$

If  $\epsilon = 0$  and each  $\phi_t$  is an isomorphism, the family  $(\phi_t)_{t \in T}$  is said to be an *isomorphism* of persistence modules.

An  $\epsilon$ -*interleaving* between two persistence modules  $\mathbb{V}$  and  $\mathbb{W}$  is a pair of  $\epsilon$ -morphisms  $(\phi_t : \mathbb{V}^t \rightarrow \mathbb{W}^{t+\epsilon})_{t \in T}$  and  $(\psi_t : \mathbb{W}^t \rightarrow \mathbb{V}^{t+\epsilon})_{t \in T}$  such that the following diagrams commute for every  $t \in T$ :

$$\begin{array}{ccc} \mathbb{V}^t & \xrightarrow{v_t^{t+2\epsilon}} & \mathbb{V}^{t+2\epsilon} \\ \searrow \phi_t & & \nearrow \psi_{t+\epsilon} \\ & \mathbb{W}^{t+\epsilon} & \end{array} \qquad \begin{array}{ccc} & \mathbb{V}^{t+\epsilon} & \\ \nearrow \psi_t & & \searrow \phi_{t+\epsilon} \\ \mathbb{W}^t & \xrightarrow{w_t^{t+2\epsilon}} & \mathbb{W}^{t+2\epsilon} \end{array}$$

The interleaving pseudo-distance between  $\mathbb{V}$  and  $\mathbb{W}$  is defined as

$$d_i(\mathbb{V}, \mathbb{W}) = \inf\{\epsilon \geq 0, \mathbb{V} \text{ and } \mathbb{W} \text{ are } \epsilon\text{-interleaved}\}.$$

In some cases, the proximity between persistence modules is expressed with a function. Let  $\eta : T \rightarrow T$  be a non-decreasing function such that for any  $t \in T$ ,  $\eta(t) \geq t$ . A  $\eta$ -*interleaving* between two persistence modules  $\mathbb{V}$  and  $\mathbb{W}$  is a pair of families of linear maps  $(\phi_t : \mathbb{V}^t \rightarrow \mathbb{W}^{\eta(t)})_{t \in T}$  and  $(\psi_t : \mathbb{W}^t \rightarrow \mathbb{V}^{\eta(t)})_{t \in T}$  such that the following diagrams commute for every  $t \in T$ :

$$\begin{array}{ccc} \mathbb{V}^t & \xrightarrow{v_t^{\eta(t)}} & \mathbb{V}^{\eta(t)} \\ \searrow \phi_t & & \nearrow \psi_{\eta(t)} \\ & \mathbb{W}^{\eta(t)} & \end{array} \qquad \begin{array}{ccc} & \mathbb{V}^{\eta(t)} & \\ \nearrow \psi_t & & \searrow \phi_{\eta(t)} \\ \mathbb{W}^t & \xrightarrow{v_t^{\eta(t)}} & \mathbb{W}^{\eta(t)} \end{array}$$

When  $\eta$  is  $t \mapsto t + c$  for some  $c \geq 0$ , it is called an additive  $c$ -interleaving and corresponds with the previous definition. When  $\eta$  is  $t \mapsto ct$  for some  $c \geq 1$ , it is called a multiplicative  $c$ -interleaving.

A persistent module  $\mathbb{V}$  is said to be  $q$ -tame if for every  $s, t \in T$  such that  $s < t$ , the map  $v_s^t$  is of finite rank. The  $q$ -tameness of a persistence module ensures that we can define a notion of persistence diagram – see [5]. Moreover, given two  $q$ -tame persistence modules  $\mathbb{V}, \mathbb{W}$  with persistence diagrams  $D(\mathbb{V}), D(\mathbb{W})$ , the so-called isometry theorem states that  $d_b(D(\mathbb{V}), D(\mathbb{W})) = d_i(\mathbb{V}, \mathbb{W})$  ([5, Theorem 4.11]) where  $d_b(\cdot, \cdot)$  denotes the bottleneck distance between diagrams.

**Relation between filtrations and persistence modules.** Applying the homology functor to a filtration gives rise to a persistence module where the linear maps between homology groups are induced by the inclusion maps between sets (or simplicial complexes). As a consequence, if two filtrations are  $\epsilon$ -interleaved then their associated homology persistence modules are also  $\epsilon$ -interleaved, the interleaving homomorphisms being induced by the interleaving inclusion maps. Moreover, if the modules are  $q$ -tame, then the bottleneck distance between their persistence diagrams is upperbounded by  $\epsilon$ .

The filtrations considered in this paper are obtained as union of growing balls. Their associated persistence module is the same as the persistence module of a filtered simplicial complex via the persistent nerve lemma ([7], Lemma 3.4). Indeed, consider a filtration  $(V^t)_{t \in T}$  of  $E$  and assume that there exists a family of points  $(x_i)_{i \in I} \in E^I$  and a family of non-decreasing functions  $r_i : T \rightarrow \mathbb{R}^+ \cup \{-\infty\}$ ,  $i \in I$ , such that, for every  $t \in T$ ,  $V^t$  is equal to the union of closed balls  $\bigcup_I \bar{B}(x_i, r_i(t))$ , with the convention  $\bar{B}(x_i, -\infty) = \emptyset$ . For every  $t \in T$ , let  $\mathcal{V}^t$  denote the cover  $\{\bar{B}(x_i, r_i(t)), i \in I\}$  of  $V^t$ , and  $S^t$  be its nerve. Let  $\mathbb{V}$  be the persistence module associated with the filtration  $(V^t)_{t \in T}$ , and  $\mathbb{V}_{\mathcal{N}}$  the one associated with the simplicial filtration  $(S^t)_{t \in T}$ . Then  $\mathbb{V}$  and  $\mathbb{V}_{\mathcal{N}}$  are isomorphic persistence modules. In particular, if  $\mathbb{V}$  is  $q$ -tame,  $\mathbb{V}$  and  $\mathbb{V}_{\mathcal{N}}$  have the same persistence diagrams.

### 3 Weighted Čech filtrations

In order to define the DTM-filtrations, we go through an intermediate and more general construction, namely the weighted Čech filtrations. It generalizes the usual notion of Čech filtration of a subset of  $\mathbb{R}^d$ , and shares comparable regularity properties.

#### 3.1 Definition

In the sequel of the paper, the Euclidean space  $E = \mathbb{R}^d$ , the index set  $T = \mathbb{R}^+$  and a real number  $p \geq 1$  are fixed. Consider  $X \subseteq E$  and  $f : X \rightarrow \mathbb{R}^+$ . For every  $x \in X$  and  $t \in T$ , we define

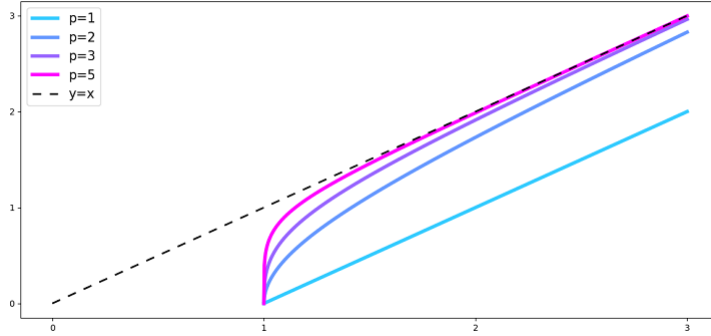
$$r_x(t) = \begin{cases} -\infty & \text{if } t < f(x), \\ (t^p - f(x)^p)^{\frac{1}{p}} & \text{otherwise.} \end{cases}$$

We denote by  $\bar{B}_f(x, t) = \bar{B}(x, r_x(t))$  the closed Euclidean ball of center  $x$  and radius  $r_x(t)$ . By convention, a Euclidean ball of radius  $-\infty$  is the empty set. For  $p = \infty$ , we also define

$$r_x(t) = \begin{cases} -\infty & \text{if } t < f(x), \\ t & \text{otherwise,} \end{cases}$$

and the balls  $\bar{B}_f(x, t) = \bar{B}(x, r_x(t))$ . Some of these radius functions are represented in Figure 2.





■ **Figure 2** Graph of  $t \mapsto r_x(t)$  for  $f(x) = 1$  and several values of  $p$ .

► **Definition 1.** Let  $X \subseteq E$  and  $f : X \rightarrow \mathbb{R}^+$ . For every  $t \in T$ , we define the following set:

$$V^t[X, f] = \bigcup_{x \in X} \overline{B}_f(x, t).$$

The family  $V[X, f] = (V^t[X, f])_{t \geq 0}$  is a filtration of  $E$ . It is called the weighted Čech filtration with parameters  $(X, f, p)$ . We denote by  $\mathbb{V}[X, f]$  its persistent (singular) homology module.

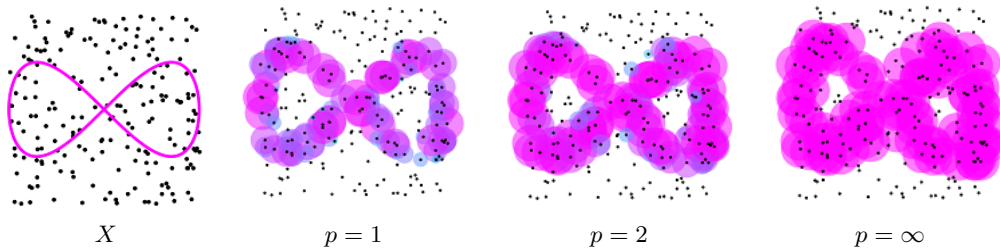
Note that  $V[X, f]$  and  $\mathbb{V}[X, f]$  depend on fixed parameter  $p$ , that is not made explicit in the notation.

Introduce  $\mathcal{V}^t[X, f] = \{\overline{B}_f(x, t)\}_{x \in X}$ . It is a cover of  $V^t[X, f]$  by closed Euclidean balls. Let  $\mathcal{N}(\mathcal{V}^t[X, f])$  be the nerve of the cover  $\mathcal{V}^t[X, f]$ . It is a simplicial complex over the vertex set  $X$ . The family  $\mathcal{N}(V[X, f]) = (\mathcal{N}(\mathcal{V}^t[X, f]))_{t \geq 0}$  is a filtered simplicial complex. We denote by  $\mathbb{V}_{\mathcal{N}}[X, f]$  its persistent (simplicial) homology module. As a consequence of the persistent nerve theorem [7, Lemma 3.4],  $\mathbb{V}[X, f]$  and  $\mathbb{V}_{\mathcal{N}}[X, f]$  are isomorphic persistent modules.

When  $f = 0$ ,  $V[X, f]$  does not depend on  $p \geq 1$ , and it is the filtration of  $E$  by the sublevel sets of the distance function to  $X$ . In the sequel, we denote it by  $V[X, 0]$ . The corresponding filtered simplicial complex,  $\mathcal{N}(V[X, 0])$ , is known as the usual Čech complex of  $X$ .

When  $p = 2$ , the filtration value of  $y \in E$ , i.e. the infimum of the  $t$  such that  $y \in V^t[X, f]$ , is called the power distance of  $y$  associated to the weighted set  $(X, f)$  in [3, Definition 4.1]. The filtration  $V[X, f]$  is called the weighted Čech filtration ([3, Definition 5.1]).

**Example.** Consider the point cloud  $X$  drawn on the left of Figure 3 (black). It is a 200-sample of the uniform distribution on  $[-1, 1]^2 \subseteq \mathbb{R}^2$ . We choose  $f$  to be the distance function to the lemniscate of Bernoulli (magenta). Let  $t = 0.2$ . Figure 3 represents the sets  $V^t[X, f]$  for several values of  $p$ . The balls are colored differently according to their radius.



■ **Figure 3** The set  $X$  and the sets  $V^t[X, f]$  for  $t = 0.2$  and several values of  $p$ .



The following proposition states the regularity of the persistent module  $\mathbb{V}[X, f]$ .

► **Proposition 2.** *If  $X \subseteq E$  is finite and  $f$  is any function, then  $\mathbb{V}[X, f]$  is a pointwise finite-dimensional persistence module.*

*More generally, if  $X$  is a bounded subset of  $E$  and  $f$  is any function, then  $\mathbb{V}[X, f]$  is  $q$ -tame.*

### 3.2 Stability

We still consider a subset  $X \subseteq E$  and a function  $f : X \rightarrow \mathbb{R}^+$ . Using the fact that two  $\epsilon$ -interleaved filtrations induce  $\epsilon$ -interleaved persistence modules, the stability results for the filtration  $V[X, f]$  of this subsection immediately translate as stability results for the persistence module  $\mathbb{V}[X, f]$ .

The following proposition relates the stability of the filtration  $V[X, f]$  with respect to  $f$ .

► **Proposition 3.** *Let  $g : X \rightarrow \mathbb{R}^+$  be a function such that  $\sup_{x \in X} |f(x) - g(x)| \leq \epsilon$ . Then the filtrations  $V[X, f]$  and  $V[X, g]$  are  $\epsilon$ -interleaved.*

The following proposition states the stability of  $V[X, f]$  with respect to  $X$ . It generalizes [3, Proposition 4.3] (case  $p = 2$ ).

► **Proposition 4.** *Let  $Y \subseteq E$  and suppose that  $f : X \cup Y \rightarrow \mathbb{R}^+$  is  $c$ -Lipschitz,  $c \geq 0$ . Suppose that  $X$  and  $Y$  are compact and that the Hausdorff distance  $d_H(X, Y) \leq \epsilon$ . Then the filtrations  $V[X, f]$  and  $V[Y, f]$  are  $k$ -interleaved with  $k = \epsilon(1 + c^p)^{\frac{1}{p}}$ .*

One can show that the bounds in Proposition 3 and 4 are tight.

When considering data with outliers, the observed set  $X$  may be very distant from the underlying signal  $Y$  in Hausdorff distance. Therefore, the tight bound in Proposition 4 may be unsatisfactory. Moreover, a usual choice of  $f$  would be  $d_X$ , the distance function to  $X$ . But the bound in Proposition 3 then becomes  $\|d_X - d_Y\|_\infty = d_H(X, Y)$ . We address this issue in Section 4 by considering an outliers-robust function  $f$ , the so-called distance-to-measure function (DTM).

### 3.3 Weighted Vietoris-Rips filtrations

Rather than computing the persistence of the Čech filtration of a point cloud  $X \subseteq E$ , one sometimes consider the corresponding Vietoris-Rips filtration, which is usually easier to compute.

If  $G$  is a graph with vertex set  $X$ , its corresponding clique complex is the simplicial complex over  $X$  consisting of the sets of vertices of cliques of  $G$ . If  $S$  is a simplicial complex, its corresponding flag complex is the clique complex of its 1-skeleton.

Recall that  $\mathcal{N}(\mathcal{V}^t[X, f])$  denotes the nerve of  $\mathcal{V}^t[X, f]$ , where  $\mathcal{V}^t[X, f]$  is the cover  $\{\bar{B}_f(x, t)\}_{x \in X}$  of  $V^t[X, f]$ .

► **Definition 5.** *We denote by  $\text{Rips}(\mathcal{V}^t[X, f])$  the flag complex of  $\mathcal{N}(\mathcal{V}^t[X, f])$ , and by  $\text{Rips}(\mathcal{V}[X, f])$  the corresponding filtered simplicial complex. It is called the weighted Rips complex with parameters  $(X, f, p)$ .*

The following proposition states that the filtered simplicial complexes  $\mathcal{N}(\mathcal{V}[X, f])$  and  $\text{Rips}(\mathcal{V}[X, f])$  are 2-interleaved multiplicatively, generalizing the classical case of the Čech and Vietoris-Rips filtrations (case  $f = 0$ ).

► **Proposition 6.** For every  $t \geq 0$ ,

$$\mathcal{N}(\mathcal{V}^t[X, f]) \subseteq \text{Rips}(\mathcal{V}^t[X, f]) \subseteq \mathcal{N}(\mathcal{V}^{2t}[X, f])$$

Using Theorem 3.1 of [1], the multiplicative interleaving  $\text{Rips}(\mathcal{V}^t[X, f]) \subseteq \mathcal{N}(\mathcal{V}^{2t}[X, f])$  can be improved to  $\text{Rips}(\mathcal{V}^t[X, f]) \subseteq \mathcal{N}(\mathcal{V}^{ct}[X, f])$ , where  $c = \sqrt{\frac{2d}{d+1}}$  and  $d$  is the dimension of the ambient space  $E = \mathbb{R}^d$ .

Note that weighted Rips filtration shares the same stability properties as the weighted Čech filtration. Indeed, the proofs of Proposition 3 and 4 immediately extend to this case.

In order to compute the flag complex  $\text{Rips}(\mathcal{V}^t[X, f])$ , it is enough to know the filtration values of its 0- and 1-simplices. The following proposition describes these values.

► **Proposition 7.** Let  $p < +\infty$ . The filtration value of a vertex  $x \in X$  is given by  $t_X(\{x\}) = f(x)$ .

The filtration value of an edge  $\{x, y\} \subseteq E$  is given by

$$t_X(\{x, y\}) = \begin{cases} \max\{f(x), f(y)\} & \text{if } \|x - y\| \leq |f(x)^p - f(y)^p|^{\frac{1}{p}}, \\ t & \text{otherwise,} \end{cases}$$

where  $t$  is the only positive root of

$$\|x - y\| = (t^p - f(x)^p)^{\frac{1}{p}} + (t^p - f(y)^p)^{\frac{1}{p}} \quad (1)$$

When  $\|x - y\| \geq |f(x)^p - f(y)^p|^{\frac{1}{p}}$ , the positive root of Equation (1) does not always admit a closed form. We give some particular cases for which it can be computed.

- For  $p = 1$ , the root is  $t_X(\{x, y\}) = \frac{f(x) + f(y) + \|x - y\|}{2}$ ,
- for  $p = 2$ , it is  $t_X(\{x, y\}) = \frac{\sqrt{((f(x) + f(y))^2 + \|x - y\|^2)((f(x) - f(y))^2 + \|x - y\|^2)}}{2\|x - y\|}$ ,
- for  $p = \infty$ , the condition reads  $\|x - y\| \geq \max\{f(x), f(y)\}$ , and the root is  $t_X(\{x, y\}) = \frac{\|x - y\|}{2}$ . In either case,  $t_X(\{x, y\}) = \max\{f(x), f(y), \frac{\|x - y\|}{2}\}$ .

We close this subsection by discussing the influence of  $p$  on the weighted Čech and Rips filtrations. Let  $D_0(\mathcal{N}(\mathcal{V}[X, f, p]))$  be the persistence diagram of the 0th-homology of  $\mathcal{N}(\mathcal{V}[X, f, p])$ . We say that a point  $(b, d)$  of  $D_0(\mathcal{V}[X, f, p])$  is non-trivial if  $b \neq d$ . Let  $D_0(\text{Rips}(\mathcal{V}[X, f, p]))$  be the persistence diagram of the 0th-homology of  $\text{Rips}(\mathcal{V}[X, f, p])$ . Note that  $D_0(\mathcal{N}(\mathcal{V}[X, f, p])) = D_0(\text{Rips}(\mathcal{V}[X, f, p]))$  since the corresponding filtrations share the same 1-skeleton.

► **Proposition 8.** The number of non-trivial points in  $D_0(\text{Rips}(\mathcal{V}[X, f, p]))$  is non-increasing with respect to  $p \in [1, +\infty)$ . The same holds for  $D_0(\mathcal{N}(\mathcal{V}[X, f, p]))$ .

Figure 7 in Subsection 4.4 illustrates the previous proposition in the case of the DTM-filtrations. Greater values of  $p$  lead to sparser 0th-homology diagrams.

Now, consider  $k > 0$ , and let  $D_k(\mathcal{N}(\mathcal{V}[X, f, p]))$  be the persistence diagram of the  $k$ th-homology of  $\mathcal{N}(\mathcal{V}[X, f, p])$ . In this case, one can easily build examples showing that the number of non-trivial points of  $D_k(\mathcal{N}(\mathcal{V}[X, f, p]))$  does not have to be non-increasing with respect to  $p$ . The same holds for  $D_k(\text{Rips}(\mathcal{V}[X, f, p]))$ .

**4 DTM-filtrations**

The results of previous section suggest that in order to construct a weighted Čech filtration  $V[X, f]$  that is robust to outliers, it is necessary to choose a function  $f$  that depends on  $X$  and that is itself robust to outliers. The so-called distance-to-measure function (DTM) satisfies such properties, motivating the introduction of the DTM-filtrations in this section.

**4.1 The distance to measure (DTM)**

Let  $\mu$  be a probability measure over  $E = \mathbb{R}^d$ , and  $m \in [0, 1]$  a parameter. For every  $x \in \mathbb{R}^d$ , let  $\delta_{\mu,m}$  be the function defined on  $E$  by  $\delta_{\mu,m}(x) = \inf\{r \geq 0, \mu(\overline{B}(x, r)) > m\}$ .

► **Definition 9.** *Let  $m \in [0, 1[$ . The DTM  $\mu$  of parameter  $m$  is the function:*

$$d_{\mu,m} : E \rightarrow \mathbb{R}$$

$$x \mapsto \sqrt{\frac{1}{m} \int_0^m \delta_{\mu,t}^2(x) dt}$$

When  $m$  is fixed – which is the case in the following subsections – and when there is no risk of confusion, we write  $d_\mu$  instead of  $d_{\mu,m}$ .

Notice that when  $m = 0$ ,  $d_{\mu,m}$  is the distance function to  $\text{supp}(\mu)$ , the support of  $\mu$ .

► **Proposition 10** ([4], Corollary 3.7). *For every probability measure  $\mu$  and  $m \in [0, 1)$ ,  $d_{\mu,m}$  is 1-Lipschitz.*

A fundamental property of the DTM is its stability with respect to the probability measure  $\mu$  in the Wasserstein metric. Recall that given two probability measures  $\mu$  and  $\nu$  over  $E$ , a transport plan between  $\mu$  and  $\nu$  is a probability measure  $\pi$  over  $E \times E$  whose marginals are  $\mu$  and  $\nu$ . The Wasserstein distance with quadratic cost between  $\mu$  and  $\nu$  is defined as  $W_2(\mu, \nu) = \left( \inf_{\pi} \int_{E \times E} \|x - y\|^2 d\pi(x, y) \right)^{\frac{1}{2}}$ , where the infimum is taken over all the transport plans  $\pi$ . When  $\mu = \mu_X$  and  $\nu = \mu_Y$  are the empirical measures of the finite point clouds  $X$  and  $Y$ , i.e the normalized sums of the Dirac measures on the points of  $X$  and  $Y$  respectively, we write  $W_2(X, Y)$  instead of  $W_2(\mu_X, \mu_Y)$ .

► **Proposition 11** ([4], Theorem 3.5). *Let  $\mu, \nu$  be two probability measures, and  $m \in (0, 1)$ . Then*

$$\|d_{\mu,m} - d_{\nu,m}\|_{\infty} \leq m^{-\frac{1}{2}} W_2(\mu, \nu).$$

Notice that for every  $x \in E$ ,  $d_\mu(x)$  is not lower than the distance from  $x$  to  $\text{supp}(\mu)$ , the support of  $\mu$ . This remark, along with the propositions 10 and 11, are the only properties of the DTM that will be used to prove the results in the sequel of the paper.

In practice, the DTM can be computed. If  $X$  is a finite subset of  $E$  of cardinal  $n$ , we denote by  $\mu_X$  its empirical measure. Assume that  $m = \frac{k_0}{n}$ , with  $k_0$  an integer. In this case,  $d_{\mu_X,m}$  reformulates as follows: for every  $x \in E$ ,

$$d_{\mu_X,m}^2(x) = \frac{1}{k_0} \sum_{k=1}^{k_0} \|x - p_k(x)\|^2,$$

where  $p_1(x), \dots, p_{k_0}(x)$  are a choice of  $k_0$ -nearest neighbors of  $x$  in  $X$ .

### 4.2 DTM-filtrations

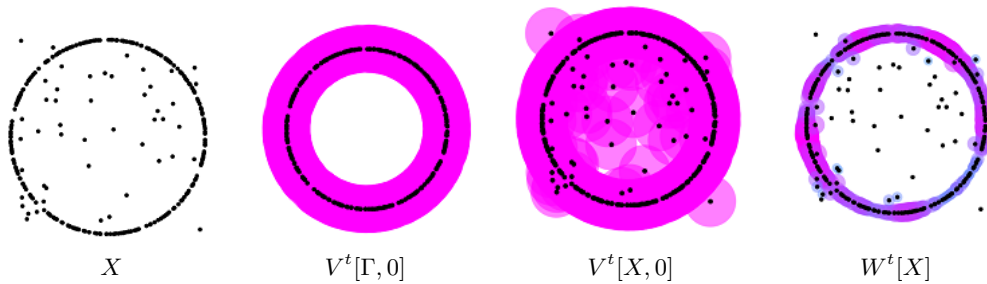
In the following, the two parameters  $p \in [1, +\infty]$  and  $m \in (0, 1)$  are fixed.

► **Definition 12.** Let  $X \subseteq E$  be a finite point cloud,  $\mu_X$  the empirical measure of  $X$ , and  $d_{\mu_X}$  the corresponding DTM of parameter  $m$ . The weighted Čech filtration  $V[X, d_{\mu_X}]$ , as defined in Definition 1, is called the DTM-filtration associated with the parameters  $(X, m, p)$ . It is denoted by  $W[X]$ . The corresponding persistence module is denoted by  $\mathbb{W}[X]$ .

Let  $\mathcal{W}^t[X] = \mathcal{V}^t[X, d_{\mu_X}]$  denote the cover of  $W^t[X]$  as defined in section 3, and let  $\mathcal{N}(\mathcal{W}^t[X])$  be its nerve. The family  $\mathcal{N}(\mathcal{W}[X]) = (\mathcal{N}(\mathcal{W}^t[X]))_{t \geq 0}$  is a filtered simplicial complex, and its persistent (simplicial) homology module is denoted by  $\mathbb{W}_{\mathcal{N}}[X]$ . By the persistent nerve lemma, the persistence modules  $\mathbb{W}[X]$  and  $\mathbb{W}_{\mathcal{N}}[X]$  are isomorphic.

As in Definition 5,  $\text{Rips}(\mathcal{W}^t[X])$  denotes the flag complex of  $\mathcal{N}(\mathcal{W}^t[X])$ , and  $\text{Rips}(\mathcal{W}[X])$  the corresponding filtered simplicial complex.

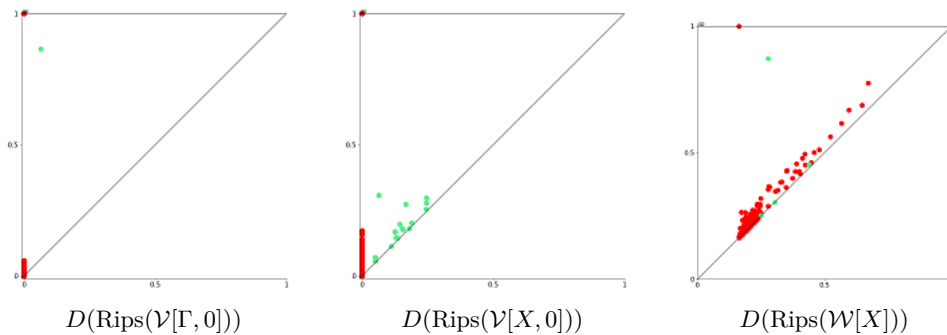
**Example.** Consider the point cloud  $X$  drawn on the left of Figure 4. It is the union of  $\tilde{X}$  and  $\Gamma$ , where  $\tilde{X}$  is a 50-sample of the uniform distribution on  $[-1, 1]^2 \subseteq \mathbb{R}^2$ , and  $\Gamma$  is a 300-sample of the uniform distribution on the unit circle. We consider the weighted Čech filtrations  $V[\Gamma, 0]$  and  $V[X, 0]$ , and the DTM-filtration  $W[X]$ , for  $p = 1$  and  $m = 0.1$ . They are represented in Figure 4 for the value  $t = 0.3$ .



■ **Figure 4** The set  $X$  and the sets  $V^t[\Gamma, 0]$ ,  $V^t[X, 0]$  and  $W^t[X]$  for  $p = 1$ ,  $m = 0.1$  and  $t = 0.3$ .

Because of the outliers  $\tilde{X}$ , the value of  $t$  from which the sets  $V^t[X, 0]$  are contractible is small. On the other hand, we observe that the set  $W^t[X]$  does not suffer too much from the presence of outliers.

We plot in Figure 5 the persistence diagrams of the persistence modules associated to  $\text{Rips}(\mathcal{V}[\Gamma, 0])$ ,  $\text{Rips}(\mathcal{V}[X, 0])$  and  $\text{Rips}(\mathcal{W}[X])$  ( $p = 1$ ,  $m = 0.1$ ).



■ **Figure 5** Persistence diagrams of some simplicial filtrations. Points in red (resp. green) represent the persistent homology in dimension 0 (resp. 1).

Observe that the diagrams  $D(\text{Rips}(\mathcal{V}[\Gamma, 0]))$  and  $D(\text{Rips}(\mathcal{W}[X]))$  appear to be close to each other, while  $D(\text{Rips}(\mathcal{V}[X, 0]))$  does not.

Applying the results of Section 3, we immediately obtain the following proposition.

► **Proposition 13.** *Consider two measures  $\mu, \nu$  on  $E$  with compact supports  $X$  and  $Y$ . Then*

$$d_i(V[X, d_\mu], V[Y, d_\nu]) \leq m^{-\frac{1}{2}} W_2(\mu, \nu) + 2^{\frac{1}{p}} d_H(X, Y).$$

*In particular, if  $X$  and  $Y$  are finite subsets of  $E$ , using  $\mu = \mu_X$  and  $\nu = \nu_Y$ , we obtain*

$$d_i(W[X], W[Y]) \leq m^{-\frac{1}{2}} W_2(X, Y) + 2^{\frac{1}{p}} d_H(X, Y).$$

Note that this stability result is worse than the stability of the usual Čech filtrations, which only involves the Hausdorff distance:  $d_i(V[X, 0], V[Y, 0]) \leq d_H(X, Y)$ . However, the term  $W_2(X, Y)$  is inevitable.

In the case where the Hausdorff distance  $d_H(X, Y)$  is small, it would be more robust to consider these usual Čech filtrations. However, in the case where it is large, the usual Čech filtrations may be very distant. On the other hand, the DTM-filtrations may still be close, as we discuss in the next subsection.

### 4.3 Stability when $p = 1$

We first consider the case  $p = 1$ , for which the proofs are simpler and results are stronger. We fix  $m \in (0, 1)$ . If  $\mu$  is a probability measure on  $E$  with compact support  $\text{supp}(\mu)$ , we define

$$c(\mu, m) = \sup_{\text{supp}(\mu)} (d_{\mu, m}).$$

If  $\mu = \mu_\Gamma$  is the empirical measure of a finite set  $\Gamma \subseteq E$ , we denote it  $c(\Gamma, m)$ .

► **Proposition 14.** *Let  $\mu$  be a probability measure on  $E$  with compact support  $\Gamma$ . Let  $d_\mu$  be the corresponding DTM. Consider a set  $X \subseteq E$  such that  $\Gamma \subseteq X$ . The weighted Čech filtrations  $V[\Gamma, d_\mu]$  and  $V[X, d_\mu]$  are  $c(\mu, m)$ -interleaved.*

*Moreover, if  $Y \subseteq E$  is another set such that  $\Gamma \subseteq Y$ ,  $V[X, d_\mu]$  and  $V[Y, d_\mu]$  are  $c(\mu, m)$ -interleaved.*

*In particular, if  $\Gamma$  is a finite set and  $\mu = \mu_\Gamma$  its empirical measure,  $W[\Gamma]$  and  $V[X, d_{\mu_\Gamma}]$  are  $c(\Gamma, m)$ -interleaved.*

► **Theorem 15.** *Consider two measures  $\mu, \nu$  on  $E$  with supports  $X$  and  $Y$ . Let  $\mu', \nu'$  be two measures with compact supports  $\Gamma$  and  $\Omega$  such that  $\Gamma \subseteq X$  and  $\Omega \subseteq Y$ . We have*

$$d_i(V[X, d_\mu], V[Y, d_\nu]) \leq m^{-\frac{1}{2}} W_2(\mu, \mu') + m^{-\frac{1}{2}} W_2(\mu', \nu') + m^{-\frac{1}{2}} W_2(\nu', \nu) + c(\mu', m) + c(\nu', m).$$

*In particular, if  $X$  and  $Y$  are finite, we have*

$$d_i(W[X], W[Y]) \leq m^{-\frac{1}{2}} W_2(X, \Gamma) + m^{-\frac{1}{2}} W_2(\Gamma, \Omega) + m^{-\frac{1}{2}} W_2(\Omega, Y) + c(\Gamma, m) + c(\Omega, m).$$

*Moreover, with  $\Omega = Y$ , we obtain*

$$d_i(W[X], W[\Omega]) \leq m^{-\frac{1}{2}} W_2(X, \Gamma) + m^{-\frac{1}{2}} W_2(\Gamma, \Omega) + c(\Gamma, m) + c(\Omega, m).$$

The last inequality of Theorem 15 can be seen as an approximation result. Indeed, suppose that  $\Omega$  is some underlying set of interest, and  $X$  is a sample of it with, possibly, noise or outliers. If one can find a subset  $\Gamma$  of  $X$  such that  $X$  and  $\Gamma$  are close to each other – in the Wasserstein metric – and such that  $\Gamma$  and  $\Omega$  are also close, then the filtrations  $W[X]$

58:12 DTM-Based Filtrations

and  $W[\Omega]$  are close. Their closeness depends on the constants  $c(\Gamma, m)$  and  $c(\Omega, m)$ . More generally, if  $X$  is finite and  $\mu'$  is a measure with compact support  $\Omega \subset X$  not necessarily finite, note that the first inequality gives

$$d_i(W[X], V[\Omega, d_{\mu'}]) \leq m^{-\frac{1}{2}}W_2(X, \Gamma) + m^{-\frac{1}{2}}W_2(\mu_\Gamma, \mu') + c(\Gamma, m) + c(\mu', m).$$

For any probability measure  $\mu$  of support  $\Gamma \subseteq E$ , the constant  $c(\mu, m)$  might be seen as a bias term, expressing the behaviour of the DTM over  $\Gamma$ . It relates to the concentration of  $\mu$  on its support. Recall that a measure  $\mu$  with support  $\Gamma$  is said to be  $(a, b)$ -standard, with  $a, b \geq 0$ , if for all  $x \in \Gamma$  and  $r \geq 0$ ,  $\mu(\overline{B}(x, r)) \geq \min\{ar^b, 1\}$ . For example, the Hausdorff measure associated to a compact  $b$ -dimensional submanifold of  $E$  is  $(a, b)$ -standard for some  $a > 0$ . In this case, a simple computation shows that there exists a constant  $C$ , depending only on  $a$  and  $b$ , such that for all  $x \in \Gamma$ ,  $d_{\mu, m}(x) \leq Cm^{\frac{1}{b}}$ . Therefore,  $c(\mu, m) \leq Cm^{\frac{1}{b}}$ .

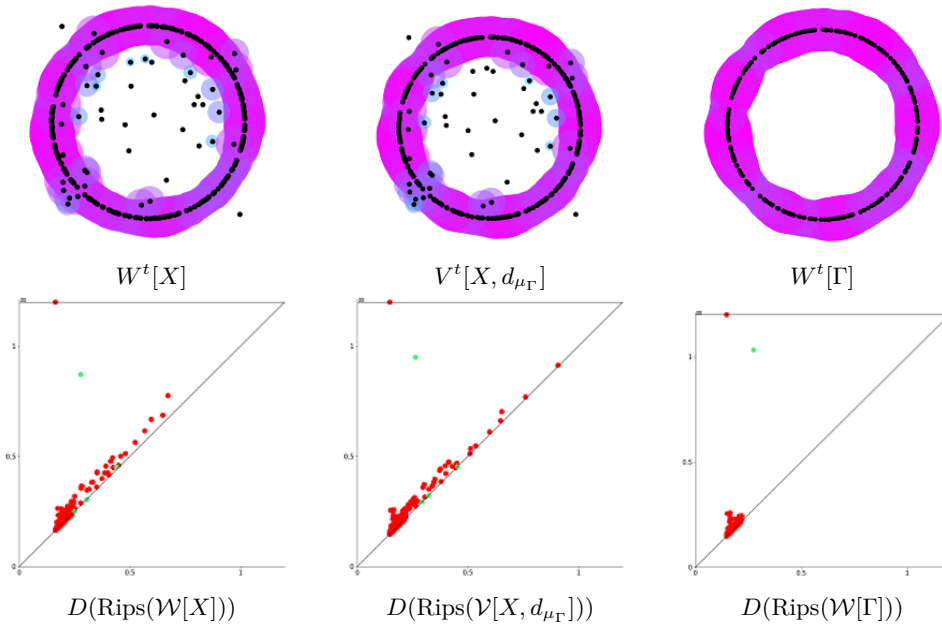
Regarding the second inequality of Theorem 15, suppose for the sake of simplicity that one can choose  $\Gamma = \Omega$ . The bound of Theorem 15 then reads

$$d_i(W[X], W[Y]) \leq m^{-\frac{1}{2}}W_2(X, \Gamma) + m^{-\frac{1}{2}}W_2(\Gamma, Y) + 2c(\Gamma, m).$$

Therefore, the DTM-filtrations  $W[X]$  and  $W[Y]$  are close to each other if  $\mu_X$  and  $\mu_Y$  are both close to a common measure  $\mu_\Gamma$ . This would be the case if  $X$  and  $Y$  are noisy samples of  $\Gamma$ . This bound, expressed in terms of Wasserstein distance rather than Hausdorff distance, shows the robustness of the DTM-filtration to outliers.

Notice that, in practice, for finite data sets  $X, Y$  and for given  $\Gamma$  and  $\Omega$ , the constants  $c(\Gamma, m)$  and  $c(\Omega, m)$  can be explicitly computed, as it amounts to evaluating the DTM on  $\Gamma$  and  $\Omega$ . This remark holds for the bounds of Theorem 15.

**Example.** Consider the set  $X = \tilde{X} \cup \Gamma$  as defined in the example page 10. Figure 6 displays the sets  $W^t[X]$ ,  $V^t[X, d_{\mu_\Gamma}]$  and  $W^t[\Gamma]$  for the values  $p = 1$ ,  $m = 0.1$  and  $t = 0.4$  and the persistence diagrams of the corresponding weighted Rips filtrations, illustrating the stability properties of Proposition 14 and Theorem 15.



■ **Figure 6** Filtrations for  $t = 0.4$ , and their corresponding persistence diagrams.

The following proposition relates the DTM-filtration to the filtration of  $E$  by the sublevel sets of the DTM.

► **Proposition 16.** *Let  $\mu$  be a probability measure on  $E$  with compact support  $K$ . Let  $m \in [0, 1)$  and denote by  $V$  the sublevel sets filtration of  $d_\mu$ . Consider a finite set  $X \subseteq E$ . Then*

$$d_i(V, W[X]) \leq m^{-\frac{1}{2}}W_2(\mu, \mu_X) + 2\epsilon + c(\mu, m),$$

with  $\epsilon = d_H(K \cup X, X)$ .

As a consequence, one can use the DTM-filtration to approximate the persistent homology of the sublevel sets filtration of the DTM, which is expensive to compute in practice.

We close this subsection by noting that a natural strengthening of Theorem 15 does not hold: let  $m \in (0, 1)$  and  $E = \mathbb{R}^n$  with  $n \geq 1$ . There is no constant  $C$  such that, for every probability measure  $\mu, \nu$  on  $E$  with supports  $X$  and  $Y$ , we have:

$$d_i(V[X, d_{\mu,m}], V[Y, d_{\nu,m}]) \leq CW_2(\mu, \nu).$$

The same goes for the weaker statement

$$d_i(\mathbb{V}[X, d_{\mu,m}], \mathbb{V}[Y, d_{\nu,m}]) \leq CW_2(\mu, \nu).$$

#### 4.4 Stability when $p > 1$

Now assume that  $p > 1$ ,  $m \in (0, 1)$  being still fixed. In this case, stability properties turn out to be more difficult to establish. For small values of  $t$ , Lemma 18 gives a tight non-additive interleaving between the filtrations. However, for large values of  $t$ , the filtrations are poorly interleaved. To overcome this issue we consider stability at the homological level, i.e. between the persistence modules associated to the DTM filtrations.

If  $\mu$  is a probability measure on  $E$  with compact support  $\Gamma$ , we define

$$c(\mu, m, p) = \sup_{\Gamma} (d_{\mu,m}) + \kappa(p)t_\mu(\Gamma),$$

where  $\kappa(p) = 1 - \frac{1}{p}$ , and  $t_\mu(\Gamma)$  is the filtration value of the simplex  $\Gamma$  in  $\mathcal{N}(\mathcal{V}[\Gamma, d_\mu])$ , the (simplicial) weighted Čech filtration. Equivalently,  $t_\mu(\Gamma)$  is the value  $t$  from which all the balls  $\bar{B}_{d_\mu}(\gamma, t)$ ,  $\gamma \in \Gamma$ , share a common point.

If  $\mu = \mu_\Gamma$  is the empirical measure of a finite set  $\Gamma \subseteq E$ , we denote it  $c(\Gamma, m, p)$ .

Note that we have the inequality  $\frac{1}{2}\text{diam}(\Gamma) \leq t_\mu(\Gamma) \leq 2\text{diam}(\Gamma)$ .

► **Proposition 17.** *Let  $\mu$  be a measure on  $E$  with compact support  $\Gamma$ , and  $d_\mu$  be the corresponding DTM of parameter  $m$ . Consider a set  $X \subseteq E$  such that  $\Gamma \subseteq X$ . The persistence modules  $\mathbb{V}[\Gamma, d_\mu]$  and  $\mathbb{V}[X, d_\mu]$  are  $c(\mu, m, p)$ -interleaved.*

*Moreover, if  $Y \subseteq E$  is another set such that  $\Gamma \subseteq Y$ ,  $\mathbb{V}[X, d_\mu]$  and  $\mathbb{V}[Y, d_\mu]$  are  $c(\mu, m, p)$ -interleaved.*

*In particular, if  $\Gamma$  is a finite set and  $\mu = \mu_\Gamma$  its empirical measure,  $\mathbb{W}[\Gamma]$  and  $\mathbb{V}[X, d_{\mu_\Gamma}]$  are  $c(\Gamma, m, p)$ -interleaved.*

The proof involves the two following ingredients. The first lemma gives a (non-additive) interleaving between the filtrations  $W[\Gamma]$  and  $V[X, d_{\mu_\Gamma}]$ , relevant for low values of  $t$ , while the second proposition gives a result for large values of  $t$ .

► **Lemma 18.** Let  $\mu, \Gamma$  and  $X$  be as defined in Proposition 17. Let  $\phi : t \mapsto 2^{1-\frac{1}{p}}t + \sup_{\Gamma} d_{\mu}$ . Then for every  $t \geq 0$ ,

$$V^t[\Gamma, d_{\mu}] \subseteq V^t[X, d_{\mu}] \subseteq V^{\phi(t)}[\Gamma, d_{\mu}].$$

► **Proposition 19.** Let  $\mu, \Gamma$  and  $X$  be as defined in Proposition 17. Consider the map  $v_*^t : \mathbb{V}^t[X, d_{\mu}] \rightarrow \mathbb{V}^{t+c}[X, d_{\mu}]$  induced in homology by the inclusion  $v^t : V^t[X, d_{\mu}] \rightarrow V^{t+c}[X, d_{\mu}]$ . If  $t \geq t_{\mu}(\Gamma)$ , then  $v^t$  is trivial.

► **Theorem 20.** Consider two measures  $\mu, \nu$  on  $E$  with supports  $X$  and  $Y$ . Let  $\mu', \nu'$  be two measures with compact supports  $\Gamma$  and  $\Omega$  such that  $\Gamma \subseteq X$  and  $\Omega \subseteq Y$ . We have

$$d_i(\mathbb{V}[X, d_{\mu}], \mathbb{V}[Y, d_{\nu}]) \leq m^{-\frac{1}{2}}W_2(\mu, \mu') + m^{-\frac{1}{2}}W_2(\mu', \nu') + m^{-\frac{1}{2}}W_2(\nu', \nu) + c(\mu', m, p) + c(\nu', m, p).$$

In particular, if  $X$  and  $Y$  are finite, we have

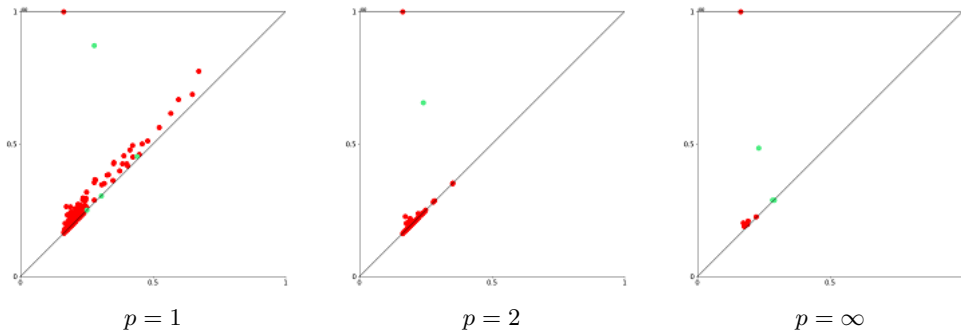
$$d_i(\mathbb{W}[X], \mathbb{W}[Y]) \leq m^{-\frac{1}{2}}W_2(X, \Gamma) + m^{-\frac{1}{2}}W_2(\Gamma, \Omega) + m^{-\frac{1}{2}}W_2(\Omega, Y) + c(\Gamma, m, p) + c(\Omega, m, p).$$

Moreover, with  $\Omega = Y$ , we obtain

$$d_i(\mathbb{W}[X], \mathbb{W}[\Gamma]) \leq m^{-\frac{1}{2}}W_2(X, \Gamma) + m^{-\frac{1}{2}}W_2(\Gamma, \Omega) + c(\Gamma, m, p) + c(\Omega, m, p).$$

Notice that when  $p = 1$ , the constant  $c(\Gamma, m, p)$  is equal to the constant  $c(\Gamma, m)$  defined in Subsection 4.3, and we recover Theorem 15 in homology.

As an illustration of these results, we represent in Figure 7 the persistence diagrams associated to the filtration  $\text{Rips}(\mathcal{W}[X])$  for several values of  $p$ . The point cloud  $X$  is the one defined in the example page 10. Observe that, as stated in Proposition 8, the number of red points (homology in dimension 0) is non-increasing with respect to  $p$ .



■ **Figure 7** Persistence diagrams of the simplicial filtrations  $\text{Rips}(\mathcal{W}[X])$  for several values of  $p$ .

## 5 Conclusion

In this paper we have introduced the DTM-filtrations that depend on a parameter  $p \geq 1$ . This new family of filtrations extends the filtration introduced in [3] that corresponds to the case  $p = 2$ .

The established stability properties are, as far as we know, of a new type: the closeness of two DTM-filtrations associated to two data sets relies on the existence of a well-sampled underlying object that approximates both data sets in the Wasserstein metric. This makes



the DTM filtrations robust to outliers. Even though large values of  $p$  lead to persistence diagrams with less points in the 0th homology, the choice of  $p = 1$  gives the strongest stability results. When  $p > 1$ , the interleaving bound is less significant since it involves the diameter of the underlying object, but the obtained bound is consistent with the case  $p = 1$  as it converges to the bound for  $p = 1$  as  $p$  goes to 1.

It is interesting to notice that the proofs rely on only a few properties of the DTM. As a consequence, the results should extend to other weight functions, such that the DTM with an exponent parameter different from 2, or kernel density estimators. Some variants concerning the radius functions in the weighted Čech filtration, are also worth considering. The analysis shows that one should choose radius functions whose asymptotic behaviour look like the one of the case  $p = 1$ . In the same spirit as in [12, 3] where sparse-weighted Rips filtrations were considered, it would also be interesting to consider sparse versions of the DTM-filtrations and to study their stability properties.

Last, the obtained stability results, depending on the choice of underlying sets, open the way to the statistical analysis of the persistence diagrams of the DTM-filtrations, a problem that will be addressed in a further work.

---

## References

- 1 Greg Bell, Austin Lawson, Joshua Martin, James Rudzinski, and Clifford Smyth. Weighted Persistent Homology. *arXiv preprint*, 2017. [arXiv:1709.00097](https://arxiv.org/abs/1709.00097).
- 2 Mickaël Buchet. *Topological inference from measures*. PhD thesis, Paris 11, 2014.
- 3 Mickaël Buchet, Frédéric Chazal, Steve Y Oudot, and Donald R Sheehy. Efficient and robust persistent homology for measures. *Computational Geometry*, 58:70–96, 2016.
- 4 F. Chazal, D. Cohen-Steiner, and Q. Mérigot. Geometric Inference for Probability Measures. *Journal on Found. of Comp. Mathematics*, 11(6):733–751, 2011.
- 5 Frédéric Chazal, Vin de Silva, Marc Glisse, and Steve Oudot. *The Structure and Stability of Persistence Modules*. SpringerBriefs in Mathematics, 2016.
- 6 Frédéric Chazal, Vin De Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, 2014.
- 7 Frédéric Chazal and Steve Yann Oudot. Towards persistence-based reconstruction in euclidean spaces. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, SCG '08, pages 232–241, New York, NY, USA, 2008. ACM.
- 8 Leonidas Guibas, Dmitriy Morozov, and Quentin Mérigot. Witnessed k-distance. *Discrete & Computational Geometry*, 49(1):22–45, 2013.
- 9 Fujitsu Laboratories. Estimating the Degradation State of Old Bridges-Fijutsu Supports Ever-Increasing Bridge Inspection Tasks with AI Technology. *Fujitsu Journal*, March 2018. URL: <https://journal.jp.fujitsu.com/en/2018/03/01/01/>.
- 10 J. Phillips, B. Wang, and Y Zheng. Geometric Inference on Kernel Density Estimates. In *Proc. 31st Annu. Sympos. Comput. Geom (SoCG 2015)*, pages 857–871, 2015.
- 11 Lee M Seversky, Shelby Davis, and Matthew Berger. On time-series topological data analysis: New data and opportunities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 59–67, 2016.
- 12 Donald R. Sheehy. Linear-Size Approximations to the Vietoris-Rips Filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013.
- 13 Yuhei Umeda. Time Series Classification via Topological Data Analysis. *Transactions of the Japanese Society for Artificial Intelligence*, 32(3):D–G72\_1, 2017.
- 14 Gudhi: Geometry understanding in higher dimensions. <http://gudhi.gforge.inria.fr/>.



# A Divide-and-Conquer Algorithm for Two-Point $L_1$ Shortest Path Queries in Polygonal Domains

Haitao Wang 

Department of Computer Science, Utah State University, Logan, UT 84322, USA  
haitao.wang@usu.edu

---

## Abstract

Let  $\mathcal{P}$  be a polygonal domain of  $h$  holes and  $n$  vertices. We study the problem of constructing a data structure that can compute a shortest path between  $s$  and  $t$  in  $\mathcal{P}$  under the  $L_1$  metric for any two query points  $s$  and  $t$ . To do so, a standard approach is to first find a set of  $n_s$  “gateways” for  $s$  and a set of  $n_t$  “gateways” for  $t$  such that there exist a shortest  $s$ - $t$  path containing a gateway of  $s$  and a gateway of  $t$ , and then compute a shortest  $s$ - $t$  path using these gateways. Previous algorithms all take quadratic  $O(n_s \cdot n_t)$  time to solve this problem. In this paper, we propose a divide-and-conquer technique that solves the problem in  $O(n_s + n_t \log n_s)$  time. As a consequence, we construct a data structure of  $O(n + (h^2 \log^3 h / \log \log h))$  size in  $O(n + (h^2 \log^4 h / \log \log h))$  time such that each query can be answered in  $O(\log n)$  time.

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

**Keywords and phrases** shortest paths, two-point queries,  $L_1$  metric, polygonal domains

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.59

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.01417>.

## 1 Introduction

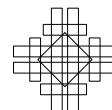
Let  $\mathcal{P}$  be a polygonal domain of  $h$  holes with a total of  $n$  vertices, i.e., there is an outer simple polygon containing  $h$  disjoint holes and each hole itself is a simple polygon. If  $h = 0$ , then  $\mathcal{P}$  becomes a simple polygon. For any two points  $s$  and  $t$ , an  $L_1$  shortest path from  $s$  to  $t$  in  $\mathcal{P}$  is a path connecting  $s$  and  $t$  with the minimum length under the  $L_1$  metric.

We consider the two-point  $L_1$  shortest path query problem: Construct a data structure for  $\mathcal{P}$  that can compute an  $L_1$  shortest path in  $\mathcal{P}$  for any two query points  $s$  and  $t$ . To do so, a standard approach is to first find a set of  $n_s$  “gateways” for  $s$  and a set of  $n_t$  “gateways” for  $t$  such that there exist a shortest  $s$ - $t$  path containing a gateway of  $s$  and a gateway of  $t$ , and then compute a shortest  $s$ - $t$  path using these gateways. Previous algorithms [6, 7] all take quadratic  $O(n_s \cdot n_t)$  time to solve this problem. In this paper, we propose a divide-and-conquer technique that solves the problem in  $O(n_s + n_t \log n_s)$  time.

As a consequence, we construct a data structure of  $O(n + (h^2 \log^3 h / \log \log h))$  size in  $O(n + (h^2 \log^4 h / \log \log h))$  time such that each query can be answered in  $O(\log n)$  time<sup>1</sup>. Previously, Chen et al. [7] built a data structure of  $O(n^2 \log n)$  size in  $O(n^2 \log^2 n)$  time that can answer each query in  $O(\log^2 n)$  time. Later Chen et al. [6] achieved  $O(\log n)$  time queries by building a data structure of  $O(n + h^2 \cdot \log h \cdot 4^{\sqrt{\log h}})$  space in  $O(n + h^2 \cdot \log^2 h \cdot 4^{\sqrt{\log h}})$  time. The preprocessing complexities of our result improve the previous work [6] by a super polylogarithmic factor. More importantly, our divide-and-conquer technique may be interesting in its own right.

---

<sup>1</sup> Throughout the paper, unless otherwise stated, when we say that the query time of a data structure is  $O(T)$ , we mean that the shortest path length can be computed in  $O(T)$  time and an actual shortest path can be output in additional linear time in the number of edges of the path.



**Related Work.** Better results exist for certain special cases. If  $\mathcal{P}$  is a simple polygon, then a shortest path in  $\mathcal{P}$  with minimum Euclidean length is also an  $L_1$  shortest path [20], and thus by using the data structure in [17, 19] for the Euclidean metric, one can build a data structure in  $O(n)$  time and space that can answer each query in  $O(\log n)$  time; recently Bae and Wang [2] proposed a simpler approach that can achieve the same performance. If  $\mathcal{P}$  and all holes of it are rectangles with axis-parallel edges, then ElGindy and Mitra [14] constructed a data structure of  $O(n^2)$  size in  $O(n^2)$  time that supports  $O(\log n)$  time queries.

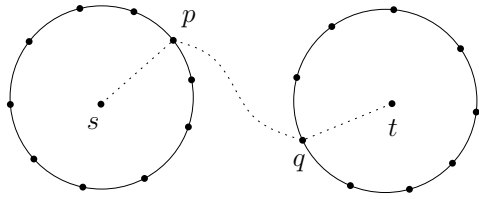
Better results are also known for *one-point queries* in the  $L_1$  metric [8, 11, 12, 22, 24, 25], i.e.,  $s$  is fixed in the input and only  $t$  is a query point. In particular, Mitchell [24, 25] built a data structure of  $O(n)$  size in  $O(n \log n)$  time that can answer each such query in  $O(\log n)$  time. Later Chen and Wang [8] reduced the preprocessing time to  $O(n + h \log h)$  if  $\mathcal{P}$  is already triangulated (which can be done in  $O(n \log n)$  or  $O(n + h \log^{1+\epsilon} h)$  time for any  $\epsilon > 0$  [3, 4]), while the query time is still  $O(\log n)$ .

The Euclidean counterparts have also been studied. For one-point queries, Hershberger and Suri [21] built a shortest path map of  $O(n)$  size with  $O(\log n)$  query time and the map can be built in  $O(n \log n)$  time and space. For two-point queries, Chiang and Mitchell [10] built a data structure of  $O(n^{11})$  size that can support  $O(\log n)$  time queries, and they also built a data structure of  $O(n + h^5)$  size with  $O(h \log n)$  query time. Other results with tradeoff between preprocessing and query time were also proposed in [10]. Also, Chen et al. [5] showed that with  $O(n^2)$  space one can answer each two-point query in  $O(\min\{|Q_s|, |Q_t|\} \cdot \log n)$  time, where  $Q_s$  (resp.,  $Q_t$ ) is the set of vertices of  $\mathcal{P}$  visible to  $s$  (resp.,  $t$ ). Guo et al. [18] gave a data structure of  $O(n^2)$  size that can support  $O(h \log n)$  time two-point queries.

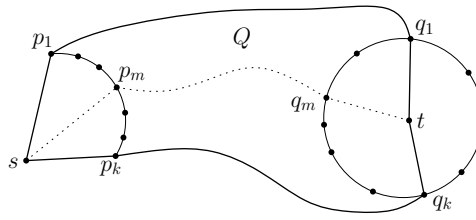
**Our Techniques.** We follow a similar scheme as in [6, 7], using a “path-preserving” graph  $G$  proposed by Clarkson et al. [11, 12] to determine a set  $V_g(q)$  of  $O(\log n)$  points (called “gateways”) for each query point  $q \in \{s, t\}$ , such that there exists an  $L_1$  shortest  $s$ - $t$  path that contains a gateway in  $V_g(s)$  and a gateway in  $V_g(t)$ . To find a shortest  $s$ - $t$  path, the main difficulty is to solve the following sub-problem. Let  $\pi(p, q)$  denote a shortest path between two points  $p$  and  $q$  in  $\mathcal{P}$ , and let  $d(p, q)$  denote its length. Suppose that the gateways of  $s$  (resp.,  $t$ ) are formed as a cycle around  $s$  (resp.,  $t$ ) such that there is a shortest  $s$ - $t$  path containing a gateway of  $s$  and a gateway of  $t$  (e.g., see Fig. 1). The point  $s$  is visible to each gateway  $p$  in  $V_g(s)$ , and thus  $d(s, p)$  can be obtained in  $O(1)$  time. The same applies to  $t$ . Also suppose in the preprocessing we have computed  $d(p, q)$  for any  $p \in V_g(s)$  and any  $q \in V_g(t)$ . The goal of the problem is to find  $p \in V_g(s)$  and  $q \in V_g(t)$  such that the value  $d(s, p) + d(p, q) + d(q, t)$  is minimized, so that a shortest  $s$ - $t$  path contains both  $p$  and  $q$ .

To solve the sub-problem, a straightforward method is to try all pairs of  $p$  and  $q$  with  $p \in V_g(s)$  and  $q \in V_g(t)$ , which is the approach used in both algorithms in [6, 7]. This takes  $O(n_s \cdot n_t)$  time, where  $n_s = |V_g(s)|$  and  $n_t = |V_g(t)|$ . In [7], both  $n_s$  and  $n_t$  are  $O(\log n)$ , which results in  $O(\log^2 n)$  query time. In [6], both  $n_s$  and  $n_t$  are reduced to  $O(\sqrt{\log n})$ , and thus the query time becomes  $O(\log n)$ , by using a larger “enhanced graph”  $G_E$  (than the original graph  $G$ ). More specifically, the size of  $G$  is  $O(n \log n)$  while the size of  $G_E$  is  $O(n\sqrt{\log n}2^{\sqrt{\log n}})$  (which is further reduced to  $O(h\sqrt{\log h}2^{\sqrt{\log h}})$  by other techniques [6]).

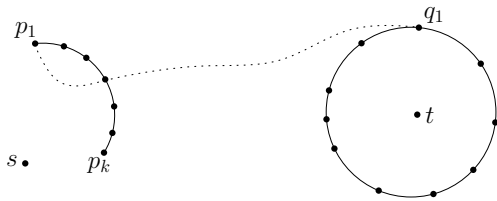
Our main contribution is to develop an  $O(n_s + n_t \log n_s)$  time algorithm for solving the above sub-problem. To this end, we explore the geometric structures of the problem and propose a divide-and-conquer technique, which can be roughly described as follows. For simplicity, suppose we only consider one piece of the gateway cycle of  $s$  (e.g., those in the first quadrant of  $s$ ) and order the gateways of  $s$  on that piece by  $p_1, p_2, \dots, p_k$  (e.g., see Fig. 2). Then, in a straightforward way, for  $p_1$ , we find a gateway, denoted by  $q_1$ , of  $t$  that



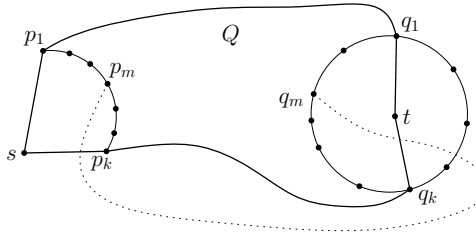
■ **Figure 1** Illustrating the gateways of  $s$  and  $t$  and a shortest  $s$ - $t$  path.



■ **Figure 2** Illustrating our divide-and-conquer scheme.



■ **Figure 3** A non-ideal situation: The shortest path from  $p_1$  to  $q_1$  crosses the gateway cycle of  $s$ .



■ **Figure 4** A non-ideal situation: The shortest path from  $p_m$  to  $q_m$  is not inside the region  $Q$ .

minimizes the value  $d(p_1, q) + d(q, t)$  for all  $q \in V_g(t)$ . Similarly, we find such a gateway  $q_k$  of  $t$  for  $p_k$ . Let  $P_1$  be the  $s$ - $t$  path  $\overline{sp_1} \cup \pi(p_1, q_1) \cup \overline{q_1t}$ . Similarly, let  $P_2$  be the path  $\overline{sp_k} \cup \pi(p_k, q_k) \cup \overline{q_kt}$ . In the “ideal” situation, the two paths do not intersect except at  $s$  and  $t$ , and they together form a cycle enclosing a plane region  $Q$  that contains all gateways  $p_1, p_2, \dots, p_k$  (e.g., see Fig. 2), and let  $V'_g(t)$  be the gateways of  $t$  that are also contained in  $Q$ . The next step is to process the median gateway  $p_m$  of  $s$  with  $m = \frac{k}{2}$ . The key observation is that we only need to consider the gateways in  $V'_g(t)$  instead of all the gateways of  $t$ , i.e., if a shortest  $s$ - $t$  path contains  $p_m$ , then there must be a shortest  $s$ - $t$  path containing  $p_m$  and a gateway in  $V'_g(t)$ . In this way, we only need to find the point, denoted by  $q_m$ , that minimizes the value  $d(p_m, q) + d(q, t)$  for all  $q \in V'_g(t)$ . Further, in the “ideal” situation, the path  $P_m = \overline{sp_m} \cup \pi(p_m, q_m) \cup \overline{q_mt}$  is inside the region  $Q$  and divides  $Q$  into two sub-regions (e.g., see Fig. 2). We then proceed on the two sub-regions recursively.

The above exhibits our algorithm in an “ideal” situation. Our major effort is to deal with the “non-ideal” situations. For example, what if the path  $P_1$  crosses the cycle piece of  $s$  (e.g., see Fig. 3), what if the path  $P_m$  is not in the region  $Q$  (e.g., see Fig. 4), what if  $q_1 = q_k$ , etc.

Our divide-and-conquer scheme may be somewhat similar to that for two-vertex shortest path queries in planar graphs, e.g., [9, 13]. However, a main difference is that in the planar graph case the query vertices are both from the input graph and the gateways are already known for each vertex (more specifically, the gateways in the planar graph case are the “border vertices” of the subgraphs in the decomposition of the input graph by separators), and thus one can compute certain information for the gateways in the preprocessing (many other techniques for shortest path queries in planar graphs, e.g., [15, 16, 26], also rely on this), while in our problem the gateways are only determined “online” during queries because query points can be anywhere in  $\mathcal{P}$ . This causes us to develop different techniques to tackle the problem (especially to resolve the non-ideal situations).

It might be tempting to see whether the Monge matrix searching techniques [1, 23] can be applied to further shave the logarithmic factor. However, due to the non-ideal situations such as those shown in Fig. 3 and Fig. 4, it is not clear to us whether this is possible.

With the above  $O(n_s + n_t \log n_s)$  time algorithm, if both  $n_s$  and  $n_t$  are bounded by  $O(\log n)$ , we can only obtain an  $O(\log n \log \log n)$  time query algorithm. To reduce the time to  $O(\log n)$ , we borrow some idea from the previous work [6] to construct a larger graph  $G_1$ , so that we can guarantee  $n_s = O(\log n)$  and  $n_t = O(\log n / \log \log n)$ , which leads to an  $O(\log n)$  time query algorithm. The size of  $G_1$  is only  $O(n \log^2 n / \log \log n)$ , which is slightly larger than the original  $O(n \log n)$ -sized graph  $G$  [7, 11, 12] and much smaller than the  $O(n \sqrt{\log n} 2^{\sqrt{\log n}})$ -sized enhanced graph  $G_E$  in [6]. Further, by the techniques similar to those used in [6], we can reduce the graph size to  $O(h \log^2 h / \log \log h)$ .

The rest of the paper is organized as follows. In Section 2, we define notation and review some previous work. In Section 3, we solve the sub-problem discussed above. In Section 4, we present our overall result. For ease of exposition, we make a general position assumption that no two vertices of  $\mathcal{P}$  including  $s$  and  $t$  have the same  $x$ - or  $y$ -coordinate. Unless otherwise stated, “length” refers to  $L_1$  length and “shortest paths” refers to  $L_1$  shortest paths. Due to the space limit, proofs and many details are omitted but can be found in the full paper.

## 2 Preliminaries

We introduce some notation and concepts, some of which are borrowed from the previous work [6, 7, 11, 12]. Two points  $p$  and  $q$  are *visible* to each other if the line segment  $\overline{pq}$  is in  $\mathcal{P}$ . For a point  $p$  and a vertical line segment  $l$  in  $\mathcal{P}$ , if there is a point  $q \in l$  such that  $\overline{pq}$  is horizontal and is in  $\mathcal{P}$ , then we say  $p$  is *horizontally visible* to  $l$  and call  $q$  the *horizontal projection* of  $p$  on  $l$ . For any point  $p$  in the plane, we use  $x(p)$  and  $y(p)$  to denote its  $x$ - and  $y$ -coordinates, respectively. For a path  $\pi$  in  $\mathcal{P}$ , we use  $|\pi|$  to denote its length.

For two points  $p$  and  $q$  in  $\mathcal{P}$ , we use  $\pi(p, q)$  to denote a shortest path from  $p$  to  $q$  and define  $d(p, q) = |\pi(p, q)|$ . For a segment  $\overline{pq}$ , let  $|\overline{pq}|$  denote its length. A path in  $\mathcal{P}$  is  *$x$ -monotone* if its intersection with any vertical line is either empty or connected. The  *$y$ -monotone* is defined similarly. If a path is both  $x$ -monotone and  $y$ -monotone, then it is  *$xy$ -monotone*. Note that an  $xy$ -monotone path in  $\mathcal{P}$  is a shortest path. Also, if there is an  $xy$ -monotone path between  $p$  and  $q$  in  $\mathcal{P}$ , then  $d(p, q) = |\overline{pq}|$  (although  $p$  may not be visible to  $q$ ).

Let  $\mathcal{V}$  denote the set of all vertices of  $\mathcal{P}$ . To differentiate from the vertices/edges in some graphs we define later, we often refer to the vertices/edges of  $\mathcal{P}$  as *polygon vertices/edges*. Let  $\partial\mathcal{P}$  denote the boundary of  $\mathcal{P}$  (including the boundaries of all the holes). For any point  $p \in \mathcal{P}$ , if we shoot a ray rightwards from  $p$ , let  $p^r$  denote the first point of  $\partial\mathcal{P}$  hit by the ray and call it the *rightward projection* of  $p$  on  $\partial\mathcal{P}$ . Similarly, we can define the leftward, upward, downward projections of  $p$  and denote them by  $p^l, p^u, p^d$ , respectively.

**A “path-preserving” graph  $G$ .** Clarkson et al. [11] proposed a graph  $G$  for computing  $L_1$  shortest paths in  $\mathcal{P}$ . We sketch  $G$  below since our algorithm will use a modified version of it.

To define  $G$ , there are two types of *Steiner points*. For each vertex of  $\mathcal{P}$ , its four projections on  $\partial\mathcal{P}$  are *type-1* Steiner points. Hence, there are  $O(n)$  Steiner points on  $\partial\mathcal{P}$ . The *type-2* Steiner points are defined on *cut-lines*, which can be organized into a binary tree  $\mathcal{T}$ , called the *cut-line tree*. Each node  $u$  of  $\mathcal{T}$  corresponds to a set  $\mathcal{V}(u)$  of vertices of  $\mathcal{P}$  and stores a cut-line  $l(u)$  that is a vertical line through the median  $x$ -coordinate of all vertices of  $\mathcal{V}(u)$ . If  $u$  is the root, then  $\mathcal{V}(u) = \mathcal{V}$ . In general, for the left (resp., right) child  $v$  of  $u$ ,  $\mathcal{V}(v)$  consists of all vertices of  $\mathcal{V}(u)$  to the left (resp., right) of  $l(u)$ . For each node  $u \in \mathcal{T}$  and each vertex  $p$  of  $\mathcal{V}(u)$ , if  $p$  is horizontally visible to  $l(u)$ , then the horizontal projection of  $p$  on  $l(u)$  is a type-2 Steiner point. Therefore,  $l(u)$  has at most  $|\mathcal{V}(u)|$  Steiner points. Since the total size  $|\mathcal{V}(u)|$  for all  $u$  in the same level of  $\mathcal{T}$  is  $O(n)$  and the height of  $\mathcal{T}$  is  $O(\log n)$ , the total number of type-2 Steiner points is  $O(n \log n)$ .

The graph  $G$  is thus defined as follows. First of all, the vertex set of  $G$  consists of all Steiner points (including all polygon vertices). Hence, it has  $O(n \log n)$  nodes. For the edges of  $G$ , for each vertex  $p$  of  $\mathcal{P}$ , if  $q$  is a Steiner point defined by  $p$ , then  $G$  has an edge  $\overline{pq}$ . For each polygon edge  $e$  of  $\mathcal{P}$ ,  $e$  may contain multiple Steiner points, and  $G$  has an edge connecting each adjacent pair of them. Further, for each cut-line  $l$  and for any two adjacent Steiner points on  $l$ , if they are visible to each other, then  $G$  has an edge connecting them.

Clearly,  $G$  has  $O(n \log n)$  nodes and edges. It was shown in [11, 12] that for any two polygon vertices of  $\mathcal{P}$ , the shortest path between them in the graph  $G$  is also a shortest path in  $\mathcal{P}$  (and thus the graph “preserves” shortest paths of the polygon vertices of  $\mathcal{P}$ ).

**Gateways.** To answer shortest path queries, Chen et al. [7] “insert” the two query points  $s$  and  $t$  into  $G$  by connecting them to some “gateways”. Intuitively, the gateways would be the vertices of  $G$  that connect to  $s$  and  $t$  respectively if  $s$  and  $t$  were vertices of  $\mathcal{P}$ , and thus they control shortest paths from  $s$  to  $t$ . Specifically, let  $V_g(s, G)$  denote the set of gateways for  $s$ , which has two subsets  $V_g^1(s, G)$  and  $V_g^2(s, G)$  of sizes  $O(1)$  and  $O(\log n)$ , respectively. We first define  $V_g^1(s, G)$ . For each projection point  $q$  of  $s$  on  $\partial\mathcal{P}$ , if  $v_1$  and  $v_2$  are the two Steiner points adjacent to  $q$  on the edge of  $\mathcal{P}$  containing  $q$ , then  $v_1$  and  $v_2$  are in  $V_g^1(s, G)$ . For  $V_g^2(s, G)$ , it is defined recursively on the cut-line tree  $\mathcal{T}$ . Let  $u$  be the root of  $\mathcal{T}$ . If  $s$  is horizontally visible to the cut-line  $l(u)$ , then  $l(u)$  is called a *projection cut-line* of  $s$  and the Steiner point on  $l(u)$  immediately above (resp., below) the horizontal projection  $s'$  of  $s$  on  $l(u)$  is a gateway in  $V_g^2(s, G)$  if it is visible to  $s'$ . Regardless of whether  $s$  is horizontally visible to  $l(u)$  or not, if  $s$  is to the left (resp., right) of  $l(u)$ , then we proceed to the left (resp., right) child of  $u$  until we reach a leaf of  $\mathcal{T}$ . Clearly,  $s$  has  $O(\log n)$  projection cut-lines, on a path from the root to a leaf in  $\mathcal{T}$ . Hence,  $|V_g^2(s, G)| = O(\log n)$ . Similarly we can define the gateway set  $V_g(t, G)$  for  $t$ . As will be shown later, for each gateway  $p$  of  $s$ ,  $\overline{sp}$  is in  $\mathcal{P}$ , and thus  $d(s, p) = |\overline{sp}|$ . The same applies to  $t$ . It is known [7] that if there is a shortest  $s$ - $t$  path containing a vertex of  $\mathcal{P}$ , then there must be a shortest  $s$ - $t$  path containing a gateway of  $s$  and a gateway of  $t$ ; otherwise, there must exist a shortest  $s$ - $t$  path  $\pi(s, t)$  that is  $xy$ -monotone with the following property: either  $\pi(s, t)$  consists of a horizontal segment and a vertical segment, or  $\pi(s, t)$  consists of three segments:  $\overline{ss'}$ ,  $\overline{s't'}$ , and  $\overline{t't}$ , where  $s'$  is a vertical (resp., horizontal) projection of  $s$  and  $t'$  is the horizontal (resp., vertical) projection of  $t$  on the same polygon edge, and we call such a shortest path a *trivial shortest path*.

**A straightforward query algorithm.** Given  $s$  and  $t$ , we can compute  $d(s, t)$  as follows. First, we check whether there exists a trivial shortest  $s$ - $t$  path, which can be done in  $O(\log n)$  time by using vertical and horizontal ray-shootings, after  $O(n \log n)$  time (or  $O(n + h \log^{1+\epsilon} h)$  time for any  $\epsilon > 0$  [3]) preprocessing to build the vertical and horizontal decompositions of  $\mathcal{P}$ . If yes, then we are done. Otherwise, we compute  $V_g(s, G)$  and  $V_g(t, G)$  in  $O(\log n)$  time after certain preprocessing [6, 7]. Suppose we have computed  $d(u, v)$  for any two vertices  $u$  and  $v$  of  $G$  in the preprocessing. Then,  $d(s, t) = \min_{p \in V_g(s, G), q \in V_g(t, G)} (|\overline{sp}| + d(p, q) + |\overline{qt}|)$  can be computed in  $O(\log^2 n)$  time since both  $|V_g(s, G)|$  and  $|V_g(t, G)|$  are  $O(\log n)$ .

**The main sub-problem.** To reduce the query time, since  $|V_g^1(s, G)| = O(1)$  and  $|V_g^1(t, G)| = O(1)$ , the main sub-problem is to determine the value  $\min_{p \in V_g^2(s, G), q \in V_g^2(t, G)} (|\overline{sp}| + d(p, q) + |\overline{qt}|)$ . This is the sub-problem we discussed in Section 1. Note that the case  $p \in V_g^1(s, G)$  and  $q \in V_g^2(t, G)$ , or the case  $p \in V_g^2(s, G)$  and  $q \in V_g^1(t, G)$  can be handled in  $O(\log n)$  time.



### 3 Solving the Main Sub-Problem

In this section, we present an  $O(n_s + n_t \log n_s)$  time algorithm for our main sub-problem, where  $n_s = |V_g^2(s, G)|$  and  $n_t = |V_g^2(t, G)|$ .

We consider the vertices of  $G$  also as the corresponding points in  $\mathcal{P}$ . Note that although  $G$  preserves shortest paths between all polygon vertices of  $\mathcal{P}$ , it may not preserve shortest paths for all vertices of  $G$ , i.e., for two vertices  $p$  and  $q$  of  $G$ , the shortest path from  $p$  to  $q$  in  $G$  may not be a shortest path in  $\mathcal{P}$ . For this reason, as preprocessing, for each vertex  $q$  of  $G$ , we compute a shortest path tree  $T(q)$  in  $\mathcal{P}$  from  $q$  to all vertices of  $G$  using the algorithm in [24, 25], which can be done in  $O(n \log^2 n)$  time since  $G$  has  $O(n \log n)$  vertices. For each vertex  $p$  of  $G$ , we use  $\pi_q(p)$  to denote the path in  $T(q)$  from the root  $q$  to  $p$ , which is a shortest path in  $\mathcal{P}$ , and we refer to the edge incident to  $p$  as the *last edge* of  $\pi_q(p)$ ; we explicitly store  $d(p, q)$  and the last edge of  $\pi_q(p)$ . Note that  $\pi_q(p)$ , computed by the algorithm [24, 25], has the following property [24, 25]: all vertices of the path other than  $p$  and  $q$  are polygon vertices of  $\mathcal{P}$ . Doing the above for all vertices  $q$  of  $G$  takes  $O(n^2 \log^3 n)$  time and  $O(n^2 \log^2 n)$  space.

Given  $s$  and  $t$ , following the discussion in Section 2, we assume that there are no trivial shortest  $s$ - $t$  paths and there is a shortest  $s$ - $t$  path containing a gateway in  $V_g^2(s, G)$  and a gateway in  $V_g^2(t, G)$ . To simplify the notation, let  $V(s) = V_g^2(s, G)$  and  $V(t) = V_g^2(t, G)$ .

A gateway of  $V(s)$  is called a *via gateway* if there is a shortest  $s$ - $t$  path containing it. Our goal is to find a via gateway, after which a shortest  $s$ - $t$  path can be computed in additional  $O(\log n)$  time by checking each gateway of  $t$ . In the following, we present an  $O(n_s + n_t \log n_s)$  time algorithm. Without loss of generality, we assume that the first quadrant of  $s$  has a via gateway. Below, we will describe our algorithm only on the gateways of  $V(s)$  in the first quadrant of  $s$  (our algorithm will run on each quadrant of  $s$  separately). By slightly abusing the notation, we still use  $V(s)$  to denote the gateways of  $V(s)$  in the first quadrant of  $s$ .

Before describing our algorithm, we introduce some geometric structures, among which the most important ones are a *gateway region* of  $s$  and an *extended gateway region* of  $t$ . Chen et al. [7] introduced the gateway region for rectilinear polygonal domains and here we extend the concept to the arbitrary polygonal domain case. In particular, our extended gateway region has several new components that are critical to our algorithm, and it may be interesting in its own right. Due to the space limit, we only discuss some of their properties that are needed for describing our algorithm later, and other properties that are used to prove the correctness of our algorithm are omitted.

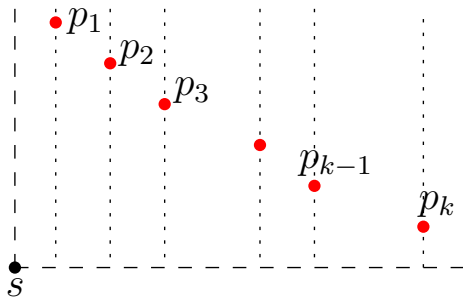
**The Gateway Region  $R(s)$ .** We define a *gateway region*  $R(s)$  for  $s$ . Let  $p_1, p_2, \dots, p_k$  be the gateways of  $s$  ordered from left to right (e.g., see Fig. 5). Note that each  $p_i$  is a type-2 Steiner point on a projection cut-line of  $s$ . Let  $l_1, l_2, \dots, l_k$  be the projection cut-lines of  $s$  that contain these gateways, respectively, and thus they are also sorted from left to right. It is known [6, 7] that the  $y$ -coordinates of  $p_1, p_2, \dots, p_k$  are in non-increasing order.

Let  $s_1$  be the intersection of  $l_1$  with the horizontal line through  $p_k$  (e.g., see Fig. 6). For each  $p_i$  with  $i \in [2, k]$ , project  $p_i$  leftwards horizontally onto  $l_{i-1}$  at a point  $p'_i$  (note that  $p'_i = p_{i-1}$  if  $y(p_{i-1}) = y(p_i)$ ). Define  $R(s)$  as the region bounded by the line segments connecting the points  $s_1, p_1, p'_2, p_2, \dots, p'_k, p_k$ , and  $s_1$  in this cyclic order.

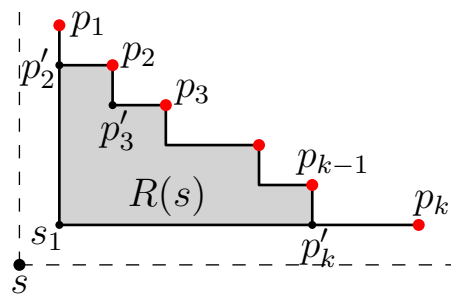
We use  $\beta_s$  to denote the boundary portion of  $R(s)$  from  $p_1$  to  $p_k$  that contains all gateways of  $V(s)$ . We call  $\beta_s$  the *ceiling*,  $s_1 p'_2$  the *left boundary*, and  $s_1 p'_k$  the *bottom boundary* of  $R(s)$ . We refer to the region  $R(s)$  excluding the points on  $\beta_s$  as the *interior* of  $R(s)$ .

► **Observation 1.**  $R(s)$  is in  $\mathcal{P}$ , and the interior of  $R(s)$  does not contain any polygon vertex.

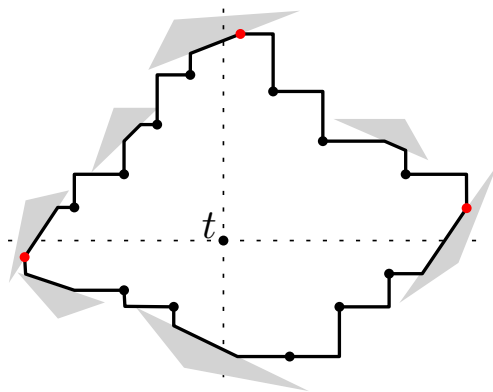




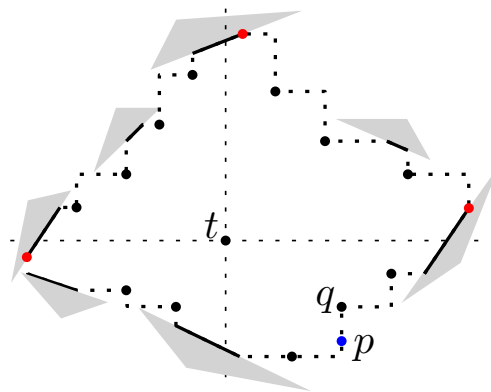
■ **Figure 5** Illustrating the gateways of  $V(s)$  and the cut-lines containing them.



■ **Figure 6** Illustrating the gateway region  $R(s)$ . The red points are gateways of  $V(s)$ .



■ **Figure 7** Illustrating  $R(t)$ , bounded by the solid segments. The black points other than  $t$  are all gateways and the three red points are special gateways.



■ **Figure 8** Illustrating the transparent edges (the dotted segments). The three red points are special gateways. For Lemma 1, a point  $p$  on a transparent edge as well as an endpoint  $q$  of the edge is also shown.

**The Extended Gateway Region  $R(t)$ .** For  $t$ , we define an *extended gateway region*  $R(t)$ . Unlike  $R(s)$ , which does not contain  $s$ ,  $R(t)$  contains  $t$ , e.g., see Fig. 7. Instead of giving the detailed definition of  $R(t)$ , which is quite lengthy, we only discuss several key properties of it.

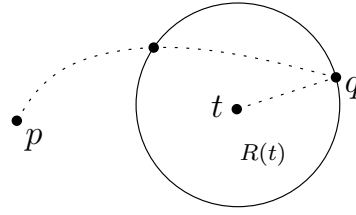
Let  $\mathcal{V}_1$  denote the set consisting of all polygon vertices and their projection points on  $\partial\mathcal{P}$ .

In general,  $R(t)$  is a simple polygon that contains  $t$ , with  $O(\log n)$  vertices. Let  $\partial R(t)$  denote its boundary. Each edge of  $\partial R(t)$  is vertical, horizontal, or on a polygon edge. If an edge of  $\partial R(t)$  is not on a polygon edge, then we call it a *transparent edge* (e.g., see Fig. 8). It is the transparent edges that separate the interior of  $R(t)$  from the outside (i.e., for any point  $p$  of  $\mathcal{P}$  outside  $R(t)$ , any path from  $p$  to  $t$  in  $\mathcal{P}$  must intersect a transparent edge of  $R(t)$ ). All gateways of  $V(t)$  are on  $\partial R(t)$ . In addition, at most four points of  $\mathcal{V}_1$  are considered as *special gateways* that are also on  $\partial R(t)$ , and we include them in  $V(t)$ .

► **Lemma 1.** *The point  $t$  is visible to each gateway in  $V(t)$  and  $R(t)$  is in  $\mathcal{P}$ . For any point  $p$  outside  $R(t)$ , there is a shortest path from  $p$  to  $t$  that contains a gateway in  $V(t)$ . For any point  $p$  on a transparent edge  $e$ , one of the endpoints  $q$  of  $e$  is a gateway in  $V(t)$  such that  $\overline{pq} \cup \overline{qt}$  is an  $xy$ -monotone (and thus a shortest) path from  $p$  to  $t$  (e.g., see Fig. 8).*

In addition, the properties of  $R(t)$  guarantee that for any point  $p$  outside  $R(t)$ , a shortest path from  $p$  to  $t$  cannot separate  $\partial R(t)$  into two disjoint pieces (see Fig. 9).

If we store the four projections on  $\partial\mathcal{P}$  for each Steiner point of  $G$  (this costs  $O(n \log n)$  additional space), then  $R(t)$  can be explicitly computed in  $O(\log n)$  time.



■ **Figure 9** The following situation cannot occur: A shortest path from  $t$  to a point  $p$  outside  $R(t)$  separates the boundary of  $R(t)$  (the solid circle) into two or more disjoint pieces.

We remark that  $R(s)$  and  $R(t)$  are defined differently because  $s$  and  $t$  are not treated symmetrically in our algorithm. For example, we need  $R(t)$  to have the properties in Lemma 1, which are not necessary for  $R(s)$ . Also, as will be clear later, treating  $s$  and  $t$  differently helps us to further reduce the complexities of our data structure.

### 3.1 The Query Algorithm

Consider the gateway region  $R(s)$  of  $s$ . Note that for any  $p_i \in V(s)$ , there is always a shortest path from  $s$  to  $p_i$  containing  $s_1$  as we can show that  $\overline{ss_1}$  in  $\mathcal{P}$ . Recall that we have assumed that there exists a shortest  $s$ - $t$  path that contains a gateway of  $V(s)$ . The above implies that there exists a shortest path from  $s_1$  to  $t$  that contains a gateway of  $V(s)$ , and if we can find such a path, by attaching  $\overline{ss_1}$  to the path, we can obtain a shortest  $s$ - $t$  path. For convenience, in the following, we will focus on finding a shortest path from  $s_1$  to  $t$  that contains a gateway of  $V(s)$ . By slightly abusing the notation, we still use  $s$  to represent  $s_1$ . Again, our goal is to compute a via gateway of  $s$  in  $V(s)$ . We first check whether there is a trivial shortest  $s$ - $t$  path in  $O(\log n)$  time. If yes, we are done. Otherwise, we have the following lemma.

► **Lemma 2.** *If  $R(t)$  contains a gateway  $p$  of  $V(s)$ , then  $\overline{sp} \cup \overline{pt}$  is a shortest  $s$ - $t$  path; otherwise,  $R(s)$  does not intersect  $R(t)$ .*

We check whether  $R(t)$  contains a gateway of  $V(s)$ . Due to the properties of  $R(t)$ , this can be done in  $O(n_t + n_s)$  time, as follows. We check the four quadrants of  $t$  separately. Let  $R_1(t)$  be  $R(t)$  in the first quadrant of  $t$ . To check whether  $R_1(t)$  contains a gateway of  $V(s)$ , we can simply scan the gateways of  $V(s)$  and the gateways of  $V(t)$  in  $R_1(t)$  simultaneously from left to right (somewhat like merge sort). We do the same for other quadrants of  $t$ .

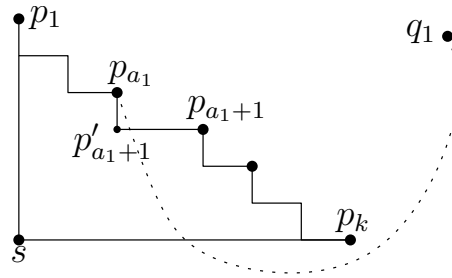
If  $R(t)$  contains a gateway of  $V(s)$ , then by Lemma 2, we have found a shortest  $s$ - $t$  path. Otherwise,  $R(s)$  and  $R(t)$  are disjoint and we proceed as follows. By Lemma 1, for each  $p \in V(s)$ ,  $d(p, t) = \min_{q \in V(t)} (d(p, q) + |qt|)$ , and we call such a gateway  $q$  of  $V(t)$  minimizing the above value a *coupled gateway* of  $p$  and use  $c(p)$  to denote it.

Our algorithm will compute a “candidate” coupled gateway  $c'(p)$  for every gateway  $p$  of  $V(s)$  such that if  $p \in V(s)$  is via gateway, then  $c(p) = c'(p)$ . Therefore, once the algorithm is done, the gateway  $p$  that minimizes the value  $|\overline{sp}| + d(p, c'(p)) + |\overline{c'(p)t}|$  is a via gateway.

For any two points  $a$  and  $b$  on the ceiling  $\beta_s$  of  $R(s)$ , we use  $\beta_s[a, b]$  to denote the sub-path of  $\beta_s$  between  $a$  and  $b$ , which is  $xy$ -monotone. This means that we can compute  $d(p_i, p_j) = |\overline{p_i p_j}|$  in constant time for every two gateways  $p_i$  and  $p_j$  in  $V(s)$ .

We consider  $V(t)$  as a cyclic list of points in *counterclockwise* order around  $t$  (we use “counterclockwise” since the list of  $V(s) = \{p_1, p_2, \dots, p_k\}$  are in clockwise order around  $s$ ).

We first compute  $c(p_1)$  in a straightforward manner, i.e., check every gateway of  $V(t)$ . This takes  $O(n_t)$  time (since  $d(p, q)$  for any  $p \in V(s)$  and  $q \in V(t)$  is already computed in our preprocessing). We also compute  $c(p_k)$  in the same way. If there are multiple  $c(p_k)$ 's, then



■ **Figure 10** The shortest path  $\pi_{q_1}(p_{a_1})$  goes through the interior of  $R(s)$ .

we let  $c(p_k)$  refer to the first one from  $c(p_1)$  in the counterclockwise order around  $t$ . Further, if there is more than one  $c(p_1)$  from the current  $c(p_1)$  to  $c(p_k)$  in the counterclockwise order, then we update  $c(p_1)$  to the one closest to  $c(p_k)$ . To simplify the notation, let  $q_1 = c(p_1)$  and  $q_k = c(p_k)$ . Note that  $q_1 = q_k$  is possible. The following lemma will be useful for circumventing the “non-ideal” situation depicted in Fig. 3. Its correctness relies on the fact that the ceiling  $\beta_s$  of  $R(s)$  is  $xy$ -monotone (and thus is a shortest path).

► **Lemma 3.** *For any  $p_i$  of  $V(s)$ , if  $d(p_1, p_i) + d(p_i, q_1) = d(p_1, q_1)$ , then  $d(p_1, p_j) + d(p_j, q_1) = d(p_1, q_1)$  and  $c(p_j) = q_1$  for each  $j \in [1, i]$ ; similarly, if  $d(p_k, p_i) + d(p_i, q_k) = d(p_k, q_k)$ , then  $d(p_k, p_j) + d(p_j, q_k) = d(p_k, q_k)$  and  $c(p_j) = q_k$  for each  $j \in [i, k]$ .*

Let  $a_1$  be the largest index  $i \in [1, k]$  such that  $d(p_1, p_i) + d(p_i, q_1) = d(p_1, q_1)$ , which can be computed in  $O(a_1)$  time, as follows. Starting from  $i = 2$ , we simply check whether  $d(p_1, p_i) + d(p_i, q_1) = d(p_1, q_1)$ , which can be done in  $O(1)$  time since  $d(p_1, p_i) = |\overline{p_1 p_i}|$  can be computed in constant time and  $d(p_i, q_1)$  has been computed in the preprocessing. If yes, we proceed with  $i + 1$ ; otherwise, we stop the algorithm and set  $a_1 = i - 1$ . We call the above a *stair-walking procedure*. The correctness is due to Lemma 3.

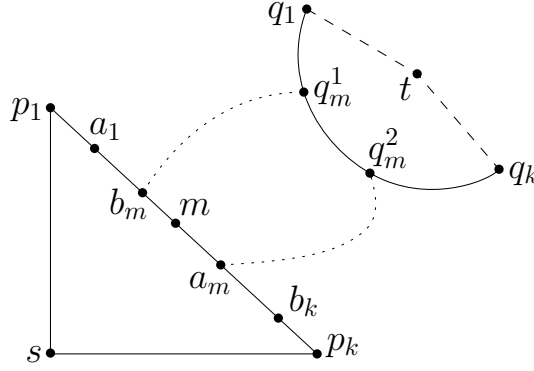
Similarly, define  $b_k$  to be the smallest index  $i \in [1, k]$  such that  $d(p_k, p_i) + d(p_i, q_k) = d(p_k, q_k)$ . By a symmetric stair-walking procedure, we can compute  $b_k$  as well. By Lemma 3, for each  $i \in [1, a_1] \cup [b_k, k]$ ,  $c(p_i)$  is known. Hence, if  $a_1 \geq b_k$ , then  $c(p)$  for each  $p \in V(s)$  is computed and we can finish the algorithm. Otherwise, we proceed as follows.

Recall that  $\pi_{q_1}(p_{a_1})$  is the shortest path from  $q_1$  to  $p_{a_1}$  obtained from the shortest path tree  $T(q_1)$ , and  $\pi_{q_k}(p_{b_k})$  is from  $T(q_k)$ . Lemmas 4 and 5 are for dealing with the non-ideal situation in which  $\pi_{q_1}(p_{a_1})$  (resp.,  $\pi_{q_k}(p_{b_k})$ ) goes through the interior of  $R(s)$  (see Fig. 10).

► **Lemma 4.** *(1)  $\pi_{q_1}(p_{a_1})$  contains a point in the interior of  $R(s)$  only if its last edge (i.e., the edge incident to  $p_{a_1}$ ) intersects the bottom boundary of  $R(s)$ . (2)  $\pi_{q_k}(p_{b_k})$  contains a point in the interior of  $R(s)$  only if its last edge (i.e., the edge incident to  $p_{b_k}$ ) intersects the left boundary of  $R(s)$ .*

► **Lemma 5.** *If the last edge of  $\pi_{q_1}(p_{a_1})$  intersects the bottom boundary of  $R(s)$ , or the last edge of  $\pi_{q_1}(p_{b_k})$  intersects the left boundary of  $R(s)$ , then  $p_j$  for each  $j \in [a_1 + 1, b_k - 1]$  cannot be a via gateway.*

Due to our preprocessing, we can check in constant time whether the last edge of  $\pi_{q_1}(p_{a_1})$  intersects the bottom boundary of  $R(s)$ . Similarly, we can check whether the last edge of  $\pi_{q_1}(p_{b_k})$  intersects the left boundary of  $R(s)$ . If the answer is yes for either case, then by Lemma 5, we can stop the algorithm (i.e., no need to compute the coupled gateways for any  $p_i$  with  $i \in [a_1 + 1, b_k - 1]$ ). Otherwise, by Lemma 4, neither  $\pi_{q_1}(p_{a_1})$  nor  $\pi_{q_1}(p_{b_k})$  contains a point in the interior of  $R(s)$ . We proceed as follows.



■ **Figure 11** Illustrating a schematic view of the indices:  $a_1$ ,  $b_m$ ,  $m$ ,  $a_m$ , and  $b_k$ .

Recall that  $q_1 = q_k$  is possible. Depending on whether  $q_1 = q_k$ , there are two cases. In the sequel, we describe our algorithm for the *unequal case*  $q_1 \neq q_k$ . The *equal-case*  $q_1 = q_k$  can be reduced to the unequal case, which is omitted and can be found in the full paper.

In the following, we assume that  $q_1 \neq q_k$ . Since  $q_1 \neq q_k$ ,  $q_1$  and  $q_k$  partition the cyclic list  $V(t)$  into two sequential lists, one of which has  $q_1$  as the first point and  $q_k$  as the last point following the counterclockwise order around  $t$ , and we use  $V_t(1, k)$  to denote that list. Lemma 6 follows from our particular way of defining  $q_1$  and  $q_k$ .

► **Lemma 6.** *The two paths  $\pi_{q_1}(p_{a_1})$  and  $\pi_{q_k}(p_{b_k})$  do not intersect.*

For any  $i$  and  $j$  with  $1 \leq i \leq j \leq k$ , we use the interval  $[i, j]$  to represent the gateways  $p_i, p_{i+1}, \dots, p_j$ . Our algorithm works on the interval  $[1, k]$  and  $V_t(1, k)$ .

► **Lemma 7.** *For any gateway  $p_j$  with  $j \in [a_1 + 1, b_k - 1]$ , if  $p_j$  is a via gateway, then it has a coupled gateway in  $V_t(1, k)$ .*

By Lemma 7, to compute the candidate coupled gateways for all  $p_i$  with  $i \in [a_1 + 1, b_k - 1]$ , we only need to consider  $V_t(1, k)$ . In the following, we work on the problem recursively. We may consider each recursive step as working on a subproblem, denoted by  $([i', j'], [i, j], V_t(i, j))$  with  $[i', j'] \subseteq [i, j] \subseteq [1, k]$ , where the goal is to find candidate coupled gateways from a sublist  $V_t(i, j)$  of  $V_t(1, k)$  for the gateways in  $[i', j']$ , and further, there exist a shortest path from  $p_{a_i}$  to the first point of  $V_t(i, j)$  and a shortest path from  $p_{b_j}$  to the last point of  $V_t(i, j)$  such that the two paths do not intersect and neither path contains a point in the interior of  $R(s)$ . Initially, our subproblem is  $([a_1 + 1, b_k - 1], [1, k], V_t(1, k))$ . We proceed as follows.

If  $b_k - 1 = a_1 + 1$ , then the interval  $[a_1 + 1, b_k - 1]$  has only one gateway  $p$ . We simply check all gateways of  $V_t(1, k)$  to find the point  $q$  that minimizes the value  $d(p, q) + d(q, t)$  among all  $q \in V_t(1, k)$ , and then return  $q$  as the candidate coupled gateway of  $p$ . The algorithm can stop. Otherwise, we proceed as follows.

Let  $m = \lfloor (a_1 + b_k)/2 \rfloor$ . We compute a gateway in  $V_t(1, k)$  that minimizes the value  $d(p_m, q) + d(q, t)$  for all  $q \in V_t(1, k)$ , and in case of a tie, we use  $q_m^1$  and  $q_m^2$  to refer to the first and the last such gateways in  $V_t(1, k)$ , respectively. Let  $V_t(1, m)$  and  $V_t(m, k)$  denote the sublists of  $V_t(1, k)$  from  $q_1$  to  $q_m^1$  and from  $q_m^2$  to  $q_k$ , respectively.

Define  $a_m$  to be the largest index  $i \in [m, b_k - 1]$  such that  $d(p_m, q_m^2) = d(p_m, p_i) + d(p_i, q_m^2)$  and  $b_m$  the smallest index  $i \in [a_1 + 1, m]$  such that  $d(p_m, q_m^1) = d(p_m, p_i) + d(p_i, q_m^1)$ . See Fig. 11. We can compute  $a_m$  and  $b_m$  by a similar stair-walking procedure as before. With Lemma 7 and similarly as Lemma 3, for each  $i \in [b_m, m - 1]$ , if  $p_i$  is a via gateway, then  $q_m^1$  is a coupled gateway of  $p_i$ , and for each  $i \in [m + 1, a_m]$ , if  $p_i$  is a via gateway, then  $q_m^2$  is a coupled gateway of  $p_i$ . Thus we can set candidate coupled gateways accordingly.

If  $a_m = b_k - 1$  and  $b_m = a_1 + 1$ , then the candidate coupled gateways of all gateways in  $[1, k]$  have been computed and we can stop the algorithm. If  $a_m = b_k - 1$  but  $b_m > a_1 + 1$ , we work recursively on the subproblem  $([a_1 + 1, b_m - 1], [1, k], V_t(1, k))$  (note that the size of the first interval is reduced by at least half). Similarly, if  $b_m = a_1 + 1$  but  $a_m < b_k - 1$ , then we work recursively on the subproblem  $([a_m + 1, b_k - 1], [1, k], V_t(1, k))$ . Otherwise, both  $b_m > a_1 + 1$  and  $a_m < b_k - 1$  hold, and we proceed as follows. We have the following two lemmas that are similar to Lemmas 4 and 5. By definition, either  $a_m > b_m$  or  $a_m = b_m = m$ .

► **Lemma 8.** (1) The path  $\pi_{q_m^2}(p_{a_m})$  contains a point in the interior of  $R(s)$  only if the last edge of the path intersects the bottom boundary of  $R(s)$ , in which case the intersection at the bottom boundary of  $R(s)$  has  $x$ -coordinate in  $[x(s), x(p_{a_m+1})]$ . (2) The path  $\pi_{q_m^1}(p_{b_m})$  contains a point in the interior of  $R(s)$  only if the last edge of the path intersects the left boundary of  $R(s)$ , in which case the intersection at the left boundary of  $R(s)$  has  $y$ -coordinate in  $[y(s), y(p_{b_m-1})]$ .

► **Lemma 9.** (1) If the last edge of  $\pi_{q_m^2}(p_{a_m})$  intersects the bottom boundary of  $R(s)$ , then  $p_i$  cannot be a via gateway for any  $i \in [a_m + 1, b_k - 1]$ . (2) If the last edge of  $\pi_{q_m^1}(p_{b_m})$  intersects the left boundary of  $R(s)$ , then  $p_i$  cannot be a via gateway for any  $i \in [a_1 + 1, b_m - 1]$ .

In constant time we can check whether the two cases in Lemma 9 happen. If both cases happen, then we can stop the algorithm. If the second case happens and the first one does not, then we recursively work on the subproblem  $([a_m + 1, b_k - 1], [1, k], V_t(1, k))$ . If the first case happens and the second one does not, then we recursively work on the subproblem  $([a_1 + 1, b_m - 1], [1, k], V_t(1, k))$ . In the following, we assume that neither case happens. By Lemma 8, neither  $\pi_{q_m^2}(p_{a_m})$  nor  $\pi_{q_m^1}(p_{b_m})$  contains a point in the interior of  $R(s)$ . Consequently, we have the following lemma.

► **Lemma 10.** (1) For each  $i \in [a_m + 1, b_k - 1]$ , if  $p_i$  is a via gateway, then  $p_i$  has a coupled gateway in  $V_t(m, k)$ . If  $q_m^2 \neq q_k$ , then  $\pi_{q_m^2}(p_{a_m})$  does not intersect  $\pi_{q_k}(p_{b_k})$ . (2) For each  $i \in [a_1 + 1, b_m - 1]$ , if  $p_i$  is a via gateway, then  $p_i$  has a coupled gateway in  $V_t(1, m)$ . If  $q_m^1 \neq q_1$ , then  $\pi_{q_m^1}(p_{b_m})$  does not intersect  $\pi_{q_1}(p_{a_1})$ .

Based on Lemma 10, our algorithm proceeds as follows. If  $q_m^2 = q_k$ , then we set  $q_k$  as the candidate coupled gateway for each  $p_i$  with  $i \in [a_m + 1, b_k - 1]$ . Otherwise, we call the algorithm recursively on the subproblem  $([a_m + 1, b_k - 1], [m, k], V_t(m, k))$ . Similarly, if  $q_m^1 = q_1$ , then we set  $q_1$  as the candidate coupled gateway for each  $p_i$  with  $i \in [a_1 + 1, b_m - 1]$ . Otherwise, we call the algorithm recursively on the subproblem  $([a_1 + 1, b_m - 1], [1, m], V_t(1, m))$ .

For the running time, notice that the stair-walking procedure spends  $O(1)$  time on finding a coupled gateway for a gateway of  $V(s)$ . Hence, the overall time of the procedure in the entire algorithm is  $O(n_s)$ . Consider a subproblem  $([i', j'], [i, j], V_t(i, j))$ . To solve it, after spending  $O(|V_t(i, j)|)$  time, we either reduce the problem to another subproblem in which the first interval is at most half the size of  $[i', j']$  and the third gateway set is still  $V_t(i, j)$ , or reduce it to two sub-problems such that each of them has the first interval at most half the size of  $[i', j']$  and the third gateway sets of the two sub-problems are two disjoint subsets of  $V_t(i, j)$ . Hence, if we consider the algorithm as a tree structure, the height of the tree is  $O(\log n_s)$  and the total time we spend on each level of the tree is  $O(n_t)$ . Therefore, the overall time of the algorithm is  $O(n_s + n_t \log n_s)$ .

The above describes our algorithm on the gateways of  $s$  in the first quadrant of  $s$ . We run the same algorithm for all quadrants of  $s$ , and for each quadrant, we will find an  $s$ - $t$  path. Finally, we return the path with the smallest length as our solution.

► **Lemma 11.** The running time of the query algorithm is  $O(\log n + n_s + n_t \log n_s)$ .

#### 4 Reducing the Query Time to $O(\log n)$

Since both  $n_s$  and  $n_t$  are  $O(\log n)$ , the query time of Lemma 11 is  $O(\log n \log \log n)$ . To further reduce it to  $O(\log n)$ , we need to change our graph  $G$  to a slightly larger graph  $G_1$  such that  $t$  only needs  $O(\log n / \log \log n)$  gateways while  $s$  still has  $O(\log n)$  gateways, i.e.,  $n_s = O(\log n)$  and  $n_t = O(\log n / \log \log n)$ . To this end, we introduce more Steiner points on the cut-lines. A similar idea was also used in [6] to reduce the number of gateways to  $O(\sqrt{\log n})$ . However, since we are allowed to have more gateways than  $O(\sqrt{\log n})$ , we do not need as many Steiner points as those in [6], which is the reason why we use less preprocessing.

Specifically, comparing with  $G$ , the new graph  $G_1$  has the following changes. As in [6], we first define “super-levels”. Recall that the cut-line tree  $\mathcal{T}$  has  $O(\log n)$  levels (with the root at the first level). We further partition all levels of the tree into  $O(\log n / \log \log n)$  super-levels: For any  $i$ , the  $i$ -th super-level contains the levels from  $(i - 1) \cdot \log \log n + 1$  to  $i \cdot \log \log n$ . Hence, each super-level has at most  $\log \log n$  levels.

Let  $u$  be a node at the highest level of the  $i$ -th super level of  $\mathcal{T}$ . Let  $\mathcal{T}_u$  be the sub-tree of  $\mathcal{T}$  rooted at  $u$  excluding the nodes outside the  $i$ -th level (thus  $\mathcal{T}_u$  has at most  $\log n - 1$  nodes). Recall that  $u$  is associated with a subset  $\mathcal{V}(u)$  of polygon vertices and each vertex  $v \in \mathcal{T}_u$  is associated with a cut-line  $l(v)$ . For each point  $p \in \mathcal{V}(u)$  and each vertex  $v \in \mathcal{T}_u$ , if  $p$  is horizontally visible to  $l(v)$ , then  $p$  defines a *type-3* Steiner point on  $l(v)$ . In this way,  $p$  defines  $O(\log n)$  type-3 Steiner points on the cut-lines of  $\mathcal{T}_u$  (in contrast,  $p$  defines only  $O(\log \log n)$  type-2 Steiner points on the cut-lines of  $\mathcal{T}_u$  in our original graph  $G$ ). Hence, each polygon vertex  $p$  defines a total of  $O(\log^2 n / \log \log n)$  type-3 Steiner points since  $\mathcal{T}$  has  $O(\log n / \log \log n)$  super-levels. The total number of type-3 Steiner points on all cut-lines is  $O(n \log^2 n / \log \log n)$ . Note that each type-2 Steiner point in our original graph  $G$  becomes a type-3 Steiner point. For convenience of discussion, those type-3 Steiner points of  $G_1$  that are originally type-2 Steiner points of  $G$  are also called type-2 Steiner points of  $G_1$ .

Type-1 Steiner points are defined in the same way as before, so their number is  $O(n)$ . We still use  $\mathcal{V}_1$  to denote the set of all type-1 Steiner points and all polygon vertices. We use  $\mathcal{V}_2$  to denote the set of all type-2 Steiner points of  $G_1$ .

The edges of  $G_1$  are defined with respect to all Steiner points in the same way as  $G$ . We omit the details. In summary,  $G_1$  has  $O(n \log^2 n / \log \log n)$  vertices and edges. Note that the original graph  $G$  is a sub-graph of  $G_1$  in that every vertex of  $G$  is also a vertex of  $G_1$  and each path of  $G$  corresponds to a path in  $G_1$  with the same length.

Consider a query point  $t$ . The gateway set  $V_g^1(t, G_1)$  is defined the same as before, and thus its size is  $O(1)$ . Thanks to more Steiner points, the size of  $V_g^2(t, G_1)$  can now be reduced to  $O(\log n / \log \log n)$ . Specifically,  $V_g^2(t, G_1)$  is defined as follows (similar to that in [6]).

As in [6], we first define the *relevant projection cut-lines* of  $t$ . We only discuss the right side of  $t$ , and the left side is symmetric. Recall that  $t$  has at most one projection cut-line in each level of  $\mathcal{T}$ . Among all projection cut-lines that are in the same super-level, the one closest to  $t$  is called a *relevant projection cut-line* of  $t$ . Since there are  $O(\log n / \log \log n)$  super-levels and each super-level has at most one relevant projection cut-line to the right of  $t$ ,  $t$  has  $O(\log n / \log \log n)$  relevant projection cut-lines. For each such cut-line  $l$ , the type-3 Steiner point (if any) immediately above (resp., below) the horizontal projection  $t'$  of  $t$  on  $l$  is included in  $V_g^2(t, G_1)$  if it is visible to  $t'$ . Thus,  $|V_g^2(t, G_1)| = O(\log n / \log \log n)$ .

By Lemma 11, if  $|V_g^2(t, G_1)| = O(\log n / \log \log n)$ , the query time becomes  $O(\log n)$  as long as  $|V_g^2(s, G_1)| = O(\log n)$ . This implies that for  $s$ , we can simply use its original gateway set on type-2 Steiner points, i.e.,  $V_g^2(s, G_1) = V_g^2(s, G)$ . As will be clear later, this will help save time and space in the preprocessing. We also define  $V_g^1(s, G_1)$  in the same way as before.



► **Lemma 12.** *For any two query points  $s$  and  $t$ , if there does not exist a trivial shortest  $s$ - $t$  path, then there is a shortest  $s$ - $t$  path containing a gateway of  $s$  and a gateway of  $t$ .*

► **Lemma 13.** *With  $O(n \log^3 n / \log \log n)$  time and  $O(n \log^2 n / \log \log n)$  space preprocessing, we can compute  $V_g(s, G_1)$  and  $V_g(t, G_1)$  in  $O(\log n)$  time for any two query points  $s$  and  $t$ .*

In the preprocessing, for each vertex  $q$  of  $G_1$  (which is also considered as a point in  $\mathcal{P}$ ), we compute a shortest path tree  $T(q)$  but only for the points in  $\mathcal{V}_1 \cup \mathcal{V}_2$  using the algorithm [24, 25]. Since  $|\mathcal{V}_1 \cup \mathcal{V}_2| = O(n \log n)$ ,  $T(p)$  has  $O(n \log n)$  vertices and can be computed in  $O(n \log^2 n)$  time [24, 25]. Since  $G_1$  has  $O(n \log^2 n / \log \log n)$  vertices, the preprocessing takes  $O(n^2 \log^4 n / \log \log n)$  time and  $O(n^2 \log^3 n / \log \log n)$  space in total.

With Lemma 11 and the new gateway sets, we can reduce the query time to  $O(\log n)$ .

Using the techniques in [6], we can further reduce the complexities of the preprocessing so that they are functions of  $h$ , in addition to  $O(n)$ , as shown in the following theorem.

► **Theorem 14.** *With  $O(n + h^2 \log^4 h / \log \log h)$  time and  $O(n + h^2 \log^3 h / \log \log h)$  space preprocessing, given  $s$  and  $t$ , we can compute their shortest path length in  $O(\log n)$  time and an actual shortest  $s$ - $t$  path can be output in time linear in the number of edges of the path.*

---

## References

- 1 A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilbur. Geometric Applications of a Matrix-Searching Algorithm. *Algorithmica*, 2:195–208, 1987.
- 2 S.W. Bae and H. Wang.  $L_1$  shortest path queries in simple polygons, 2018. [arXiv:1809.07481](https://arxiv.org/abs/1809.07481).
- 3 R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *International Journal of Computational Geometry and Applications*, 4(4):475–481, 1994.
- 4 B. Chazelle. Triangulating a Simple Polygon in Linear Time. *Discrete and Computational Geometry*, 6:485–524, 1991.
- 5 D.Z. Chen, O. Daescu, and K.S. Klenk. On geometric path query problems. *International Journal of Computational Geometry and Applications*, 11(6):617–645, 2001.
- 6 D.Z. Chen, R. Inkulu, and H. Wang. Two-Point  $L_1$  Shortest Path Queries in the Plane. *Journal of Computational Geometry*, 1:473–519, 2016.
- 7 D.Z. Chen, K.S. Klenk, and H.-Y.T. Tu. Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM Journal on Computing*, 29(4):1223–1246, 2000.
- 8 D.Z. Chen and H. Wang. Computing  $L_1$  Shortest Paths Among Polygonal Obstacles in the Plane. *Algorithmica*, 2019. <https://doi.org/10.1007/s00453-018-00540-x>.
- 9 D.Z. Chen and J. Xu. Shortest path queries in planar graphs. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 469–478, 2000.
- 10 Y.-J. Chiang and J.S.B. Mitchell. Two-point Euclidean shortest path queries in the plane. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 215–224, 1999.
- 11 K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in  $O(n \log^2 n)$  time. In *Proceedings of the 3rd Annual Symposium on Computational Geometry (SoCG)*, pages 251–257, 1987.
- 12 K. Clarkson, S. Kapoor, and P. Vaidya. Rectilinear shortest paths through polygonal obstacles in  $O(n \log^{2/3} n)$  time. Manuscript, 1988.
- 13 H.N. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. In *Proceedings of the 22nd International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 151–165, 1996.
- 14 H.A. ElGindy and P. Mitra. Orthogonal shortest route queries among axis parallel rectangular obstacles. *International Journal of Computational Geometry and Application*, 4:3–24, 1994.
- 15 J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72:868–889, 2006.

- 16 P. Gawrychowski, S. Mozes, O. Weimann, and C. Wulff-Nilsen. Better tradeoffs for exact distance oracles in planar graphs. In *Proceedings of the 29th Annual Symposium on Discrete Algorithms (SODA)*, pages 515–529, 2018.
- 17 L.J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989.
- 18 H. Guo, A. Maheshwari, and J.-R. Sack. Shortest Path Queries in Polygonal Domains. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, pages 200–211, 2008.
- 19 J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Information Processing Letters*, 38(5):231–235, 1991.
- 20 J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4(2):63–97, 1994.
- 21 J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999.
- 22 R. Inkulu and S. Kapoor. Planar rectilinear shortest path computation using corridors. *Computational Geometry: Theory and Applications*, 42(9):873–884, 2009.
- 23 M.M. Klawe and D.J. Kleitman. An Almost Linear Time Algorithm for Generalized Matrix Searching. *SIAM Journal on Discrete Mathematics*, 3:81–97, 1990.
- 24 J.S.B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. Abstracts of the *1st Canadian Conference on Computational Geometry*, 1989.
- 25 J.S.B. Mitchell.  $L_1$  shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.
- 26 S. Mozes and C. Sommer. Exact distance oracles for planar graphs. In *Proceedings of the 23rd Annual Symposium on Discrete Algorithms (SODA)*, pages 209–222, 2012.



# Near-Optimal Algorithms for Shortest Paths in Weighted Unit-Disk Graphs\*

Haitao Wang

Department of Computer Science, Utah State University, Logan, UT 84322, USA  
<https://cs.usu.edu/people/haitaowang>  
haitao.wang@usu.edu

Jie Xue

Department of Computer Science and Engineering, University of Minnesota – Twin Cities, Minneapolis, MN 55455, USA  
<http://cs.umn.edu/~xuexx193>  
xuexx193@umn.edu

---

## Abstract

We revisit a classical graph-theoretic problem, the *single-source shortest-path* (SSSP) problem, in weighted unit-disk graphs. We first propose an exact (and deterministic) algorithm which solves the problem in  $O(n \log^2 n)$  time using linear space, where  $n$  is the number of the vertices of the graph. This significantly improves the previous deterministic algorithm by Cabello and Jejíčič [CGTA'15] which uses  $O(n^{1+\delta})$  time and  $O(n^{1+\delta})$  space (for any small constant  $\delta > 0$ ) and the previous randomized algorithm by Kaplan et al. [SODA'17] which uses  $O(n \log^{12+o(1)} n)$  expected time and  $O(n \log^3 n)$  space. More specifically, we show that if the 2D offline insertion-only (additively-)weighted nearest-neighbor problem with  $k$  operations (i.e., insertions and queries) can be solved in  $f(k)$  time, then the SSSP problem in weighted unit-disk graphs can be solved in  $O(n \log n + f(n))$  time. Using the same framework with some new ideas, we also obtain a  $(1+\varepsilon)$ -approximate algorithm for the problem, using  $O(n \log n + n \log^2(1/\varepsilon))$  time and linear space. This improves the previous  $(1+\varepsilon)$ -approximate algorithm by Chan and Skrepetos [SoCG'18] which uses  $O((1/\varepsilon)^2 n \log n)$  time and  $O((1/\varepsilon)^2 n)$  space. Because of the  $\Omega(n \log n)$ -time lower bound of the problem (even when approximation is allowed), both of our algorithms are almost optimal.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Single-source shortest paths, Weighted unit-disk graphs, Geometric graph algorithms

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.60

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.05255>.

**Funding** The research of Jie Xue is supported, in part, by a Doctoral Dissertation Fellowship from the Graduate School of the University of Minnesota.

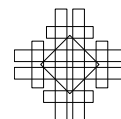
**Acknowledgements** The authors would like to thank Timothy Chan for the discussion, and in particular, for suggesting the algorithm for Lemma 7. Jie Xue would like to thank his advisor Ravi Janardan for his consistent advice and support.

## 1 Introduction

Given a set  $S$  of  $n$  points in the plane, its *unit-disk graph* is an undirected graph in which the vertices are points of  $S$  and two vertices are connected by an edge iff the (Euclidean) distance between them is at most 1. Unit-disk graphs can be viewed as the intersection graphs of equal-sized disks in the plane, and find many applications such as modeling the topology of ad-hoc communication networks. As an important class of geometric intersection graphs,

---

\* The work was partially done when Jie Xue was visiting Utah State University.



unit-disk graphs have been extensively studied in computational geometry. Many problems that are difficult in general graphs have been efficiently solved (exactly or approximately) in unit-disk graphs by exploiting their underlying geometric structures.

In this paper, we consider a classical graph-theoretic problem, the *single-source shortest-path* (SSSP) problem, in unit-disk graphs. Given an edge-weighted graph  $G = (V, E)$  and a source vertex  $s \in V$ , the SSSP problem aims to compute shortest paths from  $s$  to all other vertices in  $G$  (or equivalently a shortest-path tree from  $s$ ). In unit-disk graphs, there are two natural ways to weight the edges. The first way is to equally weight all the edge (usually called *unweighted* unit-disk graphs), while the second way is to assign each edge  $(a, b)$  a weight equal to the (Euclidean) distance between  $a$  and  $b$  (usually called *weighted* unit-disk graphs). The SSSP problem in a general graph has a trivial  $\Omega(|E|)$ -time lower bound, because specifying the edges of the graph already takes  $\Omega(|E|)$  time. However, this lower bound does not hold in unit-disk graphs. A unit-disk graph (either unweighted or weighted), though having quadratic number of edges in worst case (e.g., all the vertices are very close to each other), can be represented by only giving the locations of its vertices in the plane. This linear-complexity representation allows us to solve the SSSP problem without explicitly constructing the graph and hence beat the  $\Omega(|E|)$ -time lower bound.

In unweighted unit-disk graphs, the SSSP problem is relatively easy, and various algorithms are known for solving it *optimally* in  $O(n \log n)$  time [2, 3]. However, the weighted case is much more challenging. Despite of much effort made over years [2, 5, 9, 10, 12], state-of-the-art algorithms are still far away from being optimal. In this paper, we present new exact and approximation algorithms for the problem in weighted unit-disk graphs, which significantly improve the previous results and almost match the lower bound of the problem.

**Organization.** The remaining paper is organized as follows. In Sect. 1.1, we discuss the related work and our contributions. Sect. 1.2 presents some notations used throughout the paper. Our exact and approximation algorithms are given in Sect. 2 and 3, respectively. Due to the page limit, some lemma proofs are omitted but can be found in the full version [13].

## 1.1 Related work and our contributions

Besides the SSSP problem, many graph-theoretic problems have also been studied in unit-disk graphs, such as maximum independent set [11], maximum clique [6], distance oracle [5, 9], diameter computing [5, 9], all-pair shortest paths [3, 4], etc. Most of these problems have much more efficient solutions in unit-disk graphs than in general graphs.

The SSSP problem in unit-disk graphs has received a considerable attention in the last decades. The problem has an  $\Omega(n \log n)$ -time lower bound even when approximation is allowed, because deciding the connectivity of a unit-disk graph requires  $\Omega(n \log n)$  time [2]. In unweighted unit-disk graphs, at least two  $O(n \log n)$ -time SSSP algorithms were known [2, 3], which are optimal. If the vertices are pre-sorted by their  $x$ - and  $y$ -coordinates, the algorithm in [3] can solve the problem in  $O(n)$  time. In weighted unit-disk graphs, the SSSP problem was studied in [2, 5, 9, 10, 12]. Both exact and approximation algorithms were given to solve the problem in sub-quadratic time. For the exact case, the best known results are the deterministic algorithm by Cabello and Jejíč [2] which uses  $O(n^{1+\delta})$  time and  $O(n^{1+\delta})$  space (for any small constant  $\delta > 0$ ) and the randomized algorithm by Kaplan et al. [10] which uses  $O(n \log^{12+o(1)} n)$  expected time and  $O(n \log^3 n)$  space. For the approximation case, the best known result is the  $(1 + \varepsilon)$ -approximate algorithm by Chan and Skrepetos [5] which uses  $O((1/\varepsilon)^2 n \log n)$  time and  $O((1/\varepsilon)^2 n)$  space.

In this paper, we first propose an exact SSSP algorithm in weighted unit-disk graphs which uses  $O(n \log^2 n)$  time and  $O(n)$  space, significantly improving the results in [2, 10]. Using the same framework together with some new ideas, we also obtain a  $(1 + \varepsilon)$ -approximate algorithm which uses  $O(n \log n + n \log^2(1/\varepsilon))$  time and  $O(n)$  space, improving the result in [5]. Table 1 presents the comparison of our new algorithms with the previous results.

■ **Table 1** Summary of the previous and our new algorithms for SSSP in weighted unit-disk graphs.

| Type        | Source       | Time                                         | Space                           | Rand./Det.    |
|-------------|--------------|----------------------------------------------|---------------------------------|---------------|
| Exact       | [12]         | $O(n^{4/3+\delta})$                          | $O(n^{1+\delta})$               | Deterministic |
|             | [2]          | $O(n^{1+\delta})$                            | $O(n^{1+\delta})$               | Deterministic |
|             | [10]         | $O(n \log^{12+o(1)} n)$                      | $O(n \log^3 n)$                 | Randomized    |
|             | Corollary 9  | $O(n \log^2 n)$                              | $O(n)$                          | Deterministic |
| Approximate | [9]          | $O((1/\varepsilon)^3 n^{1.5} \sqrt{\log n})$ | $O((1/\varepsilon)^4 n \log n)$ | Deterministic |
|             | [5]          | $O((1/\varepsilon)^2 n \log n)$              | $O((1/\varepsilon)^2 n)$        | Deterministic |
|             | Corollary 15 | $O(n \log n + n \log^2(1/\varepsilon))$      | $O(n)$                          | Deterministic |

More specifically, our algorithms solve the SSSP problem in weighted unit-disk graphs by reducing it to the (2D) offline insertion-only additively-weighted nearest-neighbor (OIWNN) problem, in which we are given a sequence of operations each of which is either an insertion (inserting a weighted point in  $\mathbb{R}^2$  to the dataset) or a weighted nearest-neighbor query (asking for the additively-weighted nearest neighbor of a given query point in the dataset) and our goal is to answer all the queries. The reductions imply the following results.

- If the OIWNN problem with  $k$  operations can be solved in  $f(k)$  time, then the exact SSSP problem in weighted unit-disk graphs can be solved in  $O(n \log n + f(n))$  time.
- If the OIWNN problem with  $k_1$  operations in which at most  $k_2$  operations are insertions can be solved in  $f(k_1, k_2)$  time, then the  $(1 + \varepsilon)$ -approximate SSSP problem in weighted unit-disk graphs can be solved in  $O(n \log n + n \log(1/\varepsilon) + f(n, O(\varepsilon^{-2})))$  time.

Our time bounds in Table 1 are derived from the above results by arguing that  $f(k) = O(k \log^2 k)$  and  $f(k_1, k_2) = O(k_1 \log^2 k_2)$ . Therefore, the bottleneck of our algorithms in fact comes from the OIWNN problem.

As an immediate application, our approximation algorithm can be applied to improve the preprocessing time of the distance oracles in weighted unit-disk graphs given by Chan and Skrepetos [5] (see the full version [13] for the details).

## 1.2 Notations

**Basic notations.** Throughout the paper, the notation  $\|\cdot\|$  denotes the Euclidean norm; therefore, for two points  $a, b \in \mathbb{R}^2$ ,  $\|a - b\|$  is the Euclidean distance between  $a$  and  $b$ . For a point  $a \in \mathbb{R}^2$ , we use  $\odot_a$  to denote the unit disk (i.e., disk of radius 1) centered at  $a$ .

**Graphs.** Let  $G = (V, E)$  be an edge-weighted undirected graph. A path in  $G$  is represented as a sequence  $\pi = \langle z_1, \dots, z_t \rangle$  where  $z_1, \dots, z_t \in V$  and  $(z_i, z_{i+1}) \in E$  for all  $i \in \{1, \dots, t-1\}$ ; the *length* of  $\pi$  is the sum of the weights of the edges  $(z_1, z_2), \dots, (z_{t-1}, z_t)$ . For two vertices  $u, v \in V$ , we use  $\pi_G(u, v)$  to denote the shortest path from  $u$  to  $v$  in  $G$  and use  $d_G(u, v)$  to denote the length of  $\pi_G(u, v)$ . We say  $v' \in V$  is the  $u$ -predecessor of  $v$  if  $(v', v) \in E$  is the last edge of  $\pi_G(u, v)$ . For two paths  $\pi$  and  $\pi'$  in  $G$  where  $\pi$  is from  $u$  to  $v$  and  $\pi'$  is from  $v$  to  $w$ , we denote by  $\pi \circ \pi'$  the *concatenation* of  $\pi$  and  $\pi'$ , which is a path from  $u$  to  $w$  in  $G$ .

## 2 The exact algorithm

In this section, we describe our exact algorithm. Given a set  $S$  of  $n$  points in the plane and a source  $s \in S$ , our goal is to compute a shortest-path tree from  $s$  in the weighted unit-disk graph  $G$  induced by  $S$ . For all  $a \in S$ , we use  $\lambda_a \in S$  to denote the  $s$ -predecessor of  $a$ . Specifically, we aim to compute two tables  $\text{dist}[\cdot]$  and  $\text{pred}[\cdot]$  indexed by the points in  $S$ , where  $\text{dist}[a] = d_G(s, a)$  and  $\text{pred}[a] = \lambda_a$ .

We first briefly review how the well-known Dijkstra's algorithm computes shortest paths from a source  $s$  in a graph  $G$ . Initially, the algorithm sets all  $\text{dist}$ -values to infinity except  $\text{dist}[s] = 0$ , and sets  $A = S$ . Then it keeps doing the following procedure until  $A = \emptyset$ .

1. Pick the vertex  $c \in A$  with the smallest  $\text{dist}$ -value.
2. For all  $b \in A$  that are neighbors of  $c$ , update the value  $\text{dist}[b]$  using  $c$ , i.e.,  $\text{dist}[b] \leftarrow \min\{\text{dist}[b], \text{dist}[c] + w(c, b)\}$ , where  $w(c, b)$  is the weight of the edge  $(c, b)$ .
3. Remove  $c$  from  $A$ .

Directly applying Dijkstra's algorithm to solve the SSSP problem in a weighted unit-disk graph takes quadratic time, since the graph can have  $\Omega(n^2)$  edges in worst-case.

Our algorithm will follow the spirit of Dijkstra's algorithm in a high level and exploit many insights of unit-disk graphs in order to achieve a near-linear running time. First of all, we (implicitly) build a grid  $\Gamma$  on the plane, which consists of square cells with side-length  $1/2$  (a similar grid is also used in [3]). Assume for convenience that no point in  $S$  lies on a grid line, and hence each point in  $S$  is contained in exactly one cell of  $\Gamma$ . A *patch* of  $\Gamma$  is a square area consisting of  $5 \times 5$  cells of  $\Gamma$ . For a point  $a \in S$ , let  $\square_a$  denote the cell of  $\Gamma$  containing  $a$  and  $\boxplus_a$  denote the patch of  $\Gamma$  whose central cell is  $\square_a$ . For a set  $P$  of points in  $\mathbb{R}^2$  and a cell  $\square$  (resp., a patch  $\boxplus$ ) of  $\Gamma$ , define  $P_\square = P \cap \square$  (resp.,  $P_{\boxplus} = P \cap \boxplus$ ). We notice the following simple fact.

► **Fact 1.** *For all  $a \in S$ , we have  $S_{\square_a} \subseteq \text{NB}_G(a) \subseteq S_{\boxplus_a}$ , where  $\text{NB}_G(a)$  is the set of all neighbors of  $a$  in  $G$ .*

We compute and store  $S_\square$  (resp.,  $S_{\boxplus}$ ) for all cells  $\square$  (resp., patches  $\boxplus$ ) of  $\Gamma$  that contain at least one point in  $S$ . In addition, we associate pointers to each  $a \in S$  so that from  $a$  one can get access to the stored sets  $S_{\square_a}$  and  $S_{\boxplus_a}$ . The above preprocessing can be easily done in  $O(n \log n)$  time and  $O(n)$  space after computing  $\square_a$  for all  $a \in S$ . We give in the full version a method to compute  $\square_a$  for all  $a \in S$  in  $O(n \log n)$  time without using the floor function.

In order to present our algorithm, we first define a sub-routine `UPDATE` as follows. Suppose we are now at some point of the algorithm. If  $U$  and  $V$  are two subsets of  $S$ , then the procedure `UPDATE( $U, V$ )` conceptually does the following.

1.  $\text{dist}'[u] \leftarrow \text{dist}[u]$  for all  $u \in U$ .
2.  $p_v \leftarrow \arg \min_{u \in U \cap \odot_v} \{\text{dist}'[u] + \|u - v\|\}$  for all  $v \in V$ .
3. For all  $v \in V$ , if  $\text{dist}[v] > \text{dist}'[p_v] + \|p_v - v\|$ , then update  $\text{dist}[v] \leftarrow \text{dist}'[p_v] + \|p_v - v\|$  and  $\text{pred}[v] \leftarrow p_v$ .

In words, `UPDATE( $U, V$ )` updates the shortest-path information of the points in  $V$  using the shortest-path information of the points in  $U$ . We use lazy update by copying the  $\text{dist}[\cdot]$  table to  $\text{dist}'[\cdot]$  to guarantee that the order we consider the points in  $V$  does not influence the result of the update (note that  $U$  and  $V$  may not be disjoint). However, when  $U$  and  $V$  are not disjoint, lazy update may result in an inconsistency of shortest-path information, i.e.,  $\text{dist}[v] > \text{dist}[\text{pred}[v]] + \|\text{pred}[v] - v\|$  for some  $v \in V$  after `UPDATE( $U, V$ )`. This can happen when  $p_v \in U \cap V$ : for example, we update  $\text{dist}[v]$  to  $\text{dist}'[p_v] + \|p_v - v\|$  and at the same

time  $\text{dist}[p_v]$  also gets updated (hence  $\text{dist}[p_v] < \text{dist}'[p_v]$ ), then  $\text{dist}[v] > \text{dist}[p_v] + \|p_v - v\|$  after  $\text{UPDATE}(U, V)$ . We call such a phenomenon *data inconsistency*. Although  $\text{UPDATE}$  can result in data inconsistency in general, we shall guarantee it never happens in our algorithm.

The main framework of our algorithm is quite simple, which is presented in Algorithm 1. Similarly to Dijkstra's algorithm, we also maintain a subset  $A \subseteq S$  during the algorithm and pick the point  $c \in A$  with the smallest  $\text{dist}$ -value in each iteration (line 6). The difference is that, instead of using  $c$  to update (the shortest-path information of) its neighbors, our algorithm tries to use all points in  $A_{\square_c}$  to update their neighbors (line 8) and then remove them simultaneously from  $A$  (line 9). However, it is not guaranteed that the shortest-path information of all the points in  $A_{\square_c}$  is correct when  $c$  is picked. Therefore, before using the points in  $A_{\square_c}$  to update their neighbors, we use an extra procedure to "correct" the shortest-path information of these points, which is not needed in Dijkstra's algorithm. Surprisingly, we achieve this by simply updating the points in  $A_{\square_c}$  once using the current shortest-path information of their neighbors (line 7).

---

**Algorithm 1** SSSP( $S, s$ ).
 

---

```

1:  $\text{dist}[a] \leftarrow \infty$  for all  $a \in S$ 
2:  $\text{pred}[a] \leftarrow \text{NIL}$  for all  $a \in S$ 
3:  $\text{dist}[s] \leftarrow 0$ 
4:  $A \leftarrow S$ 
5: while  $A \neq \emptyset$  do ▷ Main loop
6:    $c \leftarrow \arg \min_{a \in A} \{\text{dist}[a]\}$ 
7:    $\text{UPDATE}(A_{\boxplus_c}, A_{\square_c})$  ▷ First update
8:    $\text{UPDATE}(A_{\square_c}, A_{\boxplus_c})$  ▷ Second update
9:    $A \leftarrow A \setminus A_{\square_c}$ 
10: return  $\text{dist}[\cdot]$  and  $\text{pred}[\cdot]$ 

```

---

The correctness of our algorithm is non-obvious. Suppose  $m$  is the number of the iterations in the main loop. Let  $c_i$  be the point  $c$  picked in the  $i$ -th iteration.

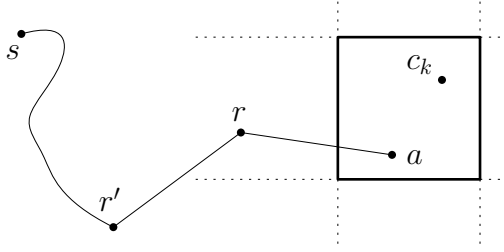
► **Fact 2.** *The points  $c_1, \dots, c_m$  belong to different cells in  $\Gamma$ .*

To prove the algorithm correctness, we first show that the  $\text{dist}$ -values of all points in  $S$  are correctly computed eventually. Clearly, during the entire algorithm, the  $\text{dist}$ -values can only decrease and never become smaller than the true shortest-path distances, i.e., we always have  $\text{dist}[a] \geq d_G(s, a)$  for all  $a \in S$ . Keeping this in mind, we prove the following lemma.

► **Lemma 3.** *Algorithm 1 has the following properties.*

- (1) *When the  $i$ -th iteration begins,  $\text{dist}[a] = d_G(s, a)$  for all  $a \in S$  with  $d_G(s, a) \leq d_G(s, c_i)$ .*
- (2) *After the first update of the  $i$ -th iteration,  $\text{dist}[a] = d_G(s, a)$  for all  $a \in S_{\square_{c_i}}$ .*
- (3) *When the  $i$ -th iteration ends,  $\text{dist}[a] = d_G(s, a)$  for all  $a \in S$  with  $\lambda_a \in S_{\square_{c_i}}$ .*

**Proof.** We first notice that the property (3) follows immediately from the property (2) due to the second update. Indeed, for a point  $a \in S$ , if  $\lambda_a \in S_{\square_{c_i}}$ , then  $a \in S_{\boxplus_{c_i}}$ . If  $a \in A_{\boxplus_{c_i}}$ , then the property (2) implies that the second update makes  $\text{dist}[a] = d_G(s, a)$ . If  $a \in S_{\boxplus_{c_i}} \setminus A_{\boxplus_{c_i}}$ , then  $a \in A_{\square_{c_j}}$  for some  $j < i$  (since  $a$  got removed from  $A$  in a previous iteration) and the property (2) guarantees that  $\text{dist}[a] = d_G(s, a)$  after the first update of the  $j$ -th iteration. As such, we only need to verify the first two properties. We achieve this using induction on  $i$ . The base case is  $i = 1$ . Note that  $c_1 = s$  and  $d_G(s, c_1) = 0$ . Thus, to see (1), we only need to guarantee that  $\text{dist}[s] = d_G(s, s) = 0$  when the first iteration begins, which is



■ **Figure 1** Illustrating points  $a$ ,  $r$ , and  $r'$ . The solid path is  $\pi_G(s, a)$ . The solid square is  $\square_{c_k}$ .

clearly true. After the first update of the first iteration, we have  $\text{dist}[a] = \|s - a\| = d_G(s, a)$  for all  $a \in S_{\square_{c_1}}$ , hence the property **(2)** is satisfied. Assume the lemma holds for all  $i < k$ , and we show it also holds in the  $k$ -th iteration.

To see the property **(1)**, let  $a \in S$  be a point such that  $d_G(s, a) \leq d_G(s, c_k)$ . Consider the moment when the  $k$ -th iteration begins. Assume for a contradiction that  $\text{dist}[a] > d_G(s, a)$  at that time. Suppose  $\pi_G(s, a) = \langle z_0, z_1, \dots, z_t \rangle$  where  $z_0 = s$  and  $z_t = a$ . Define  $j$  as the largest index such that  $\text{dist}[z_j] = d_G(s, z_j)$ . Note that  $j \in \{0, \dots, t-1\}$  because  $\text{dist}[s] = d_G(s, s)$  and  $\text{dist}[a] \neq d_G(s, a)$ . Therefore,  $\text{dist}[z_j] = d_G(s, z_j) < d_G(s, a) \leq d_G(s, c_k) \leq \text{dist}[c_k]$ . This implies  $z_j \notin A$  (otherwise it contradicts the fact that  $c_k$  is the point in  $A$  with the smallest  $\text{dist}$ -value). It follows  $z_j \in S_{\square_{c_i}}$  for some  $i < k$ , as it got removed from  $A$  in some previous iteration. Then by our induction hypothesis and the property **(3)**, we have  $\text{dist}[z_{j+1}] = d_G(s, z_{j+1})$  at the end of the  $i$ -th iteration and thus at the beginning of the  $k$ -th iteration, because  $\lambda_{z_{j+1}} = z_j$ . However, this contradicts the fact that  $\text{dist}[z_{j+1}] > d_G(s, z_{j+1})$ . As such,  $\text{dist}[a] = d_G(s, a)$  when the  $k$ -th iteration begins.

Next, we prove the property **(2)**. For convenience, in what follows, we use  $A$  to denote the set  $A$  during the  $k$ -th iteration (before line 9). We have  $S_{\square_{c_k}} = A_{\square_{c_k}}$ , since  $A = S \setminus (\bigcup_{i=1}^{k-1} S_{\square_{c_i}})$  and  $c_k \notin \square_{c_i}$  for all  $i < k$  by Fact 2. Let  $a \in A_{\square_{c_k}}$  be a point and  $r = \lambda_a$ . We want to show that  $\text{dist}[a] = d_G(s, a)$  after the first update of the  $k$ -th iteration. If  $r \notin A$ , then  $r$  got removed from  $A$  in the  $i$ -th iteration for some  $i < k$ , namely,  $r \in S_{\square_{c_i}}$ . By our induction hypothesis and the property **(3)**, we have  $\text{dist}[a] = d_G(s, a)$  at the end of  $i$ -th iteration (and thus in all the next iterations). So assume  $r \in A$  (this implies that  $r \neq s$  and thus  $\lambda_r$  exists). In this case, a key observation is that before the first update of the  $k$ -th iteration,  $\text{dist}[r] = d_G(s, r)$ . To see this, let  $r' = \lambda_r$  (e.g., see Fig. 1). Note that  $\|r' - a\| > 1$ , otherwise the path  $\pi_G(s, r') \circ \langle r', a \rangle$  would be shorter than  $\pi_G(s, r') \circ \langle r', r, a \rangle = \pi_G(s, a)$ , contradicting the fact that  $\pi_G(s, a)$  is the shortest path from  $s$  to  $a$ . It follows that

$$d_G(s, a) = d_G(s, r') + d_G(r', a) \geq d_G(s, r') + \|r' - a\| > d_G(s, r') + 1.$$

On the other hand, since  $a \in \square_{c_k}$ , we have

$$d_G(s, a) \leq d_G(s, c_k) + d_G(c_k, a) = d_G(s, c_k) + \|c_k - a\| \leq d_G(s, c_k) + 1.$$

Therefore,  $d_G(s, r') < d_G(s, c_k)$ , and by the property **(1)** we have  $\text{dist}[r'] = d_G(s, r')$  when the  $k$ -th iteration begins. This further implies  $r' \notin A$ , since  $\text{dist}[r'] = d_G(s, r') < d_G(s, c_k) = \text{dist}[c_k]$  when the  $k$ -th iteration begins. Hence,  $r'$  got removed from  $A$  in the  $i$ -th iteration for some  $i < k$ . Using our induction hypothesis and the property **(3)**, we have  $\text{dist}[r] = d_G(s, r)$  at the end of the  $i$ -th iteration (and thus in all the next iterations). Note that  $r \in \boxplus_{c_k}$ , because  $r \in \odot_a$ . We further have  $r \in A_{\boxplus_{c_k}}$ , as we assumed  $r \in A$ . Hence, the first update of the  $k$ -th iteration makes  $\text{dist}[a] = d_G(s, a)$ . This proves the property **(2)**. ◀

Lemma 3 implies that  $\text{dist}[a] = d_G(s, a)$  for all  $a \in S$  at the end of Algorithm 1. Indeed, any point  $a \in S$  belongs to  $S_{\square_{c_i}}$  for some  $i \in \{1, \dots, m\}$ , thus the property (2) of Lemma 3 guarantees  $\text{dist}[a] = d_G(s, a)$ . Next, we check the correctness of the  $\text{pred}[\cdot]$  table. We want  $\text{dist}[a] = \text{dist}[\text{pred}[a]] + \|\text{pred}[a] - a\|$  for all  $a \in S$ . However, as mentioned before, the sub-routine UPDATE in general may result in data inconsistency, making this equation false. The next lemma shows this can not happen in our algorithm.

► **Lemma 4.** *At any moment of Algorithm 1, we always have  $\text{dist}[a] = \text{dist}[\text{pred}[a]] + \|\text{pred}[a] - a\|$  for all  $a \in S$ .*

Now we see that Algorithm 1 correctly computes shortest paths from  $s$ . However, it is still not clear why simultaneously processing all points in one cell in each iteration makes our algorithm faster than the standard Dijkstra’s algorithm. In what follows, we focus on the time complexity of the algorithm. At this point, let us ignore the two UPDATE sub-routines and show how to efficiently implement the remaining part of the algorithm. In each iteration, all the work can be done in constant time except lines 6 and 9. To efficiently implement lines 6 and 9, we maintain the set  $A$  in a (balanced) binary search tree using the  $\text{dist}$ -values as keys. In this way, line 6 can be done in  $O(\log n)$  time, and lines 9 can be done in  $O(|S_{\square_c}| \cdot \log n)$  time. Note that whenever the  $\text{dist}$ -value of a point in  $A$  is updated, we also need to update the binary search tree in  $O(\log n)$  time. This occurs in the two UPDATE sub-routines, which has at most  $O(|S_{\square_c}| + |S_{\boxplus_c}|) = O(|S_{\boxplus_c}|)$  modifications of the  $\text{dist}$ -values. Therefore, the time for updating the binary search tree is  $O(|S_{\boxplus_c}| \cdot \log n)$ . To summarize, the time cost of the  $i$ -th iteration, without the UPDATE sub-routines, is  $O(|S_{\boxplus_{c_i}}| \cdot \log n)$ . Since  $\sum_{i=1}^m |S_{\boxplus_{c_i}}| \leq 25n$  by Fact 2, the overall time is  $O(n \log n)$ . In the following two sections, we shall consider the time complexities of the two UPDATE sub-routines. To efficiently implement the first UPDATE is relatively easy, while the second one is more challenging.

### 2.1 First update

In this section, we show how to implement the first update (line 7) in  $O(|S_{\boxplus_c}| \cdot \log n)$  time. As mentioned before, we can obtain the points in  $S_{\boxplus_c}$  using the pointer associated to  $c$ , and then further find the points in  $A_{\boxplus_c}$  and  $A_{\square_c}$ . After this, we do  $\text{dist}'[a] \leftarrow \text{dist}[a]$  for all  $a \in A_{\boxplus_c}$ . To implement  $\text{UPDATE}(A_{\boxplus_c}, A_{\square_c})$ , the critical step is to find, for every  $r \in A_{\square_c}$ , a point  $p \in A_{\boxplus_c} \cap \odot_r$  that minimizes  $\text{dist}'[p] + \|p - r\|$ . This is equivalent to searching the weighted nearest-neighbor of  $r$  in the unit disk  $\odot_r$  (if we regard  $A_{\boxplus_c}$  as a weighted dataset where the weight of each point equals its  $\text{dist}'$ -value). Unfortunately, it is currently not known how to efficiently solve this problem. Therefore, we need to exploit some special property of the problem in hand. An observation here is that  $c$  is the point in  $A_{\boxplus_c}$  with the smallest  $\text{dist}'$ -value and all the points in  $A_{\square_c}$  are of distance at most 1 to  $c$  (because  $c \in A_{\square_c}$ ). Using this observation, we prove the following key lemma.

► **Lemma 5.** *Before the first update of each iteration, for all  $r \in A_{\square_c}$ , we have*

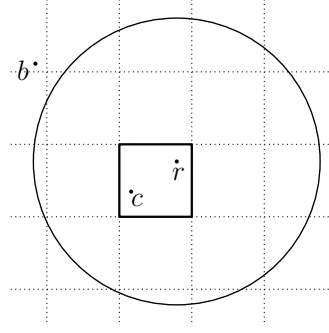
$$\arg \min_{a \in A_{\boxplus_c} \cap \odot_r} \{\text{dist}'[a] + \|a - r\|\} = \arg \min_{a \in A_{\boxplus_c}} \{\text{dist}'[a] + \|a - r\|\}.$$

**Proof.** Let  $p = \arg \min_{a \in A_{\boxplus_c} \cap \odot_r} \{\text{dist}'[a] + \|a - r\|\}$ . Define  $B = A_{\boxplus_c} \setminus (A_{\boxplus_c} \cap \odot_r)$ . It suffices to show that  $\text{dist}'[p] + \|p - r\| < \text{dist}'[b] + \|b - r\|$  for all  $b \in B$ . Fix a point  $b \in B$  (e.g., see Fig. 2). We have  $\|b - r\| > 1$  by construction. On the other hand, since  $r \in A_{\square_c}$ , we have  $c \in \odot_r$  and hence  $\|c - r\| \leq 1$ . Furthermore,  $\text{dist}'[c] \leq \text{dist}'[b]$ , because  $b \in A$  and  $c$  is the point in  $A$  with the smallest  $\text{dist}$ -value (as well as the smallest  $\text{dist}'$ -value). It follows that

$$\text{dist}'[p] + \|p - r\| \leq \text{dist}'[c] + \|c - r\| < \text{dist}'[b] + \|b - r\|,$$

where the first “ $\leq$ ” follows from the definition of  $p$  and the fact that  $c \in A_{\square_c} \cap \odot_r$ . ◀





■ **Figure 2** Illustrating the proof of Lemma 5. The solid square is  $\square_c$  and the solid circle is  $\odot_r$ .

The above lemma makes the problem easy. Indeed, for every  $r \in A_{\square_c}$ , we only need to find a point  $p \in A_{\odot_r}$  that minimizes  $\text{dist}'[p] + \|p - r\|$  and Lemma 5 guarantees that  $p \in \odot_r$ . This is just the standard (additively-)weighted nearest-neighbor search, which can be solved by building a weighted Voronoi Diagram (WVD) on  $A_{\odot_r}$  and then querying for each  $r \in A_{\square_c}$ . Building the WVD takes  $O(|A_{\odot_r}| \cdot \log |A_{\odot_r}|)$  time and linear space [8], and each query can be answered in  $O(\log |A_{\odot_r}|)$  time. The last step, updating the dist-values and pred-values of the points in  $A_{\square_c}$ , is easy. So the first update of the  $i$ -th iteration can be done in  $O(|S_{\odot_{c_i}}| \cdot \log n)$  time. Since  $\sum_{i=1}^m |S_{\odot_{c_i}}| \leq 25n$ , the total time for the first update is  $O(n \log n)$ .

## 2.2 Second update

In this section, we consider the second update (line 8) in Algorithm 1. Unfortunately, the trick used in the first update does not apply, which makes the second update more difficult. Here we design a more general algorithm, which can implement  $\text{UPDATE}(U, V)$  for arbitrary subsets  $U, V \subseteq S$  in  $O(f(k) + k \log k)$  time where  $k = |U| + |V|$  and  $f(k)$  is the time cost of the OIWNN problem with  $k$  operations (i.e., insertions and queries). The framework of the algorithm is presented in Algorithm 2. After copying  $\text{dist}[\cdot]$  to  $\text{dist}'[\cdot]$ , we first sort the points in  $U$  in increasing order of their  $\text{dist}'$ -values (line 2). Then we compute  $|U|$  disjoint subsets  $V_1, \dots, V_{|U|}$  of  $V$  (line 4), where  $V_i$  consists of the points contained in  $\odot_{u_i}$  but not contained in  $\odot_{u_j}$  for any  $j < i$ . Note that  $\bigcup_{i=1}^{|U|} V_i$  consists of all the points in  $V$  who have neighbors in  $U$ , and hence we only need to update the shortest-path information of these points. For each point  $v \in V_i$ , what we do is to find its weighted nearest-neighbor  $p$  in  $\{u_i, \dots, u_{|U|}\}$  where the weights are the  $\text{dist}'$ -values (line 9), and update the shortest-path information of  $v$  by attempting to use  $p$  as predecessor (line 10-12).

We first prove the correctness of Algorithm 2. Consider a point  $v \in V_i$ . The purpose of  $\text{UPDATE}(U, V)$  is to find the weighted nearest-neighbor of  $v$  in  $U \cap \odot_v$  (and use it to update the shortest-path information of  $v$ ), while what we find in line 9 is the weighted nearest-neighbor  $p$  in  $\{u_i, \dots, u_{|U|}\}$ . We notice that  $U \cap \odot_v \subseteq \{u_i, \dots, u_{|U|}\}$  because  $v \notin \odot_{u_j}$  for all  $j < i$  by the definition of  $V_i$ . Therefore, we only need to show that the point  $p$  computed by line 9 is contained in  $U \cap \odot_v$ .

► **Lemma 6.** *At line 9 of Algorithm 2, we have  $p \in U \cap \odot_v$ .*

**Proof.** Clearly, we have  $p \in U$  since  $B = \{u_i, \dots, u_{|U|}\} \subseteq U$ . It suffices to show  $\|p - v\| \leq 1$ . Assume for a contradiction that  $\|p - v\| > 1$ . We have  $\|u_i - v\| \leq 1$  since  $v \in V_i$ . Furthermore,  $\text{dist}'[u_i] \leq \text{dist}'[p]$  because  $p \in \{u_i, \dots, u_{|U|}\}$  and  $\text{dist}'[u_i] \leq \text{dist}'[u_j]$  for all  $j \geq i$ . Hence,

$$\text{dist}'[u_i] + \|u_i - v\| \leq \text{dist}'[u_i] + 1 \leq \text{dist}'[p] + 1 < \text{dist}'[p] + \|p - v\|,$$

which contradicts the fact that  $p$  is the weighted nearest-neighbor of  $v$  in  $\{u_i, \dots, u_{|U|}\}$ . ◀



**Algorithm 2** UPDATE( $U, V$ ).

---

```

1:  $\text{dist}'[u] \leftarrow \text{dist}[u]$  for all  $u \in U$ .
2: Sort the points in  $U = \{u_1, \dots, u_{|U|}\}$  such that  $\text{dist}'[u_1] \leq \dots \leq \text{dist}'[u_{|U|}]$ 
3: for  $i = 1, \dots, |U|$  do
4:    $V_i \leftarrow \{v \in V : v \in \odot_{u_i} \text{ and } v \notin \odot_{u_j} \text{ for all } j < i\}$ 
5:  $B \leftarrow \emptyset$ 
6: for  $i = |U|, \dots, 1$  do
7:    $B \leftarrow B \cup \{u_i\}$ 
8:   for  $v \in V_i$  do
9:      $p \leftarrow \arg \min_{b \in B} \{\text{dist}'[b] + \|b - v\|\}$ 
10:    if  $\text{dist}[v] > \text{dist}'[p] + \|p - v\|$  then
11:       $\text{dist}[v] \leftarrow \text{dist}'[p] + \|p - v\|$ 
12:       $\text{pred}[v] \leftarrow p$ 

```

---

Next, we analyze the time complexity of Algorithm 2. At the beginning, we need to sort the points in  $U$  in increasing order of their  $\text{dist}$ -values, which can be done in  $O(|U| \cdot \log |U|)$  time and hence  $O(k \log k)$  time. Algorithm 2 basically consists of two loops. We first consider the second loop (line 6-12). In this loop, what we do is weighted nearest-neighbor search on  $B$  (line 9) with insertions (line 7), where the weight of each point  $b \in B$  is  $\text{dist}'[b]$ . Note that all insertions and queries here are offline, since the points  $u_1, \dots, u_{|U|}$  and the sets  $V_1, \dots, V_{|U|}$  are already known before the loop. We have  $|U|$  insertions and  $|V|$  queries, and hence  $k$  operations in total. Recall that  $f(k)$  is the time for solving the OIWNN problem with  $k$  operations. So this loop takes  $f(k)$  time.

Now we consider the first loop (line 3-4). This loop requires us to compute  $V_i$ , the subset of  $V$  consisting of the points contained in  $\odot_{u_i}$  but not contained in  $\odot_j$  for all  $j < i$ , for  $i \in \{1, \dots, |U|\}$ . We have the following lemma. With the lemma, UPDATE( $U, V$ ) can be done in  $O(f(k) + k \log k)$  time.

► **Lemma 7.** *The first loop of Algorithm 2 takes  $O(k \log k)$  time where  $k = |U| + |V|$ .*

### 2.3 Putting everything together

As argued before, except the two UPDATE sub-routines, Algorithm 1 runs in  $O(n \log n)$  time. Section 2.1 shows that the first update can be done in  $O(|S_{\boxplus_c}| \cdot \log n)$  time. Section 2.2 demonstrates that the second update of each iteration can be done in  $O(f(k) + k \log k)$  time where  $k = |A_{\square_c}| + |A_{\boxplus_c}| = O(|S_{\boxplus_c}|)$  and  $f(k)$  is the time for solving the OIWNN problem with  $k$  operations. Noting the fact  $\sum_{i=1}^m |S_{\boxplus_{c_i}}| \leq 25n$ , we can conclude the following.

► **Theorem 8.** *Suppose the OIWNN problem with  $k$  operations can be solved in  $f(k)$  time, where  $f(k)/k$  is a non-decreasing function. Then there exists an SSSP algorithm in weighted unit-disk graphs with  $O(n \log n + f(n))$  running time, where  $n$  is the number of the vertices.*

**Proof.** According to our analysis and the fact  $\sum_{i=1}^m |S_{\boxplus_{c_i}}| \leq 25n$ , the overall time of Algorithm 1 is  $O(n \log n + \sum_{i=1}^m f(|S_{\boxplus_{c_i}}|))$ . Since  $f(k)/k$  is non-decreasing, we have

$$\sum_{i=1}^m f(|S_{\boxplus_{c_i}}|) = \sum_{i=1}^m |S_{\boxplus_{c_i}}| \cdot \frac{f(|S_{\boxplus_{c_i}}|)}{|S_{\boxplus_{c_i}}|} \leq \sum_{i=1}^m |S_{\boxplus_{c_i}}| \cdot \frac{f(n)}{n} \leq 25f(n).$$

Therefore, Algorithm 1 runs in  $O(n \log n + f(n))$  time. ◀

Using the standard logarithmic method [1] (see also [7] with an additional “bulk update” operation), we can solve the OIWNN problem (even the online version) with  $k$  operations in  $O(k \log^2 k)$  time using linear space, implying  $f(k) = O(k \log^2 k)$ . To explore the offline nature of our OIWNN problem, we give in the full version [13] an easier solution with the same performance. By plugging in this algorithm, we obtain the following corollary.

► **Corollary 9.** *There exists an SSSP algorithm in weighted unit-disk graphs with  $O(n \log^2 n)$  time and  $O(n)$  space, where  $n$  is the number of the vertices.*

### 3 The approximation algorithm

We now modify our algorithm framework in the last section (Algorithm 1) to obtain a  $(1 + \varepsilon)$ -approximate algorithm for any  $\varepsilon > 0$ . Again, let  $(S, s)$  be the input of the problem where  $|S| = n$  and  $G$  be the weighted unit-disk graph induced by  $S$ . Formally, a  $(1 + \varepsilon)$ -approximate algorithm computes two tables  $\text{dist}[\cdot]$  and  $\text{pred}[\cdot]$  indexed by the points in  $S$  such that  $\text{dist}[a] \leq (1 + \varepsilon) \cdot d_G(s, a)$  and  $\text{dist}[a] = \text{dist}[\text{pred}[a]] + \|\text{pred}[a] - a\|$  for all  $a \in S$ . Note that the two tables  $\text{dist}[\cdot]$  and  $\text{pred}[\cdot]$  enclose, for each point  $a \in S$ , a path from  $s$  to  $a$  in  $G$  that is a  $(1 + \varepsilon)$ -approximation of the shortest path from  $s$  to  $a$ .

Our algorithm is shown in Algorithm 3, which differs from our exact algorithm (Algorithm 1) as follows. First, in the initialization, we directly compute the  $\text{dist}$ -values and  $\text{pred}$ -values of all the neighbors of  $s$  in  $G$  (line 4-6); note that if  $a$  is a neighbor of  $s$  then the shortest path from  $s$  to  $a$  is  $\langle s, a \rangle$ , because  $G$  is a weighted unit-disk graph. Second, the first update in Algorithm 1 is replaced with two update procedures (line 10-11). Finally, the second update in Algorithm 1 is replaced with an approximate update (line 12) in Algorithm 3, which involves a new sub-routine APPROXUPDATE defined as follows. If  $U$  and  $V$  are two *disjoint* subsets of  $S$ , APPROXUPDATE( $U, V$ ) conceptually does the following.

1. For each  $v \in V$ , pick a point  $p_v \in U \cap \odot_v$  such that  $\text{dist}[p_v] + \|p_v - v\| \leq \text{dist}[u] + \|u - v\| + \varepsilon/2$  for all  $u \in U \cap \odot_v$ .
2. For all  $v \in V$ , if  $\text{dist}[v] < \text{dist}[p_v] + \|p_v - v\|$ , then  $\text{dist}[v] \leftarrow \text{dist}[p_v] + \|p_v - v\|$  and  $\text{pred}[v] \leftarrow p_v$ .

Unlike the UPDATE sub-routine, APPROXUPDATE cannot result in data inconsistency because we require  $U$  and  $V$  to be disjoint.

The basic idea of Algorithm 3 is similar to that of our exact algorithm. To verify the correctness of the algorithm, we need to introduce some notations. For  $a \in S$ , let  $l_a$  be the number of the edges on the path  $\pi_G(s, a)$  and define  $\tau_a = d_G(s, a) + (l_a - 1) \cdot (\varepsilon/2)$ . Also, as in Section 2, we use  $\lambda_a$  to denote the  $s$ -predecessor of  $a$ . We first notice the following fact.

► **Fact 10.** *For all  $a \in S$ ,  $\tau_a \leq (1 + \varepsilon) \cdot d_G(s, a)$ .*

**Proof.** Suppose  $\pi_G(s, a) = \langle z_0, z_1, \dots, z_{l_a} \rangle$  where  $z_0 = s$  and  $z_{l_a} = a$ . Note that  $\|z_i - z_{i+2}\| > 1$  for all  $i \in \{0, \dots, l_a - 2\}$ , for otherwise  $\langle z_0, z_1, \dots, \hat{z}_{i+1}, \dots, z_{l_a} \rangle$  would be a shorter path from  $s$  to  $a$  than  $\pi_G(s, a)$  (here  $\hat{z}_{i+1}$  means  $z_{i+1}$  is absent in the sequence). Therefore,  $d_G(s, a) \geq (l_a - 1)/2$ , and  $\tau_a = d_G(s, a) + (l_a - 1) \cdot (\varepsilon/2) \leq (1 + \varepsilon) \cdot d_G(s, a)$ . ◀

Let  $m$  be the number of iterations of the main loop and  $c_i$  be the point  $c$  picked in the  $i$ -th iteration. Note that Fact 2 also holds for Algorithm 3. Further, we have the following observation, which is similar to Lemma 3 in Section 2.

► **Lemma 11.** *Algorithm 3 has the following properties.*

- (1) *When the  $i$ -th iteration begins,  $\text{dist}[a] \leq \tau_a$  for all  $a \in S$  with  $\tau_a \leq \text{dist}[c_i]$ .*
- (2) *After line 11 of the  $i$ -th iteration,  $\text{dist}[a] \leq \tau_a$  for all  $a \in S_{\square_{c_i}}$ .*
- (3) *When the  $i$ -th iteration ends,  $\text{dist}[a] \leq \tau_a$  for all  $a \in S$  with  $\lambda_a \in S_{\square_{c_i}}$ .*

**Algorithm 3** APPROXSSSP( $S, s$ ).

---

```

1:  $\text{dist}[a] \leftarrow \infty$  for all  $a \in S$ 
2:  $\text{pred}[a] \leftarrow \text{NIL}$  for all  $a \in S$ 
3:  $\text{dist}[s] \leftarrow 0$ 
4: for  $a \in (S \setminus \{s\}) \cap \odot_s$  do
5:    $\text{dist}[a] \leftarrow \|s - a\|$ 
6:    $\text{pred}[a] \leftarrow s$ 
7:  $A \leftarrow S$ 
8: while  $A \neq \emptyset$  do ▷ Main loop
9:    $c \leftarrow \arg \min_{a \in A} \{\text{dist}[a]\}$ 
10:  UPDATE( $A_{\boxplus_c} \setminus A_{\square_c}, A_{\square_c}$ )
11:  UPDATE( $A_{\square_c}, A_{\square_c}$ )
12:  APPROXUPDATE( $A_{\square_c}, A_{\boxplus_c} \setminus A_{\square_c}$ ) ▷ Approximate update
13:   $A \leftarrow A \setminus A_{\square_c}$ 
14: return  $\text{dist}[\cdot]$  and  $\text{pred}[\cdot]$ 

```

---

By the above lemma, we see that  $\text{dist}[a] \leq \tau_a$  for all  $a \in S$  at the end of Algorithm 3. Indeed, any point  $a \in S$  belongs to  $S_{\square_{c_i}}$  for some  $i \in \{1, \dots, m\}$ , thus the property (2) of Lemma 11 guarantees that  $\text{dist}[a] \leq \tau_a$ . Using Fact 10, we further conclude that  $\text{dist}[a] \leq (1 + \varepsilon) \cdot d_G(s, a)$  for all  $a \in S$  at the end of Algorithm 3. Next, we need to check the correctness of the  $\text{pred}[\cdot]$  table. We want  $\text{dist}[a] = \text{dist}[\text{pred}[a]] + \|\text{pred}[a] - a\|$  for all  $a \in S$ . As mentioned in Section 2, the procedure UPDATE( $U, V$ ) may result in data inconsistency, namely  $\text{dist}[v] > \text{dist}[\text{pred}[v]] + \|\text{pred}[v] - v\|$  for some  $v \in V$ , when  $U$  and  $V$  are not disjoint. In Algorithm 3, the only place where this can happen is line 11 (note that the UPDATE sub-routine in line 10 acts on two disjoint sets). However, the following lemma shows that even line 11 cannot result in data inconsistency.

► **Lemma 12.** *After line 11 of each iteration in Algorithm 3, we have  $\text{dist}[a] \leq \text{dist}[b] + \|b - a\|$  for all  $a, b \in A_{\square_c}$ . In particular, at any moment of Algorithm 3, we always have  $\text{dist}[a] = \text{dist}[\text{pred}[a]] + \|\text{pred}[a] - a\|$  for all  $a \in S$ .*

The correctness of Algorithm 3 is thus proved. Later, the first statement of Lemma 12 will also be used to obtain an efficient implementation of the approximate update.

Next, we consider the time complexity of Algorithm 3. Using the same argument as in Section 2, we see that the running time of Algorithm 3 without line 10-12 is  $O(n \log n)$ . Line 10 can be implemented using the same method as in Section 2.1, namely building a WVD on the points in  $A_{\boxplus_c} \setminus A_{\square_c}$  and querying for each point in  $A_{\square_c}$  (the correctness follows from the argument in Section 2.1). Also, line 11 can be implemented in this way, because the points in  $A_{\square_c}$  are pairwise adjacent in  $G$ . Therefore, the total running time for line 10 and 11 is  $O(n \log n)$ . It suffices to analyze the time cost of line 12, the approximate update.

### 3.1 Approximate update

In order to implement the approximate update (line 12) in Algorithm 3, we (implicitly) build another grid  $\Gamma'$  on the plane, which consists of square cells with side-length  $\varepsilon/8$ . To avoid confusion, we use  $\blacksquare$  to denote a cell in  $\Gamma'$ . For a point  $a \in S$ , let  $\blacksquare_a$  denote the cell in  $\Gamma'$  containing  $a$ . For a set  $P$  of points in  $\mathbb{R}^2$  and a cell  $\blacksquare$  in  $\Gamma'$ , define  $P_{\blacksquare} = P \cap \blacksquare$ .

Line 12 of Algorithm 3 is  $\text{APPROXUPDATE}(A_{\square_c}, A_{\boxplus_c} \setminus A_{\square_c})$ . Let  $U = A_{\square_c}$  and  $V = A_{\boxplus_c} \setminus A_{\square_c}$ . We shall use two special properties of the set  $U$ : **(i)** all the points in  $U$  are contained in one cell in  $\Gamma$  and **(ii)**  $\text{dist}[u] \leq \text{dist}[u'] + \|u' - u\|$  for all  $u, u' \in U$  before the procedure  $\text{APPROXUPDATE}(U, V)$ , which follows from Lemma 12. Our algorithm for implementing  $\text{APPROXUPDATE}(U, V)$  is shown in Algorithm 4, which is a variant of Algorithm 2. Here we no longer need the  $\text{dist}'[\cdot]$  table because  $U$  and  $V$  are disjoint.

---

**Algorithm 4**  $\text{APPROXUPDATE}(U, V)$ .

---

```

1: Sort the points in  $U = \{u_1, \dots, u_{|U|}\}$  such that  $\text{dist}[u_1] \leq \dots \leq \text{dist}[u_{|U|}]$ 
2: for  $i = 1, \dots, |U|$  do
3:    $V_i \leftarrow \{v \in V : v \in \odot_{u_i} \text{ and } v \notin \odot_{u_j} \text{ for all } j < i\}$ 
4:  $U' \leftarrow \{u_j : j \geq k \text{ for all } k \text{ such that } u_k \in \blacksquare_{u_j}\}$ 
5:  $B \leftarrow \emptyset$ 
6: for  $i = |U|, \dots, 1$  do
7:   if  $u_i \in U'$  then  $B \leftarrow B \cup \{u_i\}$ 
8:   for  $v \in V_i$  do
9:      $p \leftarrow \arg \min_{b \in B} \{\text{dist}[b] + \|b - v\|\}$ 
10:    if  $p \notin \odot_v$  then  $p \leftarrow u_i$ 
11:    if  $\text{dist}[v] > \text{dist}[p] + \|p - v\|$  then
12:       $\text{dist}[v] \leftarrow \text{dist}[p] + \|p - v\|$ 
13:       $\text{pred}[v] \leftarrow p$ 

```

---

Recall the definition of the sub-routine  $\text{APPROXUPDATE}$  in Section 3. To verify the correctness of Algorithm 4, it suffices to show that just before line 11, the point  $p$  satisfies that  $p \in U \cap \odot_v$  and  $\text{dist}[p] + \|p - v\| \leq \text{dist}[r] + \|r - v\| + \varepsilon/2$  for all  $r \in U \cap \odot_v$ . The condition  $p \in \odot_v$  is clearly satisfied, because of line 10 (note that  $u_i \in \odot_v$ ). To verify the latter condition, we establish the following lemma.

► **Lemma 13.** *Just before line 11 of Algorithm 4, we have  $\text{dist}[p] + \|p - v\| \leq \text{dist}[r] + \|r - v\| + \varepsilon/2$  for all  $r \in U \cap \odot_v$ .*

For the time complexity of Algorithm 4, let  $k = |U| + |V|$ . The sorting in line 1 takes  $O(|U| \cdot \log |U|)$  time. The loop in line 2-3 can be implemented in  $O(k \log |U|)$  time using the same method as in Section 2.2. In line 4, we can compute the set  $U'$  in  $O(|U| \cdot \log(|S_{\boxplus_c}|/\varepsilon))$  time by grouping the points in  $U$  that belong to the same  $\Gamma'$ -cell (see the full version [13] for a more detailed discussion). The loop in line 6-13 is basically weighted nearest-neighbor search (line 9) with insertions (line 7). There are  $O(|V|)$  queries and  $O(|U'|)$  insertions. Note that  $|U'| = O(\varepsilon^{-2})$ , because of the property **(i)** of  $U$ . Therefore, if we use  $f(k_1, k_2)$  to denote the time cost for solving the OIWNN problem with  $k_1$  operations in which at most  $k_2$  operations are insertions, then the loop in line 6-13 takes  $f(k, O(\varepsilon^{-2}))$  time. In sum, the running time of Algorithm 4 is  $O(f(k, O(\varepsilon^{-2})) + k \log k)$  time. Therefore, the approximate update in Algorithm 3 can be done in  $O(f(|S_{\boxplus_c}|, O(\varepsilon^{-2})) + |S_{\boxplus_c}| \cdot \log(|S_{\boxplus_c}|/\varepsilon))$  time.

### 3.2 Putting everything together

Except the approximate update, Algorithm 1 runs in  $O(n \log n)$  time. Section 3.1 shows that the approximate update of each iteration can be done in  $O(f(k, O(\varepsilon^{-2})) + k \log k + k \log(1/\varepsilon))$  time where  $k = |A_{\square_c}| + |A_{\boxplus_c}| = O(|S_{\boxplus_c}|)$  and  $f(k_1, k_2)$  is the time for solving the OIWNN problem with  $k_1$  operations in which at most  $k_2$  operations are insertions. Noting the fact  $\sum_{i=1}^m |S_{\boxplus_{c_i}}| \leq 25n$ , we can conclude the following.

► **Theorem 14.** *Suppose the OIWNN problem with  $k_1$  operations in which at most  $k_2$  operations are insertions can be solved in  $f(k_1, k_2)$  time, and assume  $f(k_1, k_2)/k_1$  is a non-decreasing function of  $k_1$  for any fixed  $k_2$ . Then there exists a  $(1 + \varepsilon)$ -approximate SSSP algorithm in weighted unit-disk graphs with  $O(n \log n + n \log(1/\varepsilon) + f(n, O(\varepsilon^{-2})))$  running time, where  $n$  is the number of the vertices.*

We give in the full version [13] a linear-space algorithm with  $f(k_1, k_2) = O(k_1 \log^2 k_2)$ . By plugging in this algorithm, we can obtain the following corollary.

► **Corollary 15.** *For any  $\varepsilon > 0$ , there exists a  $(1 + \varepsilon)$ -approximate SSSP algorithm in weighted unit-disk graphs with  $O(n \log n + n \log^2(1/\varepsilon))$  time and  $O(n)$  space, where  $n$  is the number of the vertices.*

Our algorithm in the above corollary further improves the preprocessing time of the distance oracles in weighted unit-disk graphs given by Chan and Skrepetos [5] (see the full version [13] for details).

---

## References

- 1 Jou L. Bentley. Decomposable searching problems. *Information Processing Letters*, 8:244–251, 1979.
- 2 Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry: Theory and Applications*, 48:360–367, 2015.
- 3 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.
- 4 Timothy M. Chan and Dimitrios Skrepetos. All-Pairs Shortest Paths in Geometric Intersection Graphs. In *Proceedings of the 15th Algorithms and Data Structures Symposium (WADS)*, pages 253–264, 2017.
- 5 Timothy M Chan and Dimitrios Skrepetos. Approximate Shortest Paths and Distance Oracles in Weighted Unit-Disk Graphs. In *Proceedings of the 34th International Symposium on Computational Geometry (SoCG)*, pages 24:1–24:13, 2018.
- 6 Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.
- 7 Mark de Berg, Kevin Buchin, Bart M.P. Jansen, and Gerhard Woeginger. Fine-Grained Complexity Analysis of Two Classic TSP Variants. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 5:1–5:14, 2016.
- 8 Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987.
- 9 Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35:151–169, 2005.
- 10 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, Paul Seiferth, and Micha Sharir. Dynamic planar Voronoi diagrams for general distance functions and their algorithmic applications. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2495–2504, 2017.
- 11 Tomomi Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs. In *Proceedings of Japanese Conference on Discrete and Computational Geometry (JDCDG)*, pages 194–200, 1998.
- 12 Liam Roditty and Michael Segal. On bounded leg shortest paths problems. *Algorithmica*, 59:583–600, 2011.
- 13 Haitao Wang and Jie Xue. Near-Optimal Algorithms for Shortest Paths in Weighted Unit-Disk Graphs. *arXiv preprint*, 2019. [arXiv:1903.05255](https://arxiv.org/abs/1903.05255).



# Searching for the Closest-Pair in a Query Translate

**Jie Xue**

University of Minnesota, Twin Cities, Minneapolis, MN, USA  
<http://cs.umn.edu/~xuexx193>  
xuexx193@umn.edu

**Yuan Li**

Facebook Inc., Seattle, WA, USA  
lydxlx@fb.com

**Saladi Rahul**

University of Illinois at Urbana-Champaign, Urbana, IL, USA  
<http://cs.umn.edu/~rahuls>  
saladi.rahul@gmail.com

**Ravi Janardan**

University of Minnesota, Twin Cities, Minneapolis, MN, USA  
<http://cs.umn.edu/~janardan>  
janardan@umn.edu

---

## Abstract

We consider a range-search variant of the closest-pair problem. Let  $T$  be a fixed shape in the plane. We are interested in storing a given set of  $n$  points in the plane in some data structure such that for any specified translate of  $T$ , the closest pair of points contained in the translate can be reported efficiently. We present results on this problem for two important settings: when  $T$  is a polygon (possibly with holes) and when  $T$  is a general convex body whose boundary is smooth. When  $T$  is a polygon, we present a data structure using  $O(n)$  space and  $O(\log n)$  query time, which is asymptotically optimal. When  $T$  is a general convex body with a smooth boundary, we give a near-optimal data structure using  $O(n \log n)$  space and  $O(\log^2 n)$  query time. Our results settle some open questions posed by Xue et al. at SoCG 2018.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Closest pair, Range search, Geometric data structures, Translation query

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.61

**Related Version** A full version of the paper is available at <https://arxiv.org/abs/1807.09498>.

**Funding** The research of Jie Xue is supported, in part, by a Doctoral Dissertation Fellowship from the Graduate School of the University of Minnesota.

## 1 Introduction

The range closest-pair (RCP) problem, as a range-search version of the closest-pair problem, aims to store a given set  $S$  of  $n$  points in some data structure such that for a specified query range  $X \in \mathcal{X}$  chosen from a certain *query space*  $\mathcal{X}$ , the closest pair of points in  $S \cap X$  can be reported efficiently. As a range-search problem, the RCP problem is non-decomposable in the sense that even if the query range  $X$  can be written as  $X = X_1 \cup X_2$ , the closest-pair in  $S \cap X$  cannot be determined efficiently knowing the closest-pairs in  $S \cap X_1$  and  $S \cap X_2$ . The non-decomposability makes the problem quite challenging and interesting, as many traditional range-search techniques are inapplicable.

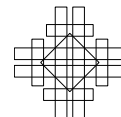
The RCP problem in  $\mathbb{R}^2$  has been well-studied over years [1, 4, 6, 7, 10, 11, 13, 14, 15]. Despite of much effort, the query ranges considered are still restricted to very simple shapes, typically orthogonal rectangles and halfplanes. It is then interesting to ask what if the query



© Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 61; pp. 61:1–61:15  
Leibniz International Proceedings in Informatics



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



ranges are of more general shapes. In this paper, we consider a new variant of the RCP problem in which the query ranges are *translates* of a fixed shape (which can be quite general). Formally, let  $\Gamma$  be a fixed shape in  $\mathbb{R}^2$  called *base shape* and  $\mathcal{L}_\Gamma$  be the collection of all translates of  $\Gamma$ . We investigate the RCP problem with the query space  $\mathcal{L}_\Gamma$  (or the  $\mathcal{L}_\Gamma$ -RCP problem). This type of query, which is for the first time mentioned in [15] (as an open question), is natural and well-motivated. First, in range-search problems, the query spaces considered are usually closed under translation; in this sense, the query space consisting of translates of a single shape seems the most “fundamental” query type. Some of the previously studied query ranges, e.g., quadrants and halfplanes [1, 7, 15], are in fact instances of translation queries (halfplanes can be viewed as translates of an “infinitely” large disc). Also, translation queries find motivation in practice. For instance, in many applications, the user may be interested in the information within a certain distance  $r$  from him/her. In this situation, the query ranges are discs of a fixed radius  $r$ , i.e., translates of a fixed disc; or more generally, if the distance  $r$  is considered under a general distance function induced by a norm  $\|\cdot\|$ , then the query ranges are translates of a  $\|\cdot\|$ -disc of radius  $r$ . Finally, there is another view of the translation queries: the base shape  $\Gamma$  can be viewed as static while the dataset is translating. With this view, a motivation of the translation queries is to monitor the information in a fixed region (i.e.,  $\Gamma$ ) for moving points (where the movement pattern only includes translation).

We investigate the problem in two important settings: when  $\Gamma$  is a polygon (possibly with holes) and when  $\Gamma$  is a general convex body whose boundary is smooth (i.e., through each point on the boundary there is a unique tangent line to  $\Gamma$ ). Our main goal is to design optimal or near-optimal data structures for the problems in terms of *space cost* and *query time*. The preprocessing of these data structures is left as an open question for future study.

Although we restrict the query ranges to be translates of a fixed shape, the problem is still challenging for a couple of reasons. First, the base shape  $\Gamma$  to be considered is quite general in both of our settings. When  $\Gamma$  is a polygon, it needs not be convex, and indeed can even have holes. In the case where  $\Gamma$  is a general convex body, we only need the aforementioned smoothness of its boundary. Second, we want the RCP data structures to be optimal or near-optimal, namely, use  $O(n \cdot \text{poly}(\log n))$  space and have  $O(\text{poly}(\log n))$  query time. This is usually difficult for a non-decomposable range-search problem.

## 1.1 Related work and our contributions

**Related work.** The closest-pair problem and range search are both classical topics; some surveys can be found in [3, 12]. The RCP problem in  $\mathbb{R}^2$  has been studied in prior work [1, 4, 6, 7, 10, 11, 13, 14, 15]. State-of-the-art RCP data structures for quadrant, strip, rectangle, and halfplane queries were given in the recent work [15]. The quadrant and halfplane RCP data structures are optimal (i.e., with linear space and logarithmic query time). The strip RCP data structure uses  $O(n \log n)$  space and  $O(\log n)$  query time, while the rectangle RCP data structure uses  $O(n \log^2 n)$  space and  $O(\log^2 n)$  query time. The work [13] considered a colored version of the RCP problem and gave efficient approximate data structures. The paper [14] studied an approximate version of the RCP problem in which the returned answer can be slightly outside the query range.

**Our contributions.** We investigate a new variant of the RCP problem in which the query ranges are translates of a fixed shape  $\Gamma$ . In the first half of the paper, we assume  $\Gamma$  is a fixed polygon (possibly with holes), and give an RCP data structure for  $\Gamma$ -translation queries using  $O(n)$  space and  $O(\log n)$  query time, which is asymptotically optimal. In the second



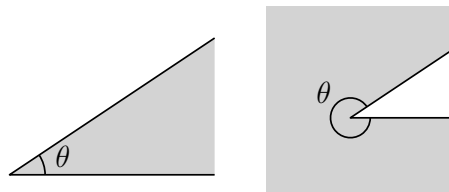
half of the paper, we assume  $\Gamma$  is a general convex body with a smooth boundary, and give a near-optimal RCP data structure for  $\Gamma$ -translation queries using  $O(n \log n)$  space and  $O(\log^2 n)$  query time. The  $O(\cdot)$  above hides constants depending on  $\Gamma$ . Our results settle some open questions posed in [15], e.g., the RCP problem with fixed-radius disc queries, etc. In order to design these data structures, we make nontrivial geometric observations and exploit the properties of the problem itself (i.e., we are searching for the *closest-pair* in a *translate*). Many of our intermediate results are of independent interest and can probably be applied to other related problems. We describe our key ideas and techniques in Section 1.3 after establishing relevant notations in Section 1.2.

**Organization.** Section 1.2 presents the notations and preliminaries used throughout the paper. Section 1.3 gives an overview of the techniques we use to solve the problems. In Section 2, we study the problem when  $\Gamma$  is a polygon. In Section 3, we study the problem when  $\Gamma$  is a general convex body with a smooth boundary. Due to limited space, some proofs and details are omitted; these can be found in the full version [16]. For the convenience of the reader, we give short proof sketches for some technical lemmas.

## 1.2 Preliminaries

**Basic notations and concepts.** For  $a, b \in \mathbb{R}^2$ , we use  $\text{dist}(a, b)$  to denote the Euclidean distance between  $a$  and  $b$ , and use  $[a, b]$  to denote the segment connecting  $a$  and  $b$ . The *length* of a pair  $\phi = (a, b)$  of points, denoted by  $|\phi|$ , is the length of the segment  $[a, b]$ , i.e.,  $|\phi| = \text{dist}(a, b)$ . For a shape  $\Gamma$  in  $\mathbb{R}^2$  and a point  $p \in \mathbb{R}^2$ , we denote by  $\Gamma_p$  the  $\Gamma$ -translate  $p + \Gamma$ . We write  $\mathcal{L}_\Gamma = \{\Gamma_p : p \in \mathbb{R}^2\}$ , i.e., the collection of all  $\Gamma$ -translates.

**Candidate pairs.** Let  $S$  be a set of points in  $\mathbb{R}^2$  and  $\mathcal{X}$  a collection of ranges. A *candidate pair* in  $S$  with respect to  $\mathcal{X}$  refers to a pair of points in  $S$  that is the closest-pair in  $S \cap X$  for some  $X \in \mathcal{X}$ . We denote by  $\Phi(S, \mathcal{X})$  the set of the candidate pairs in  $S$  w.r.t.  $\mathcal{X}$ .



■ **Figure 1** Examples of wedges and co-wedges.

**Wedges and co-wedges.** A *wedge* is a range in  $\mathbb{R}^2$  defined by an angle  $\theta \in (0, \pi)$ , which is the intersection of two halfplanes (see the left figure in Figure 1). A *co-wedge* is a range in  $\mathbb{R}^2$  defined by an angle  $\theta \in (\pi, 2\pi)$ , which is the union of two halfplanes (see the right figure in Figure 1). The boundary of a wedge or co-wedge  $W$  consists of two rays sharing a common initial point, called the two *branches* of  $W$ . When appropriate, we refer to wedges and co-wedges collectively as *(co-)wedges*.

**Convex bodies.** A *convex body* in  $\mathbb{R}^2$  refers to a compact convex shape with a nonempty interior. If  $C$  is a convex body in  $\mathbb{R}^2$ , we denote by  $\partial C$  the boundary of  $C$ , which is a simple cycle, and by  $C^\circ$  the interior of  $C$ , i.e.,  $C^\circ = C \setminus \partial C$ .

The following two lemmas will be used in various places in this paper.

► **Lemma 1.** *Let  $\Gamma$  be a fixed bounded shape in  $\mathbb{R}^2$ , and  $\mu > 0$  be a constant. Also, let  $S$  be a set of points in  $\mathbb{R}^2$ . Then for any point  $p \in \mathbb{R}^2$ , either the closest-pair in  $S \cap \Gamma_p$  has length smaller than  $\mu$ , or  $|S \cap \Gamma_p| = O(1)$ .*

► **Lemma 2.** *Let  $S$  be a set of points in  $\mathbb{R}^2$  and  $\mathcal{X}$  be a collection of ranges in  $\mathbb{R}^2$ . Suppose  $(a, b), (a', b') \in \Phi(S, \mathcal{X})$  are two pairs such that the segments  $[a, b]$  and  $[a', b']$  cross. Then there exists  $X \in \mathcal{X}$  such that either  $X \cap \{a, b, a', b'\} = \{a, b\}$  or  $X \cap \{a, b, a', b'\} = \{a', b'\}$ .*

### 1.3 Overview of key ideas and techniques

When  $\Gamma$  is a polygon (possibly with holes), we solve the problem as follows. First, we use a grid-based approach to reduce the  $\mathcal{L}_\Gamma$ -RCP problem to the RCP problem with wedge/co-wedge translation queries and the range-reporting problem with  $\Gamma$ -translation queries. The range-reporting problem can be easily solved by again reducing to the wedge/co-wedge case. Therefore, it suffices to study the RCP problem with wedge/co-wedge translation queries. For both wedge and co-wedge translation queries, we solve the problem by using candidate pairs. Specifically, we store the candidate pairs and search for the answer among them. In this approach, the critical point is the number of the candidate pairs, which determines the performance of our data structures. For both wedge and co-wedge, we prove *linear* upper bounds on the number of the candidate pairs. Although the bounds are the same, the wedge case and co-wedge case require very different proofs, both of which are quite technical and may be of independent geometric interest. These upper bounds and the above-mentioned reduction are our main technical contributions for the polygonal case.

When  $\Gamma$  is a general convex body with a smooth boundary, we solve the problem as follows. First, exploiting the smoothness of  $\partial\Gamma$ , we show that “short” candidate pairs (i.e., of length upper bounded by some constant  $\tau$ ) cannot “cross” each other<sup>1</sup>. It immediately follows that there are only a linear number of short candidate pairs (because they form a planar graph). We try to store these short candidate pairs in a data structure  $\mathcal{D}_1$  such that the shortest one contained in any query  $\Gamma_q$  can be found efficiently. However, this is a nontrivial task, as  $\Gamma$  is quite general here. To this end, we reduce the task of “searching for the shortest pair in  $\Gamma_q$ ” to several point-location queries for  $q$  in planar subdivisions. We bound the complexity of these subdivisions (and thus the cost of  $\mathcal{D}_1$ ) by making geometric observations for convex translates and using properties of the pseudo-discs. Using  $\mathcal{D}_1$ , we can answer any query  $\Gamma_q$  in which the closest-pair is short. What if the closest-pair in  $\Gamma_q$  is long (i.e., of length greater than  $\tau$ )? In this case,  $\Gamma_q$  contains only  $O(1)$  points by Lemma 1. Therefore, if  $\mathcal{D}_1$  fails to find the answer, we can simply report the  $O(1)$  points contained in  $\Gamma_q$  and find the closest-pair by brute-force. The range-reporting is done by point location in the  $\leq k$ -level of a pseudo-disc arrangement. These are our main contributions for this part.

## 2 Translation RCP queries for polygons

Let  $\Gamma$  be a fixed polygon (possibly with holes). Assume the boundary of  $\Gamma$  has no self-intersection<sup>2</sup>. We investigate the  $\mathcal{L}_\Gamma$ -RCP problem (where the closest-pair is in terms of the Euclidean metric). Throughout this section,  $O(\cdot)$  hides constants depending on  $\Gamma$ . Our main result is the following theorem, to prove which is the goal of this section.

<sup>1</sup> We say two pairs  $(a, b)$  and  $(a', b')$  cross if the segments  $[a, b]$  and  $[a', b']$  cross.

<sup>2</sup> That is, the outer boundary and the boundaries of holes are disjoint simple cycles.

► **Theorem 3.** *Let  $\Gamma$  be a fixed polygon (possibly with holes) in  $\mathbb{R}^2$ . Then there is an  $O(n)$ -space  $\mathcal{L}_\Gamma$ -RCP data structure with  $O(\log n)$  query time.*

Let  $S$  be the given dataset in  $\mathbb{R}^2$  of size  $n$ . Suppose for convenience that the pairwise distances of the points in  $S$  are distinct (so the closest-pair in any subset of  $S$  is unique).

### 2.1 Reduction to (co-)wedge translation queries

Our first step is to reduce a  $\Gamma$ -translation RCP query to several wedge/co-wedge translation RCP queries and a range-reporting query. For a vertex  $v$  of  $\Gamma$  (either on the outer boundary or on the boundary of a hole), we define a wedge (or co-wedge)  $W^v$  as follows. Consider the two edges adjacent to  $v$  in  $\Gamma$ . These two edges define two (explementary) angles at  $v$ , one of which (say  $\sigma$ ) corresponds to the interior of  $\Gamma$  (while the other corresponds to the exterior of  $\Gamma$ ). Let  $W^v$  be the (co-)wedge defined by  $\sigma$  depending on whether  $\sigma < \pi$  or  $\sigma > \pi$ .

Let  $\mathcal{W}_\Gamma = \{W^v : v \text{ is a vertex of } \Gamma\}$ . Without loss of generality, suppose that the outer boundary of  $\Gamma$  consists of at least four edges, and so does the boundary of each hole<sup>3</sup>; with this assumption, no three edges of  $\Gamma$  are pairwise adjacent. For two edges  $e$  and  $e'$  of  $\Gamma$ , let  $\text{dist}(e, e')$  denote the minimum distance between one point on  $e$  and one point on  $e'$ . Define  $\delta = \min\{\text{dist}(e, e') : e \text{ and } e' \text{ are non-adjacent edges of } \Gamma\}$ . Clearly,  $\delta$  is a positive constant depending on  $\Gamma$  only. Let  $\square$  be a square of side-length less than  $\delta/\sqrt{2}$ . Due to the choice of  $\delta$ , for any  $q \in \mathbb{R}^2$ ,  $\square$  cannot intersect two non-adjacent edges of  $\Gamma_q$ . It follows that  $\square$  intersects at most two edges of  $\Gamma_q$  (as no three edges of  $\Gamma$  are pairwise adjacent); moreover, if  $\square$  intersects two edges, they must be adjacent. Thus,  $\square \cap \Gamma_q = \square \cap W_q$  for some  $W \in \mathcal{W}_\Gamma$ .

For a decomposable range-search problem (e.g. range reporting) on  $S$ , the above simple observation already allows us to reduce a  $\Gamma$ -translation query to (co-)wedge translation queries (roughly) as follows. Let  $G$  be a grid of width  $\delta/2$  on the plane. For a cell  $\square$  of  $G$ , we define  $S_\square = S \cap \square$ . Due to the decomposability of the problem, to answer a query  $\Gamma_q$  on  $S$ , it suffices to answer the query  $\Gamma_q$  on  $S_\square$  for all  $\square$  that intersect  $\Gamma_q$ . Since each cell  $\square$  of  $G$  is a square of side-length  $\delta/2$  (which is smaller than  $\delta/\sqrt{2}$ ), we have  $\square \cap \Gamma_q = \square \cap W_q$  for some  $W \in \mathcal{W}_\Gamma$  and thus  $S_\square \cap \Gamma_q = S_\square \cap W_q$ . In other words, the query  $\Gamma_q$  on each  $S_\square$  is equivalent to a (co-)wedge translation query for some (co-)wedge  $W \in \mathcal{W}_\Gamma$ . Applying this idea to range-reporting, we conclude the following.

► **Lemma 4.** *There exists an  $O(n)$ -space range-reporting data structure for  $\Gamma$ -translation queries, which has an  $O(\log n + k)$  query time, where  $k$  is the number of the reported points.*

However, the above argument fails for a non-decomposable range-search problem, since when the problem is non-decomposable, we are not able to recover efficiently the global answer even if the answer in each cell is known. Unfortunately, our RCP problem belongs to this category. Therefore, more work is required to do the reduction. We shall take advantage of our observation in Lemma 1. We still lay a planar grid  $G$ . But this time, we set the width of  $G$  to be  $\delta/4$ . A *quad-cell*  $\boxplus$  of  $G$  is a square consisting of  $2 \times 2$  adjacent cells of  $G$ . For a quad-cell  $\boxplus$  of  $G$ , let  $S_{\boxplus} = S \cap \boxplus$ . Note that the side-length of a quad-cell of  $G$  is  $\delta/2$ , and each cell of  $G$  is contained in exactly four quad-cells of  $G$ , so is each point in  $S$ . Consider a query range  $\Gamma_q \in \mathcal{L}_\Gamma$ . The following observation follows from Lemma 1.

► **Lemma 5.** *For a quad-cell  $\boxplus$  of  $G$  such that  $|S_{\boxplus} \cap \Gamma_q| \geq 2$ , let  $\phi_{\boxplus}$  be the closest-pair in  $S_{\boxplus} \cap \Gamma_q$ . Define  $\phi^*$  as the shortest element among all  $\phi_{\boxplus}$ . If the length of  $\phi^*$  is at most  $\delta/4$ , then  $\phi^*$  is the closest-pair in  $S \cap \Gamma_q$ ; otherwise  $|S \cap \Gamma_q| = O(1)$ .*

<sup>3</sup> If this is not the case, we can add a new vertex at the midpoint of each edge to “break” it into two.

Using the above observation, we are able to do the reduction.

► **Theorem 6.** *Let  $f, g : \mathbb{N} \rightarrow \mathbb{N}$  be increasing functions where  $f(a+b) \geq f(a) + f(b)$ . If for any  $W \in \mathcal{W}_\Gamma$  there is an  $O(f(n))$ -space  $\mathcal{L}_W$ -RCP data structure with  $O(g(n))$  query time, then there is an  $O(f(n) + n)$ -space  $\mathcal{L}_\Gamma$ -RCP data structure with  $O(g(n) + \log n)$  query time.*

**Proof.** For a quad-cell  $\boxplus$  of  $G$ , let  $m_{\boxplus}$  be the number of the points in  $S_{\boxplus}$ . First, we notice that there are  $O(n)$  quad-cells  $\boxplus$  of  $G$  such that  $m_{\boxplus} > 0$  since each point in  $S$  is contained in at most four quad-cells; we call them *nonempty* quad-cells. For each nonempty quad-cell  $\boxplus$  and each  $W \in \mathcal{W}_\Gamma$ , we build an  $\mathcal{L}_W$ -RCP data structure on  $S_{\boxplus}$ ; by assumption, this data structure uses  $O(f(m_{\boxplus}))$  space. Now observe that  $m_{\boxplus} \leq n$  for all  $\boxplus$  and  $\sum m_{\boxplus} \leq 4n$ . From the condition  $f(a+b) \geq f(a) + f(b)$ , it follows that  $\sum f(m_{\boxplus}) = O(f(n))$ . Since  $|\mathcal{W}_\Gamma| = O(1)$ , the total space cost of these data structures is  $O(f(n))$ . Besides these data structures, we also build a range-reporting data structure on  $S$  for  $\Gamma$ -translation queries. As argued in Lemma 4, this data structure uses  $O(n)$  space.

To answer a query  $\Gamma_q \in \mathcal{L}_\Gamma$ , we first find all nonempty quad-cells of  $G$  that intersect  $\Gamma_q$ . The number of these quad-cells is  $O(1)$ , as it is bounded by  $O(\Delta^2/\delta^2)$  where  $\Delta$  is the diameter of  $\Gamma$ . These quad-cells can be found in  $O(\log n)$  time (see [16]). For each such quad-cell  $\boxplus$ , we find  $W \in \mathcal{W}_\Gamma$  such that  $\boxplus \cap \Gamma_q = \boxplus \cap W_q$  and query the  $\mathcal{L}_W$ -RCP data structure built on  $S_{\boxplus}$  to obtain the closest-pair  $\phi_{\boxplus}$  in  $S_{\boxplus} \cap \Gamma_q$ , which takes  $O(g(m_{\boxplus}))$  time. Since only  $O(1)$  quad-cells are considered, the time for this step is  $O(g(n))$ . Once these  $\phi_{\boxplus}$  are computed, we take the shortest element  $\phi^*$  among them. If the length of  $\phi^*$  is at most  $\delta/4$ , then  $\phi^*$  is the closest-pair in  $S \cap \Gamma_q$  by Lemma 5 and we just report  $\phi^*$ . Otherwise,  $|S \cap \Gamma_q| = O(1)$  by Lemma 5. We then compute the  $O(1)$  points in  $S \cap \Gamma_q$  using the range-reporting data structure, and compute the closest-pair in  $S \cap \Gamma_q$  by brute-force (in constant time). Since the query time of the range-reporting data structure is  $O(\log n + k)$  and  $k = O(1)$  here, the overall query time is  $O(g(n) + \log n)$ , as desired. ◀

By the above theorem, it now suffices to give efficient RCP data structures for wedge and co-wedge translation RCP queries. We resolve these problems in the following two sections.

## 2.2 Handling wedge translation queries

Let  $W$  be a fixed wedge in  $\mathbb{R}^2$  and  $\theta \in (0, \pi)$  be the angle of  $W$ . We denote by  $r$  and  $r'$  the two branches of  $W$ . For convenience, assume the vertex of  $W$  is the origin, and thus the vertex of a  $W$ -translate  $W_p$  is the point  $p$ . In this section, we shall give an  $O(n)$ -space  $\mathcal{L}_W$ -RCP data structure with  $O(\log n)$  query time.

The key ingredient of our result is a nontrivial linear upper bound for the number of the candidate pairs in  $S$  with respect to  $\mathcal{L}_W$ . This generalizes a result in [7], and requires a much more technical proof. Before working on the proof, we first establish an easy fact.

► **Lemma 7.** *Let  $A \subseteq \mathbb{R}^2$  be a finite set. There exists a (unique) smallest  $W$ -translate (under the  $\subseteq$ -order) that contains  $A$ . Furthermore, a  $W$ -translate is the smallest  $W$ -translate containing  $A$  iff it contains  $A$  and its two branches both intersect  $A$ .*

We notice that if  $\phi = (a, b)$  is a pair of points in  $S$  and  $W_p$  is the smallest  $W$ -translate containing  $\{a, b\}$  described in Lemma 7, then  $\phi \in \Phi(S, \mathcal{L}_W)$  iff  $\phi$  is the closest-pair in  $S \cap W_p$ . Using Lemma 7, we define the following notions.

► **Definition 8.** *Let  $\phi = (a, b)$  be a pair of points in  $\mathbb{R}^2$ , and  $W_p$  be the smallest  $W$ -translate containing  $\{a, b\}$  described in Lemma 7. If  $p \notin \{a, b\}$  and the smallest angle of the triangle  $\triangle pab$  is  $\angle apb$ , then we say  $\phi$  is **steep**; otherwise, we say  $\phi$  is **flat**. See Figure 2 for examples.*

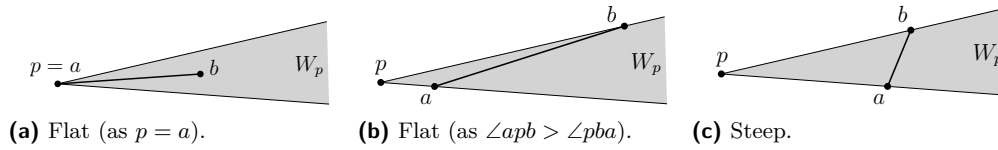


Figure 2 Examples of flat and steep pairs (the wedge  $W_p$  shown in the figures is the smallest  $W$ -translate containing  $\{a, b\}$  described in Lemma 7).

Our first observation is the following.

► **Lemma 9.** *If two candidate pairs  $\phi, \phi' \in \Phi(S, \mathcal{L}_W)$  cross, then either  $\phi$  or  $\phi'$  is steep.*

**Proof.** Suppose  $\phi = (a, b)$  and  $\phi' = (a', b')$ . Since  $\phi$  and  $\phi'$  cross, by Lemma 2 there exists some  $W_t \in \mathcal{L}_W$  whose intersection with  $\{a, b, a', b'\}$  is either  $\{a, b\}$  or  $\{a', b'\}$ ; assume  $W_t \cap \{a, b, a', b'\} = \{a, b\}$ . Let  $p \in \mathbb{R}^2$  be the point such that  $W_p$  is the smallest  $W$ -translate containing  $\{a, b\}$ . It follows that  $W_p \cap \{a, b, a', b'\} = \{a, b\}$ , because  $W_p \subseteq W_t$ . Let  $c$  be the intersection point of the segments  $[a, b]$  and  $[a', b']$ . Since  $a, b \in W_p$  and  $W_p$  is convex,  $c \in W_p$ . The two endpoints  $a', b'$  of the segment  $[a', b']$  are not contained in  $W_p$  (by assumption), but  $c \in W_p$ . Hence, the segment  $[a', b']$  intersects the boundary of  $W_p$  at two points, say  $a^*$  and

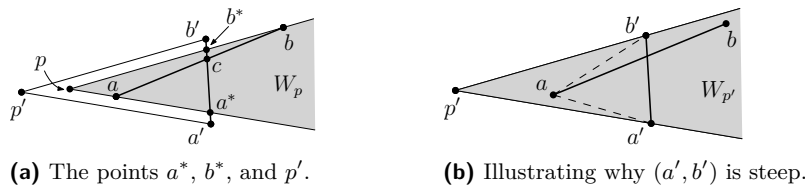


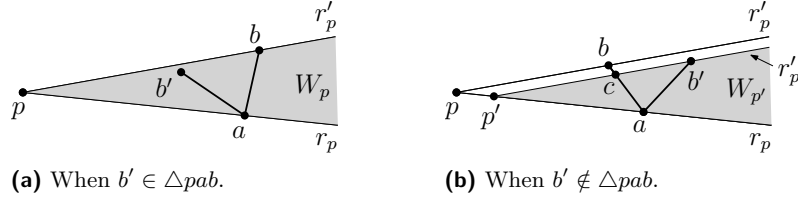
Figure 3 Illustrating Lemma 9.

$b^*$ ; assume  $a^*$  (resp.,  $b^*$ ) is the point adjacent to  $a'$  (resp.,  $b'$ ). Clearly, there exists a unique point  $p' \in \mathbb{R}^2$  such that  $\Delta pa^*b^* \subseteq \Delta p'a'b'$  and  $\Delta pa^*b^*$  is similar to  $\Delta p'a'b'$ . See Figure 3a. It is easy to see that  $W_{p'}$  is the smallest  $W$ -translate containing  $\{a', b'\}$ . Indeed,  $W_{p'}$  just corresponds to the angle  $\angle a'p'b'$ , so  $a'$  and  $b'$  lie on the two branches of  $W_{p'}$  respectively (see Figure 3b). Thus, by the criterion given in Lemma 7,  $W_{p'}$  is the smallest  $W$ -translate containing  $\{a', b'\}$ . Now we have  $p \in W_{p'}$ , which implies  $W_p \subseteq W_{p'}$  and  $a, b, a', b' \in W_{p'}$ . Since the segments  $[a, b]$  and  $[a', b']$  cross, one of  $a$  and  $b$  must lie in the triangle  $\Delta p'a'b'$ , say  $a \in \Delta p'a'b'$ . Note that  $\phi' = (a', b')$  is the closest-pair in  $W_{p'}$ , thus  $\text{dist}(a', b') < \text{dist}(a, a')$  and  $\text{dist}(a', b') < \text{dist}(a, b')$ . It follows that  $\angle a'ab' < \angle ab'a'$  and  $\angle a'ab' < \angle aa'b'$ . We further observe that  $\angle aa'b' < \angle p'a'b'$  and  $\angle ab'a' < \angle p'b'a'$ , and hence  $\angle a'ab' > \angle a'p'b'$ . Thus, we have  $\angle a'p'b' < \angle p'a'b'$  and  $\angle a'pb' < \angle p'b'a'$ , i.e.,  $\angle a'p'b'$  is the smallest angle of the triangle  $\Delta p'a'b'$ . As a result,  $\phi'$  is steep. ◀

Lemma 9 implies that the flat candidate pairs in  $\Phi(S, \mathcal{L}_W)$  do not cross each other. Therefore, the segments corresponding to the flat candidate pairs are edges of a planar graph with vertices in  $S$ , which gives a linear upper bound for the number of flat candidate pairs.

It now suffices to bound the number of steep candidate pairs in  $\Phi(S, \mathcal{L}_W)$ . Unfortunately, two steep candidate pairs (or even one steep candidate pair and one flat candidate pair) can cross, making the above non-crossing argument fail. Therefore, we need some new ideas.

► **Definition 10.** *Two pairs  $\phi, \phi' \in \Phi(S, \mathcal{L}_W)$  are **adjacent** if we can write  $\phi = (a, b)$  and  $\phi' = (a, b')$  such that  $b \neq b'$ ; we call  $\angle bab'$  the **angle** between  $\phi$  and  $\phi'$ , denoted by  $\text{ang}(\phi, \phi')$ .*



■ **Figure 4** Illustrating Lemma 11.

► **Lemma 11.** For adjacent  $\phi, \phi' \in \Phi(S, \mathcal{L}_W)$ , if  $\phi$  and  $\phi'$  are both steep, then  $\text{ang}(\phi, \phi') \geq \theta$ .

**Proof sketch.** Suppose  $\phi = (a, b)$  and  $\phi' = (a, b')$ . Let  $p \in \mathbb{R}^2$  be the point such that  $W_p$  is the smallest  $W$ -translate containing  $\{a, b, b'\}$  described in Lemma 7. We denote by  $r_p$  and  $r'_p$  the  $r$ -branch and  $r'$ -branch of  $W_p$ , respectively. Using Lemma 7 and the fact that  $\phi$  and  $\phi'$  are steep, we can deduce  $p \notin \{a, b, b'\}$ ; see the complete proof in [16] for details. We distinguish two cases:  $a$  is on the boundary of  $W_p$  or  $a$  is in the interior of  $W_p$ . Here we only discuss the first case. The second one is deferred to the complete proof.

Assume  $a$  is on the boundary of  $W_p$ . Since  $p \notin \{a, b, b'\}$ , we have  $a \neq p$ . Thus  $a$  must lie on exactly one of  $r_p$  and  $r'_p$ , say  $a \in r_p$ . Because  $W_p$  is the smallest  $W$ -translate containing  $\{a, b, b'\}$ , one of  $b$  and  $b'$  must lie on  $r'_p$  by the criterion given in Lemma 7. Without loss of generality, assume  $b \in r'_p$ . Using the criterion in Lemma 7 again, we see that  $W_p$  is also the smallest  $W$ -translate containing  $\{a, b\}$ . Thus,  $\phi$  is the closest-pair in  $S \cap W_p$  and in particular we have  $\text{dist}(a, b) < \text{dist}(b, b')$ . It follows that  $\angle ab'b < \angle bab' = \text{ang}(\phi, \phi')$ . If  $b' \in \triangle pab$ , we are done, because in this case  $\angle ab'b \geq \angle apb = \theta$  and thus  $\text{ang}(\phi, \phi') = \angle bab' > \angle ab'b \geq \angle apb = \theta$ . See Figure 4a for an illustration of this case. Next, assume  $b' \notin \triangle pab$ . This case is presented in Figure 4b. Let  $p' \in \mathbb{R}^2$  be the point such that  $W_{p'}$  is the smallest  $W$ -translate containing  $\{a, b'\}$ ; thus, we have  $W_{p'} \subseteq W_p$  and in particular  $p' \in W_p$ . Furthermore,  $p'$  must lie on the segment  $[p, a]$ , as  $a \in W_{p'}$ . Since  $\phi' = (a, b')$  is steep,  $a$  and  $b'$  lie on the two branches of  $W_{p'}$  respectively, and  $\angle p'b'a > \angle ap'b' = \theta$ . Clearly,  $b'$  lies on the  $r'$ -branch of  $W_{p'}$ , which we denote by  $r'_{p'}$ . Let  $c$  be the intersection point of  $r'_{p'}$  and the segment  $[a, b]$ . See Figure 4b. We then have  $\angle ab'b = \angle ab'c + \angle cb'b \geq \angle ab'c = \angle p'b'a > \angle ap'b' = \theta$ . Using the fact  $\angle ab'b < \angle bab' = \text{ang}(\phi, \phi')$  obtained before, we conclude  $\text{ang}(\phi, \phi') \geq \theta$ . ◀

For a point  $a \in S$ , consider the subset  $\Psi_a \subseteq \Phi(S, \mathcal{L}_W)$  consisting of all steep candidate pairs having  $a$  as one point. We claim  $|\Psi_a| = O(1)$ . Suppose  $\Psi_a = \{\psi_1, \dots, \psi_r\}$  where  $\psi_i = (a, b_i)$  and  $b_1, \dots, b_r$  are sorted in polar-angle order around  $a$ . By Lemma 11,  $\text{ang}(\psi_i, \psi_j) \geq \theta$  for any distinct  $i, j \in \{1, \dots, r\}$ . Since  $\sum_{i=1}^{r-1} \text{ang}(\psi_i, \psi_{i+1}) \leq 2\pi$ , we have  $r \leq 2\pi/\theta + 1 = O(1)$ . As such,  $\sum_{a \in S} |\Psi_a| = O(n)$ , implying that the number of steep candidate pairs is linear. As the numbers of flat and steep candidate pairs are both linear, we conclude the following.

► **Lemma 12.**  $|\Phi(S, \mathcal{L}_W)| = O(n)$ , where  $n = |S|$ .

Suppose  $\Phi(S, \mathcal{L}_W) = \{\phi_1, \dots, \phi_m\}$  where  $\phi_i = (a_i, b_i)$  and  $\phi_1, \dots, \phi_m$  are sorted in increasing order of their lengths. We have  $m = O(n)$  by Lemma 12. Now we only need a data structure which can report, for a query  $W_q \in \mathcal{L}_W$ , the smallest  $i$  such that  $a_i, b_i \in W_q$  (note that  $\phi_i$  is the closest-pair in  $S \cap W_q$ ). We design this data structure as follows. Let  $\tilde{W} = \{(x, y) : (-x, -y) \in W\}$ , which is a wedge obtained by rotating  $W$  around the origin with angle  $\pi$ . For a point  $p \in \mathbb{R}^2$ , it is clear that  $a_i, b_i \in W_p$  iff  $p \in \tilde{W}_{a_i} \cap \tilde{W}_{b_i}$ . Since the intersection of finitely many  $\tilde{W}$ -translates is a  $\tilde{W}$ -translate, we may write  $\tilde{W}_{a_i} \cap \tilde{W}_{b_i} = \tilde{W}_{c_i}$  for some  $c_i \in \mathbb{R}^2$ . It follows that  $\phi_i$  is contained in  $W_p$  iff  $p \in \tilde{W}_{c_i}$ . By successively



overlying  $\tilde{W}_{c_1}, \dots, \tilde{W}_{c_m}$ , we obtain a planar subdivision whose cells are  $\Sigma_1, \dots, \Sigma_m$  where  $\Sigma_i = \tilde{W}_{c_i} \setminus \bigcup_{j=1}^{i-1} \tilde{W}_{c_j}$ . This subdivision has  $O(m)$  complexity as overlaying a new  $\tilde{W}$ -translate can create at most two new vertices. The answer for a query  $W_q$  is  $i$  iff  $q \in \Sigma_i$ . Therefore, the problem can be solved by building on the subdivision an  $O(m)$ -space point-location data structure with  $O(\log m)$  query time. Since  $m = O(n)$ , we have the following conclusion.

► **Theorem 13.** *There is an  $O(n)$ -space  $\mathcal{L}_W$ -RCP data structure with  $O(\log n)$  query time.*

### 2.3 Handling co-wedge translation queries

Let  $C$  be a fixed co-wedge in  $\mathbb{R}^2$  and  $W$  be the *complementary* wedge of  $C$ , i.e., the closure of  $\mathbb{R}^2 \setminus C$ . We denote by  $r$  and  $r'$  the two branches of  $C$  (and also of  $W$ ). For convenience, assume  $r = \{(t, 0) : t \geq 0\}$  and  $r' = \{(\alpha t, t) : t \geq 0\}$  for some  $\alpha \in \mathbb{R}$ . With this assumption, the vertex of  $C$  (resp.,  $W$ ) is the origin and the vertex of  $C_p$  (resp.,  $W_p$ ) is  $p$  for all  $p \in \mathbb{R}^2$ . In this section, we present an  $O(n)$ -space  $\mathcal{L}_C$ -RCP data structure with  $O(\log n)$  query time.

Similar to the wedge case, the key step here is to establish a linear upper bound for  $|\Phi(S, \mathcal{L}_C)|$ . However, the techniques used here are very different. First of all, we exclude from  $\Phi(S, \mathcal{L}_C)$  the candidate pairs with respect to halfplanes. Let  $\mathcal{H}$  be the collection of halfplanes, and  $\Phi^* = \Phi(S, \mathcal{L}_C) \setminus \Phi(S, \mathcal{H})$ . It was shown in [1] that  $|\Phi(S, \mathcal{H})| = O(n)$ . Therefore, it suffices to prove that  $|\Phi^*| = O(n)$ . For a pair  $\phi = (a, b) \in \Phi^*$ , define its *associated*  $C$ -translate,  $\text{Ass}(\phi)$ , as the smallest  $W$ -translate containing  $\{a, b\}$  (Lemma 7). The pairs in  $\Phi^*$  and their associated  $C$ -translates has the following property.

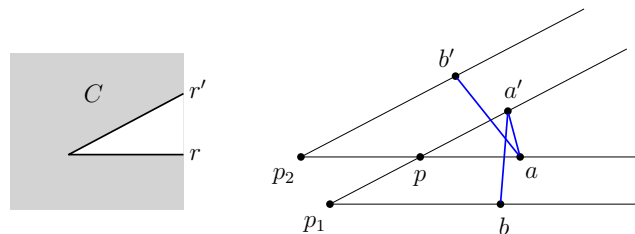
► **Lemma 14.** *Let  $\phi = (a, b) \in \Phi^*$  and  $C_p = \text{Ass}(\phi) \in \mathcal{L}_C$ . Then  $p \notin \{a, b\}$  and  $a, b$  lie on the two branches of  $C_p$  respectively. Furthermore,  $\phi$  is the closest-pair in  $S \cap C_p$ .*

Consider a pair  $\phi \in \Phi^*$  and its associated  $C$ -translate  $C_p = \text{Ass}(\phi)$ . By Lemma 14, one point of  $\phi$  lies on the  $r$ -branch of  $C_p$  and the other lies on the  $r'$ -branch of  $C_p$ ; we call them the  $r$ -point and  $r'$ -point of  $\phi$ , respectively. Let  $R \subseteq S$  (resp.,  $R' \subseteq S$ ) be the subset consisting of all the  $r$ -points (resp.,  $r'$ -points) of the pairs in  $\Phi^*$ .

► **Lemma 15.** *We have  $R \cap R' = \emptyset$ , and thus the graph  $G = (S, \Phi^*)$  is bipartite.*

For a pair  $\phi = (a, b) \in \Phi^*$  where  $a \in R$  and  $b \in R'$ , we define a vector  $\mathbf{v}_\phi = \overrightarrow{ab}$ . Our key lemma is the following. Let  $\text{ang}(\cdot, \cdot)$  denote the angle between two vectors.

► **Lemma 16.** *Let  $\Psi \subseteq \Phi^*$  be a subset such that  $\text{ang}(\mathbf{v}_\psi, \mathbf{v}_{\psi'}) \leq \pi/4$  for all  $\psi, \psi' \in \Psi$ . Then the graph  $G_\Psi = (S, \Psi)$  is acyclic, and in particular  $|\Psi| = O(n)$ .*



■ **Figure 5** Illustrating Lemma 16.

**Proof sketch.** Suppose there is a cycle in  $G_\Psi$ . Let  $\psi = (a, a')$  be the *shortest* edge in the cycle where  $a \in R$  and  $a' \in R'$ . Let  $\psi_1 = (b, a') \in \Psi$  and  $\psi_2 = (a, b') \in \Psi$  be the two adjacent edges of  $\psi$  in the cycle (so  $b \in R$  and  $b' \in R'$ ). Let  $C_\psi, C_{\psi_1}, C_{\psi_2}$  be the

## 61:10 Searching for the Closest-Pair in a Query Translate

associated  $C$ -translates of  $\psi, \psi_1, \psi_2$ , respectively. See Figure 5 for an example. We deduce that  $a', b, b' \in C_{p_1}$ ,  $C_{p_2} \subseteq C_p$ , and  $a, a', b, b' \in C_p$ ; see the complete proof in [16] for an argument. Since  $a, a', b, b' \in C_p$  and  $\phi$  is the closest-pair in  $C_p$  by Lemma 14, we have  $|\phi| < \text{dist}(a, b)$  and hence  $\angle aba' < \angle aa'b = \text{ang}(\mathbf{v}_\psi, \mathbf{v}_{\psi_1}) \leq \pi/4$ . It follows that

$$\text{ang}(\overrightarrow{ba}, \mathbf{v}_\psi) \leq \text{ang}(\overrightarrow{ba}, \mathbf{v}_{\psi_1}) + \text{ang}(\mathbf{v}_\psi, \mathbf{v}_{\psi_1}) = \angle aba' + \text{ang}(\mathbf{v}_\psi, \mathbf{v}_{\psi_1}) < \pi/2.$$

From this we can further deduce that

$$\text{ang}(\overrightarrow{bb'}, \mathbf{v}_\psi) = \text{ang}(\overrightarrow{ba} + \mathbf{v}_{\psi_2}, \mathbf{v}_\psi) \leq \max\{\text{ang}(\overrightarrow{ba}, \mathbf{v}_\psi), \text{ang}(\mathbf{v}_{\psi_2}, \mathbf{v}_\psi)\} < \pi/2.$$

Next, we establish an inequality that contradicts the above inequality. Let  $l$  be the bisector of  $[b, b']$ . Since  $a', b, b' \in C_{p_1}$  and  $\psi_1$  is the closest-pair in  $C_{p_1}$ , we have  $\text{dist}(a', b') > \text{dist}(b, a')$ , i.e.,  $a'$  is on the same side of  $l$  as  $b$ . Using the same argument symmetrically, we can deduce that  $a$  is on the same side of  $l$  as  $b'$ . As  $l$  is the bisector of  $[b, b']$ , this implies  $\text{ang}(\overrightarrow{bb'}, \mathbf{v}_\psi) = \text{ang}(\overrightarrow{bb'}, \overrightarrow{aa'}) > \pi/2$ , which is a contradiction. Thus,  $G_\Psi$  is acyclic.  $\blacktriangleleft$

With the above lemma in hand, it is quite straightforward to prove  $|\Phi^*| = O(n)$ . We evenly separate the plane into 8 sectors  $K_1, \dots, K_8$  around the origin. Define  $\Psi_i = \{\phi \in \Phi^* : \mathbf{v}_\phi \in K_i\}$  for  $i \in \{1, \dots, 8\}$ . Now each  $\Psi_i$  satisfies the condition in Lemma 16 and thus  $|\Psi_i| = O(n)$ . Since  $\Phi^* = \bigcup_{i=1}^8 \Psi_i$ , we have  $|\Phi^*| = O(n)$ . Therefore, we conclude the following.

► **Lemma 17.**  $|\Phi(S, \mathcal{L}_C)| = O(n)$ , where  $n = |S|$ .

Suppose  $\Phi(S, \mathcal{L}_C) = \{\phi_1, \dots, \phi_m\}$  where  $m = O(n)$  and  $\phi_1, \dots, \phi_m$  are sorted in increasing order of their lengths. Now we only need a data structure which can report, for a query  $C_q \in \mathcal{L}_C$ , the smallest  $i$  such that  $\phi_i$  is contained in  $C_q$ . Similar to the wedge case, we obtain such a data structure with  $O(m)$  space and  $O(\log m)$  query time (see [16]).

► **Theorem 18.** *There is an  $O(n)$ -space  $\mathcal{L}_C$ -RCP data structure with  $O(\log n)$  query time.*

Theorem 3 now follows from Theorem 6, 13, and 18.

### 3 Translation RCP queries for smooth convex bodies

Let  $\Gamma$  be a fixed convex body whose boundary is *smooth* (or *smooth convex body*), that is, through each point on the boundary there is a unique tangent line to  $\Gamma$ . Assume we can compute in  $O(1)$  time, for any line  $l$  in  $\mathbb{R}^2$ , the segment  $\Gamma \cap l$ . We investigate the  $\mathcal{L}_\Gamma$ -RCP problem (under the Euclidean metric). Throughout this section,  $O(\cdot)$  hides constants depending on  $\Gamma$ . Our main result is the following, to prove which is the goal of this section.

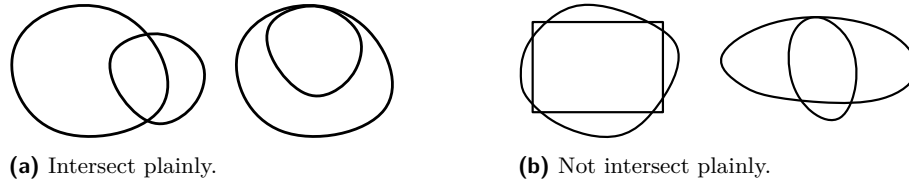
► **Theorem 19.** *Let  $\Gamma$  be a fixed smooth convex body in  $\mathbb{R}^2$ . Then there is an  $O(n \log n)$ -space  $\mathcal{L}_\Gamma$ -RCP data structure with  $O(\log^2 n)$  query time.*

Let  $S$  be the given dataset in  $\mathbb{R}^2$  of size  $n$ . Suppose for convenience that the pairwise distances of the points in  $S$  are distinct (so that the closest-pair in any subset of  $S$  is unique). Also, suppose that no three points in  $S$  are collinear. Our data structure is based on the two technical results presented below, both of which are of geometric interest. The first result states that sufficiently short candidate pairs do not cross when  $\Gamma$  is a smooth convex body.

► **Theorem 20.** *Let  $\Gamma$  be a smooth convex body. Then there exists a constant  $\tau > 0$  (depending on  $\Gamma$  only) such that if  $(a, b), (c, d) \in \Phi(S, \mathcal{L}_\Gamma)$  are two pairs whose lengths are both at most  $\tau$ , then the segments  $[a, b]$  and  $[c, d]$  do not cross.*



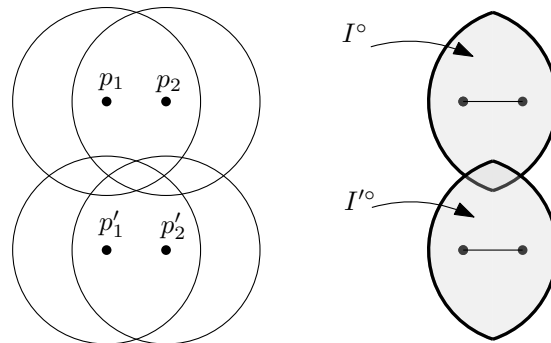
To introduce the second result, we need an important notion. For two convex bodies  $C, D$  in  $\mathbb{R}^2$  such that  $C \cap D \neq \emptyset$ , we say  $C$  and  $D$  *intersect plainly* if  $\partial C \cap D$  and  $\partial D \cap C$  are both connected; see Figure 6 for an illustration. (The reader can intuitively understand this as “the boundaries of  $C$  and  $D$  cross each other at most twice”, but it is in fact stronger.) Note that a collection of convex bodies in  $\mathbb{R}^2$  in which any two are disjoint or intersect plainly form a family of pseudo-discs [2]. Our second result is the following theorem.



■ **Figure 6** An illustration the concept of “intersect plainly”.

► **Theorem 21.** *Let  $C$  be a convex body in  $\mathbb{R}^2$ , and  $p_1, p_2, p'_1, p'_2 \in \mathbb{R}^2$  be four points (not necessarily distinct) such that  $I^\circ \neq \emptyset$  and  $I'^\circ \neq \emptyset$ , where  $I = C_{p_1} \cap C_{p_2}$  and  $I' = C_{p'_1} \cap C_{p'_2}$ . Suppose that any three of  $p_1, p_2, p'_1, p'_2$  are not collinear unless two of them coincide. If the segments  $[p_1, p_2], [p'_1, p'_2]$  do not cross and  $I \cap I' \neq \emptyset$ , then  $I$  and  $I'$  intersect plainly.*

Figure 7 gives an illustration of Theorem 21 in the case where  $C$  is a disc. Note that, even for the disc-case, without the condition that  $[p_1, p_2], [p'_1, p'_2]$  do not cross, one can easily construct a counterexample in which  $I$  and  $I'$  do not intersect plainly.



■ **Figure 7** Illustration of Theorem 21 when  $C$  is a disc.

The proof of Theorem 20 is presented later in Section 3.3. The proof of Theorem 21 is more involved and is presented the full version [16]. Next, we first assume the correctness of the two theorems and present our  $\mathcal{L}_\Gamma$ -RCP data structure. Our data structure consists of two parts  $\mathcal{D}_1$  and  $\mathcal{D}_2$  where  $\mathcal{D}_1$  handles the queries for which the length of the answer (closest-pair) is “short” and  $\mathcal{D}_2$  handles the queries for which the answer is “long”.

### 3.1 Handling short-answer queries

We describe the first part  $\mathcal{D}_1$  of our data structure. Let  $\tau > 0$  be the constant in Theorem 20 such that any two candidate pairs of lengths at most  $\tau$  do not cross. For a query  $\Gamma_q \in \mathcal{L}_\Gamma$ ,  $\mathcal{D}_1$  reports the closest-pair in  $S \cap \Gamma_q$  if its length is at most  $\tau$ , and reports nothing otherwise.

Let  $\Phi_{\leq \tau} \subseteq \Phi(S, \mathcal{L}_\Gamma)$  be the sub-collection consisting of the candidate pairs of lengths at most  $\tau$ , and suppose  $m = |\Phi_{\leq \tau}|$ . We have  $m = O(n)$ , because the graph  $G = (S, \Phi_{\leq \tau})$  is planar by Theorem 20. Define  $\tilde{\Gamma} = \{(x, y) \in \mathbb{R}^2 : (-x, -y) \in \Gamma\}$ . For a pair  $\theta = (a, b) \in \Phi_{\leq \tau}$ , we write  $I^\theta = \tilde{\Gamma}_a \cap \tilde{\Gamma}_b$ . Then  $\theta$  is contained in a query range  $\Gamma_q \in \mathcal{L}_\Gamma$  iff  $q \in I^\theta$ .

In order to design  $\mathcal{D}_1$ , we first introduce a so-called *membership* data structure (MDS). Let  $\Psi = \{\theta_1, \dots, \theta_r\} \subseteq \Phi_{\leq \tau}$  and  $U = \bigcup_{\theta \in \Psi} I^\theta$ . An MDS on  $\Psi$  can decide, for a given  $\Gamma_q \in \mathcal{L}_\Gamma$ , whether  $\Gamma_q$  contains a pair in  $\Psi$  or not. As argued before,  $\Gamma_q$  contains a pair in  $\Psi$  iff  $q \in U$ . Thus, to have an MDS on  $\Psi$ , it suffices to have a data structure that can decide if a given point is in  $U$ . By Theorem 20, the segments corresponding to any two pairs  $\theta_i, \theta_j$  do not cross each other. Also, no three points in  $S$  are collinear by assumption. Thus, by Theorem 21,  $I^{\theta_i}$  and  $I^{\theta_j}$  intersect plainly for any  $i, j \in \{1, \dots, r\}$  such that  $I^{\theta_i} \cap I^{\theta_j} \neq \emptyset$ . It follows that  $\{I^{\theta_1}, \dots, I^{\theta_r}\}$  is a family of pseudo-discs [2], and hence the complexity of their union  $U$  is  $O(r)$  by [8]. As such, optimal point location data structures (e.g., [5, 9]) can be applied to decide whether a point is contained in  $U$  in  $O(\log r)$  time, using  $O(r)$  space. We remark that, although the edges defining the boundary of  $U$  are not line-segments (existing point-location results we know of work on polygonal subdivisions), each edge is a connected portion of  $\partial \tilde{I}$  and hence can be decomposed into constant number of “fragments” that are both  $x$ -monotone and  $y$ -monotone. Recall our assumption that we can compute (in constant time)  $\Gamma \cap l$  for any line  $l$  in  $\mathbb{R}^2$ , and thus also  $\tilde{I} \cap l$  for any line  $l$ . With this assumption, the existing data structures [5, 9] can be generalized straightforwardly for our purpose. Thus, we have an MDS on  $\Psi$  with  $O(r)$  space and  $O(\log r)$  query time, which we denote by  $\mathcal{M}(\Psi)$ .

With the MDS in hand, we can now design  $\mathcal{D}_1$ . For a sub-collection  $\Psi = \{\theta_1, \dots, \theta_r\} \subseteq \Phi_{\leq \tau}$  where  $\theta_1, \dots, \theta_r$  are sorted in increasing order of their lengths, let  $\mathcal{D}_1(\Psi)$  be a data structure defined as follows. If  $r = 1$ , then  $\mathcal{D}_1(\Psi)$  simply stores the only pair  $\theta_1 \in \Psi$ . If  $r > 1$ , let  $\Psi_1 = \{\theta_1, \dots, \theta_{\lfloor r/2 \rfloor}\}$  and  $\Psi_2 = \{\theta_{\lfloor r/2 \rfloor + 1}, \dots, \theta_r\}$ . Then  $\mathcal{D}_1(\Psi)$  consists of three parts:  $\mathcal{D}_1(\Psi_1)$ ,  $\mathcal{D}_1(\Psi_2)$ , and  $\mathcal{M}_{\Psi_1}$ , where  $\mathcal{D}_1(\Psi_1)$  and  $\mathcal{D}_1(\Psi_2)$  are defined recursively. We show that we can use  $\mathcal{D}_1(\Psi)$  to find, for a query  $\Gamma_q \in \mathcal{L}_\Gamma$ , the shortest pair  $\theta^* \in \Psi$  contained in  $\Gamma_q$ . We first query  $\mathcal{M}_{\Psi_1}$  to see if  $\Gamma_q$  contains a pair in  $\Psi_1$ . If so,  $\theta^*$  must be in  $\Psi_1$ , so we recursively query  $\mathcal{D}_1(\Psi_1)$  to find it. Otherwise, we recursively query  $\mathcal{D}_1(\Psi_2)$ . In this way, we can eventually find  $\theta^*$ . Now we simply define  $\mathcal{D}_1 = \mathcal{D}_1(\Phi_{\leq \tau})$ . A direct analysis shows that the space of  $\mathcal{D}_1$  is  $O(m \log m)$  and the query time of  $\mathcal{D}_1$  is  $O(\log^2 m)$  (see [16]).

### 3.2 Handling long-answer queries

If  $\mathcal{D}_1$  fails to answer the query  $\Gamma_q$ , then the length of the closest-pair in  $S \cap \Gamma_q$  is greater than  $\tau$ . In this case, we shall use the second part  $\mathcal{D}_2$  of our data structure to answer the query.  $\mathcal{D}_2$  simply reports all the points in  $S \cap \Gamma_q$  and computes the closest-pair by brute-force. Since the length of the closest-pair in  $S \cap \Gamma_q$  is greater than  $\tau$ , we have  $|S \cap \Gamma_q| = O(1)$  by Lemma 1 and hence computing the closest-pair takes  $O(1)$  time. In order to do reporting, we consider the problem in the dual setting. Again, define  $\tilde{I} = \{(x, y) \in \mathbb{R}^2 : (-x, -y) \in \Gamma\}$ . Clearly, for any  $a \in \mathbb{R}^2$ ,  $a \in \Gamma_q$  iff  $q \in \tilde{I}_a$ . Thus, the problem is equivalent to reporting the ranges in  $\mathcal{S} = \{\tilde{I}_a : a \in S\}$  that contain  $q$ . Define the *depth*,  $\text{dep}(p)$ , of a point  $p \in \mathbb{R}^2$  as the number of the ranges in  $\mathcal{S}$  containing  $p$ . Let  $\mathcal{A}$  be the arrangement of the ranges in  $\mathcal{S}$ , and  $k$  be a sufficiently large constant. The  $\leq k$ -level of  $\mathcal{A}$ , denoted by  $\mathcal{A}_{\leq k}$ , is the sub-arrangement of  $\mathcal{A}$  contained in the region  $R_{\leq k} = \{p \in \mathbb{R}^2 : \text{dep}(p) \leq k\}$ . By Theorem 21, any two ranges  $\tilde{I}_a, \tilde{I}_b \in \mathcal{S}$  intersect plainly if they intersect (setting  $p_1 = p_2 = a$  and  $p'_1 = p'_2 = b$  when applying Theorem 21), which implies that  $\mathcal{S}$  is a family of  $n$  pseudo-discs and  $\mathcal{A}$  is a pseudo-disc arrangement. So we have the following well-known lemma.

► **Lemma 22.** *The complexity of  $\mathcal{A}_{\leq k}$  is  $O(n)$  for a constant  $k$ .*

By the above lemma, we can build a point-location data structure on  $\mathcal{A}_{\leq k}$  with  $O(n)$  space and  $O(\log n)$  query time. Also, we associate to each cell  $\Delta$  of  $\mathcal{A}_{\leq k}$  the (at most  $k$ ) ranges in  $\mathcal{S}$  containing  $\Delta$ . Now we can report the ranges in  $\mathcal{S}$  containing  $q$  as follows. Since

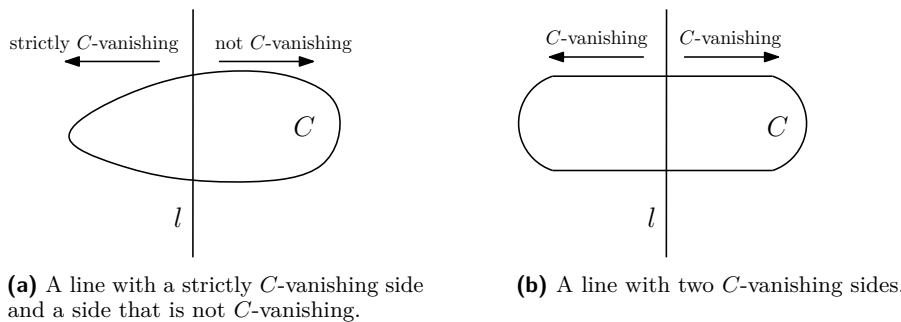
$|S \cap \Gamma_q| = O(1)$  and  $k$  is sufficiently large, we have  $|S \cap \Gamma_q| \leq k$  and hence  $q$  is in  $\mathcal{A}_{\leq k}$ . Using the point-location data structure, we find in  $O(\log n)$  time the cell  $\Delta$  of  $\mathcal{A}_{\leq k}$  containing  $q$ . Then the ranges associated to  $\Delta$  are exactly those containing  $q$ . Together with our argument above, this gives us the desired data structure  $\mathcal{D}_2$  with  $O(n)$  space and  $O(\log n)$  time. Combining  $\mathcal{D}_2$  with the data structure  $\mathcal{D}_1$  in the last section, Theorem 19 is proved.

### 3.3 Proof of Theorem 20

We begin with introducing some basic notions and geometric results regarding convex bodies in  $\mathbb{R}^2$ . Let  $C$  be a convex body in  $\mathbb{R}^2$ . For a line  $l$  in  $\mathbb{R}^2$ , we denote by  $\text{len}_C(l)$  the length of the segment  $C \cap l$ . Suppose  $l$  is  $ax + by + c = 0$ , then it cuts  $\mathbb{R}^2$  into two halfplanes,  $ax + by + c \geq 0$  and  $ax + by + c \leq 0$  (called the two *sides* of  $l$  hereafter). Let  $H$  be one side of  $l$ . For a number  $t \geq 0$ , let  $l_t$  denote the (unique) line parallel to  $l$  satisfying  $l_t \subseteq H$  and  $\text{dist}(l, l_t) = t$ . Set  $\lambda = \sup\{t \geq 0 : C \cap l_t \neq \emptyset\}$  (if  $\{t \geq 0 : C \cap l_t \neq \emptyset\} = \emptyset$ , set  $\lambda = 0$ ).

► **Definition 23.** We say  $H$  is a *C-vanishing* (resp., *strictly C-vanishing*) side of  $l$ , if  $f(t) = \text{len}_C(l_t)$  is a non-increasing (resp., decreasing) function in the domain  $[0, \lambda]$ .

Figure 8 gives an intuitive illustration of the above definition. Note that for any convex body  $C$  and line  $l$  in  $\mathbb{R}^2$ , at least one side of  $l$  is  $C$ -vanishing due to the convexity of  $C$ .



■ **Figure 8** An illustration of “ $C$ -vanishing”.

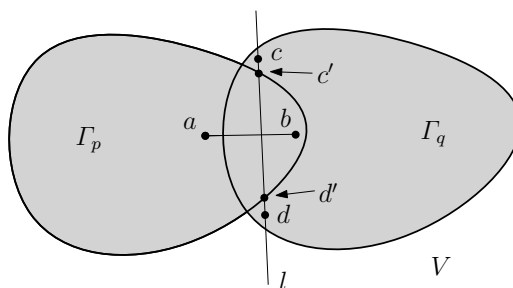
In order to prove Theorem 20, we establish the following key observations.

► **Lemma 24.** Let  $C$  be a convex body in  $\mathbb{R}^2$ ,  $p_1, p_2 \in \mathbb{R}^2$  be two points, and  $l$  be an arbitrary line in  $\mathbb{R}^2$ . Then we have the following facts.

- (1) If  $C_{p_1} \cap l \not\subseteq C_{p_2} \cap l$  and  $V$  is a  $C_{p_1}$ -vanishing side of  $l$ , then  $C_{p_1} \cap V \subseteq C_{p_2}$ .
- (2) If  $C_{p_1} \cap l$  is contained in the “interior” of  $C_{p_2} \cap l$  (i.e.,  $C_{p_2} \cap l$  excluding both endpoints) and  $V$  is a  $C_{p_1}$ -vanishing side of  $l$ , then  $C_{p_1} \cap (V \setminus l) \subseteq C_{p_2} \setminus \partial C_{p_2}$ .

► **Lemma 25.** Let  $C$  be a smooth convex body in  $\mathbb{R}^2$ . Then there exists a number  $\tau > 0$  such that, for any line  $l$  with  $0 < \text{len}_C(l) < \tau$  and any point  $r \in C$  on a  $C$ -vanishing side of  $l$ , the distance between  $r$  and an (arbitrary) endpoint of  $C \cap l$  is less than  $\text{len}_C(l)$ .

Now we are able to prove Theorem 20. Suppose  $\Gamma$  is a smooth convex body in  $\mathbb{R}^2$ . Taking  $C = \Gamma$ , we can find a constant  $\tau$  satisfying the condition in Lemma 25. We claim that  $\tau$  also satisfies the condition in Theorem 20. Let  $(a, b), (c, d) \in \Phi(S, \mathcal{L}_\Gamma)$  be two pairs of lengths at most  $\tau$ . Assume that  $[a, b]$  and  $[c, d]$  cross. By Lemma 2, there exists  $\Gamma_p \in \mathcal{L}_\Gamma$  such that either  $\Gamma_p \cap \{a, b, c, d\} = \{a, b\}$  or  $\Gamma_p \cap \{a, b, c, d\} = \{c, d\}$ ; assume  $\Gamma_p \cap \{a, b, c, d\} = \{a, b\}$ . Suppose  $(c, d)$  is the closest-pair in  $\Gamma_q \in \mathcal{L}_\Gamma$ . Let  $l$  be the line through  $c, d$ , and  $V$  be a



■ **Figure 9** Illustrating the proof of Theorem 20.

$\Gamma_p$ -vanishing side of  $l$ . See Figure 9 for an illustration of the notations. Since the segments  $[a, b]$  and  $[c, d]$  cross,  $[c, d]$  must intersect  $\Gamma_p$ . But  $c, d \notin \Gamma_p$ , thus  $\Gamma_p \cap l \subsetneq [c, d]$ . Furthermore, because  $[c, d] \subseteq \Gamma_q \cap l$ , we have  $\Gamma_p \cap l \subsetneq \Gamma_q \cap l$ . This implies  $\Gamma_p \cap V \subseteq \Gamma_q$  by (1) of Lemma 24. Note that one of  $a, b$  must be contained in  $\Gamma_p \cap V$ , say  $b \in \Gamma_p \cap V$ . Thus  $b \in \Gamma_q$ . We now show that  $\text{dist}(b, c) < \text{dist}(c, d)$  and  $\text{dist}(b, d) < \text{dist}(c, d)$ . As  $\Gamma_p \cap l \subsetneq [c, d]$  and the length of  $[c, d]$  is at most  $\tau$ , we have  $\text{len}_{\Gamma_p}(l) < \tau$ . We denote by  $c', d'$  the two endpoints of the segment  $\Gamma_p \cap l$ ; see Figure 9. By Lemma 25, the distance from  $c'$  (or  $d'$ ) to any point in  $\Gamma_p \cap V$  is less than  $\text{len}_{\Gamma_p}(l) = \text{dist}(c', d')$ . In particular,  $\text{dist}(b, c') < \text{dist}(c', d')$  and  $\text{dist}(b, d') < \text{dist}(c', d')$ . Now  $\text{dist}(b, c) \leq \text{dist}(b, c') + \text{dist}(c', c) < \text{dist}(c', d') + \text{dist}(c', c) = \text{dist}(c, d') < \text{dist}(c, d)$ . For the same reason, we have  $\text{dist}(b, d) < \text{dist}(c, d)$ . Since  $b, c, d \in \Gamma_q$ , this contradicts the fact that  $(c, d)$  is the closest-pair in  $\Gamma_q$ . The proof of Theorem 20 is now complete.

## References

- 1 M. A. Abam, P. Carmi, M. Farshi, and M. Smid. On the power of the semi-separated pair decomposition. In *Workshop on Algorithms and Data Structures*, pages 1–12. Springer, 2009.
- 2 P. K. Agarwal, J. Pach, and M. Sharir. State of the union (of geometric objects): A review. *Computational Geometry: Twenty Years Later. American Mathematical Society*, 2007.
- 3 Pankaj K Agarwal, Jeff Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.
- 4 Sang Won Bae and Michiel Smid. Closest-Pair Queries in Fat Rectangles. *arXiv preprint*, 2018. [arXiv:1809.10531](https://arxiv.org/abs/1809.10531).
- 5 Herbert Edelsbrunner, Leonidas J Guibas, and Jorge Stolfi. Optimal point location in a monotone subdivision. *SIAM Journal on Computing*, 15(2):317–340, 1986.
- 6 Prosenjit Gupta. Range-aggregate query problems involving geometric aggregation operations. *Nordic Journal of Computing*, 13(4):294–308, 2006.
- 7 Prosenjit Gupta, Ravi Janardan, Yokesh Kumar, and Michiel Smid. Data structures for range-aggregate extent queries. *Computational Geometry: Theory and Applications*, 2(47):329–347, 2014.
- 8 K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1(1):59–71, 1986.
- 9 N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Communications of the ACM*, 29(7):669–679, 1986.
- 10 Jing Shan, Donghui Zhang, and Betty Salzberg. On spatial-range closest-pair query. In *International Symposium on Spatial and Temporal Databases*, pages 252–269. Springer, 2003.
- 11 R. Sharathkumar and P. Gupta. Range-aggregate proximity queries. *Technical Report IIT/TR/2007/80. IIT Hyderabad, Telangana*, 500032, 2007.
- 12 Michiel Smid. Closest-point problems in computational geometry. In *Handbook of computational geometry*, pages 877–935. Elsevier, 2000.

- 13 Jie Xue. Colored range closest-pair problem under general distance functions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 373–390. SIAM, 2019.
- 14 Jie Xue, Yuan Li, and Ravi Janardan. Approximate range closest-pair search. In *Proceedings of the 30th Canadian Conference on Computational Geometry*, pages 282–287, 2018.
- 15 Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. New Bounds for Range Closest-Pair Problems. In *Proceedings of the 34th International Symposium on Computational Geometry*, pages 73:1–73:14, 2018.
- 16 Jie Xue, Yuan Li, Saladi Rahul, and Ravi Janardan. Searching for the closest-pair in a query translate. *arXiv preprint*, 2018. [arXiv:1807.09498](https://arxiv.org/abs/1807.09498).



# On the Complexity of the $k$ -Level in Arrangements of Pseudoplanes

Micha Sharir

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

<http://www.cs.tau.ac.il/~michas/>

[michas@post.tau.ac.il](mailto:michas@post.tau.ac.il)

Chen Ziv

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

[henziv@gmail.com](mailto:henziv@gmail.com)

---

## Abstract

A classical open problem in combinatorial geometry is to obtain tight asymptotic bounds on the maximum number of  $k$ -level vertices in an arrangement of  $n$  hyperplanes in  $\mathbb{R}^d$  (vertices with exactly  $k$  of the hyperplanes passing below them). This is essentially a dual version of the  $k$ -set problem, which, in a primal setting, seeks bounds for the maximum number of  $k$ -sets determined by  $n$  points in  $\mathbb{R}^d$ , where a  $k$ -set is a subset of size  $k$  that can be separated from its complement by a hyperplane. The  $k$ -set problem is still wide open even in the plane. In three dimensions, the best known upper and lower bounds are, respectively,  $O(nk^{3/2})$  [15] and  $nk \cdot 2^{\Omega(\sqrt{\log k})}$  [19].

In its dual version, the problem can be generalized by replacing hyperplanes by other families of surfaces (or curves in the planes). Reasonably sharp bounds have been obtained for curves in the plane [16, 18], but the known upper bounds are rather weak for more general surfaces, already in three dimensions, except for the case of triangles [1]. The best known general bound, due to Chan [7] is  $O(n^{2.997})$ , for families of surfaces that satisfy certain (fairly weak) properties.

In this paper we consider the case of *pseudoplanes* in  $\mathbb{R}^3$  (defined in detail in the introduction), and establish the upper bound  $O(nk^{5/3})$  for the number of  $k$ -level vertices in an arrangement of  $n$  pseudoplanes. The bound is obtained by establishing suitable (and nontrivial) extensions of dual versions of classical tools that have been used in studying the primal  $k$ -set problem, such as the Lovász Lemma and the Crossing Lemma.

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

**Keywords and phrases**  $k$ -level, pseudoplanes, arrangements, three dimensions,  $k$ -sets

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.62

**Related Version** A full version of this paper is available at <https://arxiv.org/abs/1903.07196>.

**Funding** Work on this paper was supported by Grants 892/13 and 260/18 from the Israel Science Foundation.

*Micha Sharir:* Work on this paper was also supported by Grant G-1367-407.6/2016 from the German-Israeli Foundation for Scientific Research and Development, by the Blavatnik Research Fund in Computer Science at Tel Aviv University, and by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

## 1 Introduction

Let  $\Lambda$  be a set of  $n$  non-vertical planes (resp., pseudoplanes, as will be formally defined shortly) in  $\mathbb{R}^3$ , in general position. We say that a point  $p$  lies at *level*  $k$  of the arrangement  $\mathcal{A}(\Lambda)$ , and write  $\lambda(p) = k$ , if exactly  $k$  planes (resp., pseudoplanes) of  $\Lambda$  pass below  $p$ . The  $k$ -level of  $\mathcal{A}(\Lambda)$  is the closure of the set of points that lie on the planes of  $\Lambda$  and are at level  $k$ . Our goal is to obtain an upper bound on the complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$ , which is



© Micha Sharir and Chen Ziv;

licensed under Creative Commons License CC-BY

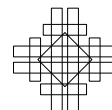
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 62; pp. 62:1–62:15

Leibniz International Proceedings in Informatics



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





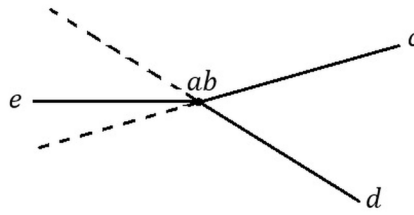
measured by the number of vertices of  $\mathcal{A}(\Lambda)$  that lie at level  $k$ . (The level may also contain vertices at level  $k - 1$  or  $k - 2$ , but we ignore this issue – it does not effect the worst-case asymptotic bound that we are after.) Using a standard duality transform that preserves the above/below relationship (see, e.g., [10]), the case of planes is the dual version of the following variant of the  $k$ -set problem: given a set of  $n$  points in  $\mathbb{R}^3$  in general position, how many triangles spanned by  $P$  are such that the plane supporting the triangle has exactly  $k$  points of  $P$  below it? We refer to these triangles as  $k$ -triangles. This has been studied by Dey and Edelsbrunner [9], in 1994, for the case of *halving triangles*, namely  $k$ -triangles with  $k = (n - 3)/2$  (and  $n$  odd). They have shown that the number of halving triangles is  $O(n^{8/3})$ . In 1998, Agarwal et al. [1] generalized this result for  $k$ -triangles, for arbitrary  $k$ , showing that their number is  $O(nk^{5/3})$ , using a probabilistic argument. In 1999, Sharir, Smorodinsky and Tardos [15] improved the upper bound for the number of  $k$ -triangles in  $S$  to  $O(nk^{3/2})$ . Chan [6] has adapted a dualized view of the technique of Sharir et al. [15] in order to study the bichromatic  $k$ -set problem: given two sets  $R$  and  $B$  of points in  $\mathbb{R}^2$  of total size  $n$  and an integer  $k$ , how many subsets of the form  $(R \cap h) \cup (B \setminus h)$  can have size exactly  $k$  over all halfplanes  $h$ ? This problem arises when we estimate the number of vertices at level  $k$ , in an arrangement of  $n$  planes in 3-space, that lie on one specific plane.

The three-dimensional case extends the more extensively studied planar case. In its primal setting, we have a set  $S$  of  $n$  points in the plane in general position, and a parameter  $k < n$ , and we seek bounds on the maximum number of  $k$ -edges, which are segments spanned by pairs of points of  $S$  so that one of the halfplanes bounded by the line supporting the segment, say the lower halfplane, contains exactly  $k$  points of  $S$ . In the dual version, we seek bounds on the maximum number of vertices of an arrangement of  $n$  nonvertical lines in general position that lie at level  $k$ . The best known upper bound for this quantity, due to Dey [8], is  $O(nk^{1/3})$ , and the best known lower bound, due to Tóth [19] is  $ne^{\Omega(\sqrt{\ln k})}$  (Nivasch [13] has slightly improved this bound for the case of halving edges).

In this paper we consider the dual version of the problem in three dimensions, where the points are mapped to planes, and the  $k$ -triangles are mapped to vertices of the arrangement of these planes at level  $k$ . We translate parts of the machinery developed in [15] to the dual setting, and then extend it to handle the case of pseudoplanes. In the primal setting, we have a set  $S$  of  $n$  points in  $\mathbb{R}^3$  in general position, and the set  $T$  of  $k$ -triangles spanned by  $S$ . We say that triangle  $\Delta_1$  *crosses* another triangle  $\Delta_2$  if the triangles share exactly one vertex, and the edge opposite to that vertex in  $\Delta_1$  intersects the interior of  $\Delta_2$ . Denote the number of ordered pairs of crossing  $k$ -triangles by  $X^k$ . The general technique in [15] is to establish an upper bound and a lower bound on  $X^k$ , and to combine these two bounds to derive an upper bound for the number of  $k$ -triangles in  $S$ .

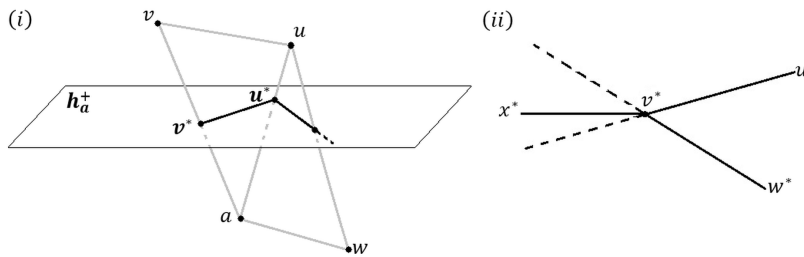
The upper bound in [15] is based on the 3-dimensional version of the Lovász Lemma, as in [5]: Any line crosses at most  $O(n^2)$  interiors of  $k$ -triangles. The lemma follows from the main property of the set  $T$ , which is its *antipodality*. Informally, the property asserts that for each pair of points  $a, b \in S$ , the  $k$ -triangles having  $ab$  as an edge form an antipodal system, in the sense that for any pair  $\Delta_{abc}, \Delta_{abd}$  of such triangles that are consecutive in the circular order around  $ab$ , the dihedral wedge that is formed by the two halfplanes that contain  $\Delta_{abc}, \Delta_{abd}$ , and are bounded by the line through  $ab$ , has the property that its *antipodal wedge*, formed by the two complementary halfplanes within the planes supporting  $\Delta_{abc}, \Delta_{abd}$ , contains a point  $e \in S$  such that  $\Delta_{abe}$  is also a  $k$ -triangle; See Figure 1.

To obtain a lower bound on  $X^k$ , the technique in [15] defines, for each  $a \in S$ , a graph  $G_a = (V_a, E_a)$  drawn in a horizontal plane  $h_a^+$  slightly above  $a$ , whose edges are, roughly, the cross-sections of the  $k$ -triangles incident to  $a$  with the plane (see Figure 2). The analysis



■ **Figure 1** The antipodality property of  $k$ -triangles: the edge  $ab$  is drawn from a side-view as a point. The triangles  $\Delta_{abc}, \Delta_{abd}$  and  $\Delta_{abe}$ , drawn as segments, are all  $k$ -triangles, where  $e$  is contained in the antipodal wedge formed by the two complementary halfplanes within the planes supporting  $\Delta_{abc}, \Delta_{abd}$ .

in [15] shows that  $G_a$  inherits the antipodality property of the  $k$ -triangles, and uses this fact to decompose  $G_a$  into a collection of convex chains, and to estimate the number of crossings between the chains. Summing these bounds over all  $a \in S$ , the lower bound on  $X^k$  follows.



■ **Figure 2** (i) The graph  $G_a$  on  $h_a^+$ .  $\Delta_{auw}, \Delta_{auv}$  are halving triangles.  $uv$  is mapped to the segment  $u^*v^*$ , and  $uw$  is mapped to a ray emanating from  $u^*$  on  $h_a^+$ . (ii) The antipodality property in  $G_a$ .

We omit further details of the way in which these lower bounds are derived in [15], because, in the dual version that we present here, we use a weaker lower bound, which is based on a dual version of the *Crossing Lemma* (see [4]), and which is easier to extend to the case of pseudoplanes. Let  $G = (V, E)$  be a simple graph, and define the *crossing number* of  $G$  as the minimum number of intersecting pairs of edges in any drawing of  $G$  in the plane. In the primal setting, the Crossing Lemma asserts that any simple graph  $G = (V, E)$  drawn in the plane, with  $|E| > 4|V|$ , has crossing number at least<sup>1</sup>  $\frac{|E|^3}{64|V|^2}$ . Using this technique for deriving a lower bound on  $X^k$ , instead of the refined technique in [15], one can show that the number of  $k$ -triangles is  $O(n^{8/3})$ , or, with the additional technique of [1],  $O(nk^{5/3})$ .

We now present the dual setting for the problem, where the input is a set  $\Lambda$  of  $n$  non-vertical planes in  $\mathbb{R}^3$  in general position.

► **Definition 1.** Let  $a, b, c \in \Lambda$ . The open region between the lower envelope and the upper envelope of  $a, b, c$  is called **the corridor of  $a, b, c$**  and is denoted by  $C_{a,b,c}$ .

The planes  $a, b, c \in \Lambda$  divide the space into eight disjoint octant-like portions, and  $C_{a,b,c}$  is the union of six of those portions, excluding the upper and the lower octants. We will be mostly interested in corridors  $C_{a,b,c}$  for which the point  $p_{a,b,c} = a \cap b \cap c$  (the unique vertex of  $C_{a,b,c}$ ) is at level  $k$ . We will refer to such corridors as  $k$ -corridors, and define  $C^k$  as the collection of  $k$ -corridors in  $\mathcal{A}(\Lambda)$ ;  $k$ -corridors serve as a dual version of  $k$ -triangles.

<sup>1</sup> The constant of proportionality has been improved in subsequent works, but we will stick to this bound.

► **Definition 2.** We say that a corridor  $C_1$  is immersed in a corridor  $C_2$  in  $\mathcal{A}(\Lambda)$  if they share exactly one plane, and the intersection line of the other two planes of  $C_1$  is fully contained in  $C_2$ . Let  $X^k$  denote the number of ordered pairs of immersed corridors in  $C^k$ .

Immersion of  $k$ -corridors is the dual notion of crossings of  $k$ -triangles. Note that if a corridor  $C_1$  is immersed in a corridor  $C_2$ , it cannot be that  $C_2$  is also immersed in  $C_1$ .

In the full version of the paper [17] (see also [20]), we provide details of this dual setup. We define  $X^k$  as the number of ordered pairs of immersed  $k$ -corridors, and present the derivation of the upper bound and the lower bound on  $X^k$ . In this abstract, we only consider in detail the extension to the case of pseudoplanes, which is our main topic of interest. We remark, though, that this translation to the dual context, although routine in principle, is rather involved and nontrivial, and requires careful handling of quite a few details.

## 2 The case of pseudoplanes

We say that a family  $\Lambda$  of  $n$  surfaces in  $\mathbb{R}^3$  is a *family of pseudoplanes* in general position if

- (i) The surfaces of  $\Lambda$  are graphs of total bivariate continuous functions.
- (ii) The intersection of any pair of surfaces in  $\Lambda$  is a connected  $x$ -monotone unbounded curve.
- (iii) Any triple of surfaces in  $\Lambda$  intersect in exactly one point.
- (iv) The  $xy$ -projections of the set of all  $\binom{n}{2}$  intersection curves of the surfaces form a *family of pseudolines* in the plane. That is, this is a collection of  $\binom{n}{2}$   $x$ -monotone unbounded curves, each pair of which intersect exactly once; see [3] for more details.

The assumption that the pseudoplanes of  $\Lambda$  are in general position means that no point is incident to more than three pseudoplanes, no intersection curve of two pseudoplanes is tangent to a third pseudoplane, and no two pseudoplanes are tangent to each other. We note that conditions (i)–(iii) are natural, but condition (iv) might appear somewhat restrictive, although it obviously holds for planes. For any  $a, b, c \in \Lambda$ , we denote the intersection curve  $a \cap b$  by  $\gamma_{a,b}$ , and the intersection point  $a \cap b \cap c$  by  $p_{a,b,c}$ .

► **Definition 3.** Let  $\gamma$  be a curve in  $\mathbb{R}^3$ . The vertical curtain through  $\gamma$ , denoted by  $\Upsilon_\gamma$ , is the collection of all  $z$ -vertical lines that intersect  $\gamma$ . The portion of  $\Upsilon_\gamma$  above (resp., below)  $\gamma$  is called the upper (resp., lower) curtain of  $\gamma$ , and is denoted by  $\Upsilon_\gamma^u$  (resp.,  $\Upsilon_\gamma^d$ ).

Let  $\gamma$  be an  $x$ -monotone unbounded connected curve in  $\mathbb{R}^3$ , and let  $p \in \gamma$ . We call each of the two connected components of  $\gamma \setminus \{p\}$  a *half-curve* of  $\gamma$  emanating from  $p$ .

The following lemma is derived from the general position of the pseudoplanes in  $\Lambda$ :

- **Lemma 4.** Let  $a, b, c \in \Lambda$ , and let  $\gamma_{a,b} = a \cap b$ ,  $p_{a,b,c} = a \cap b \cap c$ .
- (a) One of the two half-curves of  $\gamma_{a,b}$  that emanates from  $p_{a,b,c}$  lies fully below  $c$ , and the other half-curve lies fully above  $c$ .
  - (b) The collection of intersections between the surfaces of  $\Lambda$  and  $\Upsilon_{\gamma_{a,b}}$  forms an arrangement of unbounded  $x$ -monotone curves on  $\Upsilon_{\gamma_{a,b}}$ , each pair of which intersect at most once.<sup>2</sup>

**Proof.** The proof of (a) is straightforward and is omitted here. For (b), property (iv) implies that, for any  $c, d \in \Lambda \setminus \{a, b\}$ , the projection on the  $xy$ -plane of  $\gamma_{a,b}$  and  $\gamma_{c,d} = c \cap d$  intersect at most once. Thus, the intersection curves  $c \cap \Upsilon_{\gamma_{a,b}}$ ,  $d \cap \Upsilon_{\gamma_{a,b}}$  intersect at most once. ◀

<sup>2</sup> In a sense, this is a collection of pseudolines, except that they are, in general, not drawn in a plane.

Another property of  $\mathcal{A}(\Lambda)$ , shown in Agarwal and Sharir [2], is:

► **Lemma 5.** *The complexity of the lower envelope of  $\Lambda$  is  $O(n)$ .*

The notion of corridors can easily be extended to the case of pseudoplanes. That is, for any  $a, b, c \in \Lambda$ , denote by  $C_{a,b,c}$  the open region between the lower envelope and the upper envelope of  $a, b, c$ , and call it the *corridor of  $a, b, c$* . Refer to corridors  $C_{a,b,c}$  for which the intersection point  $p_{a,b,c}$  lies at level  $k$  as  *$k$ -corridors*, and define  $C^k$  as the collection of  $k$ -corridors in  $\mathcal{A}(\Lambda)$ . The following is an extension of Definition 2:

► **Definition 6.** *A corridor  $C_1$  is immersed in a corridor  $C_2$  if they share exactly one pseudoplane, and the intersection curve of the other two pseudoplanes of  $C_1$  is fully contained in  $C_2$ . Let  $X^k$  denote the number of ordered pairs of immersed corridors in  $C^k$ .*

**Organization of this section.** In Section 2.1 we derive an upper bound for  $X^k$ , using an extended dual version of the Lovász Lemma. In Section 2.2 we obtain a lower bound for  $X^k$ , using a dual version of the Crossing Lemma. In Section 2.3 we combine those two bounds to obtain an upper bound on the complexity of the  $k$ -level of the arrangement.

### 2.1 An extension of the dual version of the Lovász Lemma

The following lemma is an extension to the case of pseudoplanes of a dual version of the antipodality property in the primal setup.

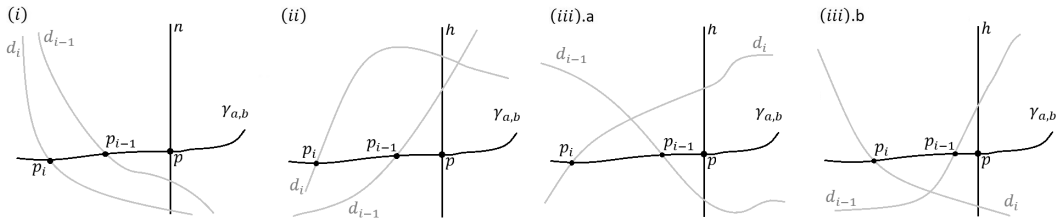
► **Lemma 7.** *Let  $\Lambda$  be as above, let  $a, b \in \Lambda$ , and let  $\gamma_{a,b} = a \cap b$  denote their intersection curve. Let  $h$  be a  $z$ -vertical line (i.e., parallel to the  $z$ -axis) that intersects  $\gamma_{a,b}$  at some point  $p$ . Let  $D = \Lambda \setminus \{a, b\}$ , and  $D^k = \{d \in D \mid C_{a,b,d} \in C^k\}$ . Denote  $h_{up} = \{d \in D \mid z_{d \cap h} > z_p\}$  and  $h_{down} = \{d \in D \mid z_{d \cap h} < z_p\}$  (by choosing  $p$  generically, we may assume that all these inequalities are indeed sharp). We then have  $||h_{up} \cap D^k| - |h_{down} \cap D^k|| \leq 2$ .*

**Proof.** Denote the two half-curves of  $\gamma_{a,b}$  emanating from  $p$  by  $\eta_1$  and  $\eta_2$ , and denote  $D_{\eta_1} = \{d \in D \mid d \text{ intersects } \eta_1\}$ ,  $D_{\eta_2} = \{d \in D \mid d \text{ intersects } \eta_2\}$ . Clearly, with a generic choice of  $p$ ,  $D_{\eta_1} \cup D_{\eta_2} = D$ , and since each triple of pseudoplanes in  $\Lambda$  intersects only once,  $D_{\eta_1} \cap D_{\eta_2} = \emptyset$ . Enumerate the pseudoplanes in  $D_{\eta_1}$  as  $d_1, \dots, d_j$ , according to the order in which their respective intersection points with  $\eta_1$ , denoted  $p_1, \dots, p_j$ , appear on  $\eta_1$  in the direction from  $p$  to the end of the half-curve. Assume there are  $1 \leq r < s \leq j$  such that  $d_r, d_s \in D_{\eta_1} \cap (h_{down} \cap D^k)$ , and denote  $D_{r,s} = \{d_i \in D_{\eta_1} \mid r \leq i \leq s\}$ .

It is easy to show that, for each point  $p_i$ , the following properties hold (as they do in the case of planes. See the full version [17]); see Figure 3.

- (i)  $\lambda(p_i) = \lambda(p_{i-1}) - 1$  if and only if both  $d_{i-1}, d_i \in h_{down}$ .
- (ii)  $\lambda(p_i) = \lambda(p_{i-1}) + 1$  if and only if both  $d_{i-1}, d_i \in h_{up}$ .
- (iii)  $\lambda(p_i) = \lambda(p_{i-1})$  if and only if one of  $d_{i-1}, d_i$  is in  $h_{up}$  and the other one is in  $h_{down}$ .

We claim that there must exist a pseudoplane  $d' \in D_{r,s}$  that is in  $h_{up} \cap D^k$ . If  $d_{r+1} \in h_{up}$  then, by property (iii),  $\lambda(d_{r+1}) = \lambda(d_r) = k$  and we are done. Otherwise,  $d_{r+1} \in h_{down}$ , and by (i) above, the level of  $p_{r+1}$  is  $k - 1$ . Note that  $r + 1 < s$  because the level of  $p_s$  is  $k$ . Because the level can change only by 0, +1, or -1 between two consecutive points  $p_i, p_{i+1}$ , there must be a point  $p_i \in D_{r,s} \subseteq D_{\eta_1}$ , so that the level of  $p_i$  is  $k$  and the level of the previous point on  $l_{a,b}$  is  $k - 1$ , which means, by (ii) above, that  $d_i \in h_{up}$ . That is, between each pair  $p_r, p_s \in \eta_1$  so that  $d_r, d_s \in h_{down} \cap D^k$ , there exists  $p_i$  so that  $d_i \in h_{up} \cap D^k$ , and our claim is established.



**Figure 3** Both  $d_i, d_{i-1}$  intersect the same half-curve of  $\gamma_{a,b}$  emanating from  $p$ : (i) The case where both  $d_i, d_{i-1} \in h_{down}$ . (ii) The case where both  $d_i, d_{i-1} \in h_{up}$ . (iii.a-b) The case where one of  $d_i, d_{i-1}$  is in  $h_{down}$ , and the other one is in  $h_{up}$ .

Similarly, between each pair  $p_r, p_s \in \eta_1$  so that  $d_r, d_s \in h_{up} \cap D^k$ , there exists  $p_i$ , for some  $r < i < s$ , so that  $d_i \in h_{down} \cap D^k$ . Both of these properties are easily seen to imply that

$$\left| |h_{up} \cap D^k \cap D_{\eta_1}| - |h_{down} \cap D^k \cap D_{\eta_1}| \right| \leq 1.$$

The same reasoning applies to  $\eta_2$ , and yields  $\left| |h_{up} \cap D^k \cap D_{\eta_2}| - |h_{down} \cap D^k \cap D_{\eta_2}| \right| \leq 1$ .

Thus,  $\left| |h_{up} \cap D^k| - |h_{down} \cap D^k| \right| \leq 2$ . ◀

We next apply this lemma to obtain an extended dual version of the Lovász Lemma. Concretely, we derive an upper bound on the number of  $k$ -corridors that fully contain an intersection curve  $\gamma_{a,b} = a \cap b$ . In the case of planes (see [17]), the curve in question is the intersection line of two dual planes, which is dual to the line connecting the two corresponding primal points. The  $k$ -corridors are dual to the  $k$ -triangles in the primal, and a  $k$ -corridor fully contains  $a \cap b$  if and only if the corresponding primal line crosses the corresponding primal  $k$ -triangle. The Lovász Lemma, in the primal, asserts that a line crosses at most  $O(n^2)$   $k$ -triangles. In the standard proof of the lemma, we translate a line from infinity towards the target line, and keep track of the number of  $k$ -triangles crossed by the line. This number changes only when the moving line sweeps through a segment connecting two input points, and we use the antipodality property to argue that the change in the number of crossed  $k$ -triangles at such an event is only  $\pm 1$ , from which the lemma follows. In the dual setup, antipodality is replaced by a suitable version of Lemma 7 (for planes), and the sweeping of the primal line becomes a sweeping of the dual line, moving in a vertical plane from, say  $+\infty$  towards the line  $a \cap b$ . The critical primal events are transformed into events where the moving line touches some intersection line  $c \cap d$ , for  $c, d \in \Lambda \setminus \{a, b\}$ . A suitable application of Lemma 7 then implies that the number of  $k$ -corridors that fully contain the moving curve changes by at most  $\pm 2$ .<sup>3</sup>

In the case of pseudoplanes, the sweeping is performed in the reverse order, from  $\gamma_{a,b}$  upwards to a curve at  $z = +\infty$ . More importantly, the sweeping is no longer by translating (a copy of)  $\gamma_{a,b}$ , but follows the *topological sweeping* paradigm of Edelsbrunner and Guibas [11] (see also [12]); the sweep curve is always fully contained in the curtain  $\Upsilon_{\gamma_{a,b}}$ .

In the context considered here, we have an arrangement of curves within  $\Upsilon_{\gamma_{a,b}}$ , so that each pair of them intersects once, and sweep it with a curve  $\gamma$ , so that initially, and at every instance during the sweep,  $\gamma$  intersects every other curve at most once. The sweep is a

<sup>3</sup> The reason why in the dual the change is  $\pm 2$  instead of  $\pm 1$  in the primal is that, for convenience in the presentation, we allow the point  $p$  in Lemma 7 to be somewhere at the middle of the curve; placing  $p$  at the “end” of the curve (at  $\pm\infty$ ) would make the change go down to  $\pm 1$ .

continuous motion of  $\gamma$ , given as a function  $\tau \rightarrow \gamma_\tau$ , for  $\tau \in \mathbb{R}^+$ , where  $\gamma_0 = \gamma$  is the initial placement of the sweeping curve and  $\gamma_\tau$  approaches the curve  $z = +\infty$  on  $\Upsilon_{\gamma_{a,b}}$  as  $\tau$  tends to  $\infty$ . Moreover,  $\gamma_\tau$  lies fully below  $\gamma_{\tau'}$ , for  $\tau < \tau'$ .

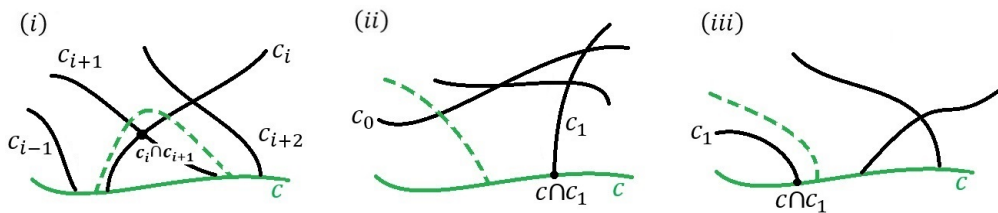
The sweeping curve  $\gamma$  is given a left-to-right orientation. Then  $\gamma$  intersects some subset of the other curves of  $\Gamma$  in some order. This ordered sequence changes when  $\gamma$  passes through a vertex of the arrangement or when the set of curves intersecting  $\gamma$  changes by an insertion or deletion of a curve, necessarily at the first or the last place in the sequence. We disregard the continuous nature of the sweep, and discretize it into a sequence of discrete steps, where each step represents one of these changes. As shown in [12], we have:

► **Lemma 8** (Hershberger and Snoeyink [12]). *Any planar arrangement of a set  $\Gamma$  of bi-infinite curves, any pair of which intersect at most once, can be swept topologically, starting with any curve  $\gamma \in \Gamma$ , so that, at any time during the sweep, the sweeping curve intersects any other curve at most once.*

Although in our case the sweep takes place within  $\Upsilon_{\gamma_{a,b}}$ , which is not a plane in general, we can flatten it in the  $x$ -direction into a plane, keeping the  $z$ -vertical direction unchanged, and thereby apply the topological sweeping machinery of [11, 12] within this curtain.

The sweeping mechanism, as described in [12], proceeds in a sequence of discrete steps, each of which implements one of three kinds of local moves, listed below, allowing to advance the sweeping curve past an intersection point, and to add or to remove curves from the set of curves intersected by the sweeping curve, without violating the 1-intersection property.

Let  $c \in \Gamma$  be the sweeping curve, and denote by  $\Xi(c) = (c_1, c_2, c_3, \dots)$  the sequence of curves of  $\Gamma$  that intersect  $c$ , sorted in the left-to-right order of their intersections along  $c$ . The basic steps of the sweep are of the following three types (see Figure 4):



■ **Figure 4** Operations by which the sweep progresses: (i) Passing over an empty triangle. (ii) Taking on the first ray. (iii) Passing over the first ray.

1. *Passing over an empty triangle:* We have a consecutive pair of curves  $c_i, c_{i+1}$  along  $c$  that intersect above  $c$ , and no other curve passes through the triangle formed by  $c, c_i, c_{i+1}$ . Then  $c$  can move past the intersection point of  $c_i, c_{i+1}$ . See Figure 4(i).
2. *Taking on the first ray:* We have a curve  $c_0$  that does not intersect  $c$ , but  $c_0$  and  $c$  are adjacent on the left (i.e., at  $x = -\infty$ ). Then we can move  $c$  upwards, make it intersect  $c_0$  at a point that lies to the left of all other intersection points, both on  $c$  and on  $c_0$ . This increases the intersection sequence  $\Xi(c)$  by one element, now its first element.
3. *Passing over the first ray:* Here the first intersection point along  $c$  is with a curve  $c_1$  so that  $c \cap c_1$  is also the first intersection along  $c_1$  from the left. Then  $c$  can move upwards, disentangling itself from  $c_1$ , and losing its intersection point with  $c_1$ , this time removing his first element of  $\Xi(c)$ .

As shown in [12], we can implement the sweep so that it only performs steps of these three types, and does not have to perform the symmetric operations to (ii) and (iii), of taking on or passing over the *last* ray.

We now establish the generalized dual version of the Lovász Lemma.

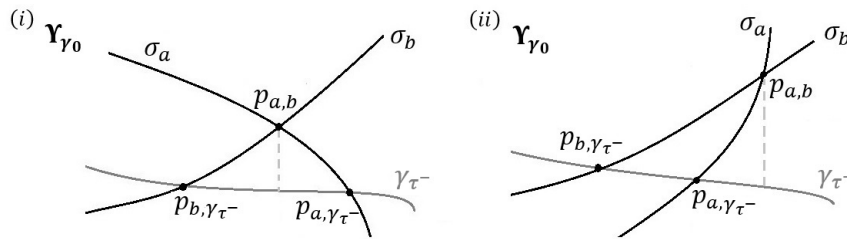
► **Lemma 9.** Any bi-infinite  $x$ -monotone curve  $\gamma_0$  such that (i)  $\gamma_0$  intersects each pseudoplane in  $\Lambda$  in exactly one point, and (ii) the  $xy$ -projection of  $\gamma_0$  intersects the  $xy$ -projection of any intersection curve of two surfaces of  $\Lambda$  at most once, is fully contained in at most  $n(n - 1)/2$  corridors in  $C^k$ .

**Proof.** Let  $\gamma_0$  be a curve as in the lemma, and consider the vertical curtain  $\Upsilon_{\gamma_0}$  that it spans. For each pseudoplane  $a \in \Lambda$ , denote by  $\sigma_a$  the intersection curve  $a \cap \Upsilon_{\gamma_0}$ , and put  $\Sigma = \{\sigma_a \mid a \in \Lambda\}$ . By the assumptions of the lemma, the collection of bi-infinite curves  $\{\gamma_0\} \cup \Sigma$  has the property that any two curves in this family intersect at most once. Moreover, as already remarked earlier, by regarding the  $xy$ -projection  $\gamma_0^*$  of  $\gamma_0$  as a homeomorphic copy of the real line, we can identify  $\Upsilon_{\gamma_0}$  as a homeomorphic copy of a vertical plane, where vertical lines are mapped to vertical lines. It follows that we can apply Lemma 8 to the arrangement of  $\{\sigma_a \mid a \in \Lambda\}$  within  $\Upsilon_{\gamma_0}$ , and conclude that this arrangement can be topologically swept with a curve that starts at  $\gamma_0$  and proceeds upwards, to infinity.

Denote by  $\gamma_\tau$  the sweeping curve at some moment  $\tau$ , where the curve coincides with  $\gamma_0$  at  $\tau = 0$ . At the beginning of the sweep,  $\gamma_0$  is fully contained in some number  $Y$  of  $k$ -corridors, and at the end of the sweep,  $\gamma_\tau$  is not contained in any of the corridors in  $C^k$ . We will establish an upper bound on the difference between the number of  $k$ -corridors that  $\gamma_\tau$  gets out of (i.e., stops being fully contained in) and the number of  $k$ -corridors that it gets into (i.e., starts being fully contained in), at any critical event during the sweep. Summing those differences will yield the asserted upper bound on  $Y$ .

Consider  $\gamma_\tau$  at some instance  $\tau$  during the sweep, and let  $a \in \Lambda$ . If  $\sigma_a$  is fully above  $\gamma_\tau$ , we get that  $\gamma_0$ , which is obtained by some motion of  $\gamma_\tau$  downwards, is fully below  $a$ , a contradiction to the assumption that  $\gamma_0$  intersects all the pseudoplanes in  $\Lambda$ . Therefore, each pseudoplane in  $\Lambda$  is either fully below  $\gamma_\tau$ , or intersects it (exactly once). Hence, during the sweeping from  $\gamma_0$ , the only valid sweeping steps are passing over an empty triangle and passing over the first ray.

Clearly, it suffices to consider what happens at instances  $\tau$  at which  $\gamma$  is about to pass through a vertex of the arrangement of  $\{\sigma_a \mid a \in \Lambda\}$  on  $\Upsilon_{\gamma_0}$ , or at instances at which  $\gamma_\tau$  is about to pass over the first ray. So let  $\tau^-$  and  $\tau^+$  denote instances immediately before and after such a critical transition. We distinguish between three types of sweeping steps.



■ **Figure 5** (i) The intersection point  $p_{a,b} = \sigma_a \cap \sigma_b$  is directly above the curve  $\gamma_{\tau^-}$  somewhere between  $p_{a,\gamma_{\tau^-}} = \sigma_a \cap \gamma_{\tau^-}$  and  $p_{b,\gamma_{\tau^-}} = \sigma_b \cap \gamma_{\tau^-}$ . (ii) The intersection point  $p_{a,b}$  is not directly above any point on the curve  $\gamma_{\tau^-}$  between  $p_{a,\gamma_{\tau^-}}$  and  $p_{b,\gamma_{\tau^-}}$ .

**Case 1:** The transition at  $\tau$  is that we pass over an empty triangle, defined by some pair of curves  $\sigma_a, \sigma_b$  and  $\gamma_{\tau^-}$ , such that the point on  $\gamma_{\tau^-}$  directly below the intersection point  $p_{a,b}$  is somewhere between  $p_{a,\gamma_{\tau^-}}$  and  $p_{b,\gamma_{\tau^-}}$  (see Figure 5(i)). Since  $\gamma_\tau$  intersects each of



$\sigma_a, \sigma_b$  at most once, almost all of the curve  $\gamma_{\tau^-}$  is between the lower envelope and the upper envelope of  $\{\sigma_a, \sigma_b\}$ , except for its portion between  $p_{a,\gamma_{\tau^-}}$  and  $p_{b,\gamma_{\tau^-}}$ . Since the triangle defined by  $\sigma_a, \sigma_b$  and  $\gamma_{\tau^-}$  is empty, each curve in  $\Sigma$  that lies below  $p_{a,b}$ , defines a corridor with  $\sigma_a, \sigma_b$ , such that  $\gamma_{\tau^-}$  lies fully in that corridor. Symmetrically,  $\gamma_{\tau^+}$ , for  $\tau^+$  sufficiently close to  $\tau$ , lies fully in each corridor defined by  $\sigma_a, \sigma_b$  and a curve in  $\Sigma$  that passes above  $p_{a,b}$  (see, e.g., Figure 6(i)). Hence, by Lemma 7, the absolute value of the difference between the number of  $k$ -corridors that  $\gamma_{\tau}$  gets out of at  $\tau$ , and the number of  $k$ -corridors that  $\gamma_{\tau}$  gets into, is at most 2.

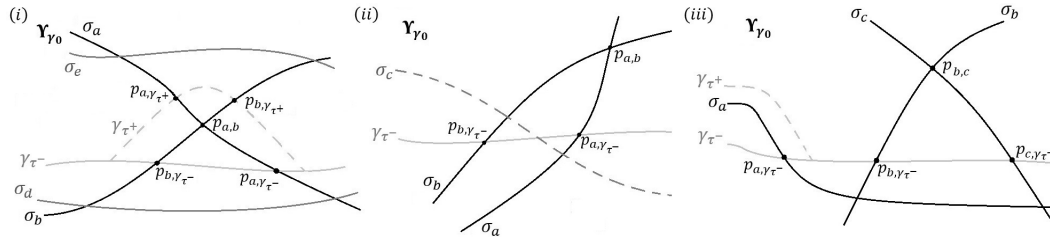


Figure 6 The three cases of critical events during the sweep.

**Case 2:** The transition at  $\tau$  is that we pass over an empty triangle, defined by  $\sigma_a, \sigma_b$  and  $\gamma_{\tau^-}$ , where now the vertical projection of the intersection point  $p_{a,b}$  onto  $\gamma_{\tau^-}$  is not between  $p_{a,\gamma_{\tau^-}}$  and  $p_{b,\gamma_{\tau^-}}$ , but lies on one side of both, say past  $p_{a,\gamma_{\tau^-}}$  to the right (see Figure 5(ii)). We claim that neither  $\gamma_{\tau^-}$  nor  $\gamma_{\tau^+}$  is fully contained in any corridor  $C_{a,b,c}$ , for  $\sigma_c \in \Sigma$ . Indeed (refer again to Figure 5), by Lemma 4, the half-curve emanating from  $p_{a,\gamma_{\tau^-}}$  on  $\gamma_{\tau^-}$  to the right is below the lower envelope of  $\{\sigma_a, \sigma_b\}$ , and the half-curve emanating from  $p_{b,\gamma_{\tau^-}}$  on  $\gamma_{\tau^-}$  to the left is above the upper envelope of  $\{\sigma_a, \sigma_b\}$ . Hence, in order for  $\gamma_{\tau^-}$  to be fully contained in a corridor  $C_{a,b,c}$  for some  $\sigma_c \in \Sigma$ ,  $\sigma_c$  must pass above  $p_{b,\gamma_{\tau^-}}$  and below  $p_{a,\gamma_{\tau^-}}$ , and therefore it must intersect the triangle defined by  $\sigma_a, \sigma_b$  and  $\gamma_{\tau^-}$  (see Figure 6(ii)). Since this triangle is empty, there is no such  $\sigma_c$ . Symmetrically<sup>4</sup>,  $\gamma_{\tau^+}$  is not contained in any corridor  $C_{a,b,c}$  for any  $c$ . Hence, at this step in the sweeping process, there is no change in the set of corridors that fully contain  $\gamma_{\tau}$ .

**Case 3:** The transition at  $\tau$  is passing over the first ray, belonging to some  $\sigma_a \in \Sigma$ . We claim that here too  $\gamma_{\tau^-}$  and  $\gamma_{\tau^+}$  are fully contained in the same corridors. Indeed, except for the left ray of  $\sigma_a$ ,  $\gamma_{\tau^-}$  and  $\gamma_{\tau^+}$  are fully above  $\sigma_a$ . Moreover, the only corridors that  $\gamma$  can get into or out of at this transition must involve  $\sigma_a$ . Let  $C_{a,b,c}$  be such a corridor. If  $\sigma_a$  appears on both the upper and the lower envelopes of  $\{\sigma_a, \sigma_b, \sigma_c\}$  then, as is easily checked, neither  $\gamma_{\tau^-}$  nor  $\gamma_{\tau^+}$  can be fully contained in  $C_{a,b,c}$ . Hence,  $\sigma_a$  must appear on exactly one of the envelopes, and then it must appear there as the middle portion of the envelope (see Figure 6(iii)). But then the left ray of  $\sigma_a$  over which  $\gamma$  is swept cannot appear on either envelope, so the transition does not cause  $\gamma$  to enter or leave  $C_{a,b,c}$ , as claimed.

In summary, the only kind of step during the sweep in which the number of  $k$ -corridors that the sweeping curve is contained in changes, is passing over an empty triangle with the structure considered in Case 1 and then, as argued above, this number can change by at most 2. There are  $\binom{n}{2} = \frac{n(n-1)}{2}$  intersection points on  $\Upsilon_{\gamma_0}$ , since each pair of curves in  $\Sigma$

<sup>4</sup> Indeed, right after the transition,  $\sigma_a, \sigma_b$  and  $\gamma_{\tau^+}$  form an empty triangle above  $p_{a,b}$ , with similar properties that allow us to apply a symmetric variant of the argument just presented.



intersects at most once. Assume without loss of generality that at most half of them are above  $\gamma_0$  (otherwise, consider sweeping  $\gamma_0$  in the opposite direction, downwards to infinity). The sweeping curve can encounter at most  $\frac{1}{2} \cdot \frac{n(n-1)}{2}$  empty triangles whose middle vertex lies above the opposite edge. Thus, at the beginning of the process,  $\gamma_0$  is fully contained in at most  $\frac{n(n-1)}{2}$   $k$ -corridors and the claim follows.  $\blacktriangleleft$

As a corollary (see also the full version [17]), we obtain the following upper bound on  $X^k$ :

► **Lemma 10.** *The number  $X^k$  of ordered pairs of  $k$ -corridors such that the first corridor is immersed in the second one, in the arrangement  $\mathcal{A}(\Lambda)$ , is at most  $\frac{3n^4}{4}$ .*

**Proof.** Fix an intersection curve  $\gamma_{a,b} = a \cap b$  of two pseudoplanes from  $\Lambda$ . By Lemma 9,  $\gamma_{a,b}$  is fully contained in at most  $n(n-1)/2$   $k$ -corridors. For each containing  $k$ -corridor  $C_{c,d,e}$ ,  $\gamma_{a,b}$  can contribute at most three ordered pairs to  $X^k$ , namely an immersion of  $C_{a,b,c}$  in  $C_{c,d,e}$ , of  $C_{a,b,d}$  in  $C_{c,d,e}$ , and of  $C_{a,b,e}$  in  $C_{c,d,e}$ . Since there are only  $\binom{n}{2}$  intersection curves in  $\mathcal{A}(\Lambda)$ , we get that there are at most  $3 \frac{n(n-1)}{2} \binom{n}{2} < \frac{3n^4}{4}$  ordered pairs of immersed  $k$ -corridors.  $\blacktriangleleft$

## 2.2 The dual version of the Crossing Lemma

In this subsection we derive a lower bound on  $X^k$ , using a dual version of the Crossing Lemma (see [4]), extended to the case of pseudoplanes. For each pseudoplane  $a \in \Lambda$ , denote by  $z_a$  the intersection point of  $a$  with the  $z$ -axis. We can choose the position of the  $z$ -axis so as to ensure that (a) all the values  $z_a$  are distinct and (b) for each  $a \in \Lambda$ ,  $z_a$  lies above (in the  $y$ -direction of the  $xy$ -projection of  $a$ ) all the intersection curves  $\gamma_{a,b}$ , for  $b \in \Lambda \setminus \{a\}$ .

► **Definition 11.** *Let  $a \in \Lambda$ . Denote by  $\Gamma_a$  the collection of the intersection curves of  $a$  and the other pseudoplanes  $b \in \Lambda$  with  $z_b > z_a$ . That is,  $\Gamma_a = \{\gamma_b := a \cap b \mid b \in \Lambda \setminus \{a\}, z_b > z_a\}$ .*

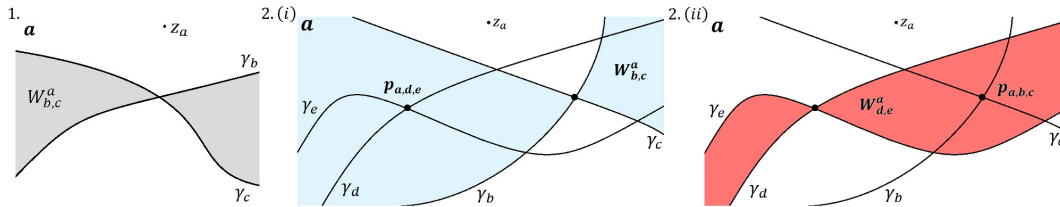
By the assumptions on  $\Lambda$ , the  $xy$ -projection of any intersection curve of two pseudoplanes in  $\Lambda$  is an  $x$ -monotone curve. Therefore,  $\Gamma_a$  forms a family of  $x$ -monotone curves on the surface  $a$ . Since  $a$  is the graph of a bivariate continuous function, it will be convenient to identify it with its  $xy$ -projection, and think of it, for the purpose of the current analysis, as a horizontal plane. Each pair of curves from  $\Gamma_a$  intersects exactly once, because each triple of pseudoplanes in  $\Lambda$  intersects exactly once. Each curve in  $\Gamma_a$  is bi-infinite and divides  $a$  into two unbounded regions. These considerations allow us to interpret  $\Gamma_a$  as a family of  $x$ -monotone pseudolines in the plane.

► **Definition 12.** *Let  $a \in \Lambda$ , and let  $\Gamma_a$  be as above. Each  $d \in \Lambda \setminus \{a\}$  for which  $\gamma_d \in \Gamma_a$  divides  $a$  into two disjoint regions: the region  $a_d^-$  on  $a$  that is fully above the pseudoplane  $d$ , and the region  $a_d^+$  on  $a$  that is fully below  $d$  (so  $a_d^-$  means that  $d$  is below  $a$ , and  $a_d^+$  means that  $d$  is above  $a$ ). These two regions are delimited by the intersection curve  $\gamma_d$  on  $a$ . Note that  $z_a \in a_d^+$ . That is,  $a_d^+$  is the region that lies above (in the  $y$ -direction) the intersection curve  $\gamma_d$ , and  $a_d^-$  is the region below  $\gamma_d$ .*

For each pair of distinct pseudoplanes  $b, c \in \Lambda \setminus \{a\}$  such that  $\gamma_b, \gamma_c \in \Gamma_a$ , define the  **$x$ -horizontal wedge  $W_{b,c}^a$**  as the region on the pseudoplane  $a$  that is contained in exactly one of the two regions  $a_b^+, a_c^+$ , that is, in exactly one of the regions that are bounded by  $\gamma_b, \gamma_c$  and contain  $z_a$  (see Figure 7(1)).

Note that our assumption on the position of  $z_a$  in  $a$  allows to regard the wedges  $W_{b,c}^a$  as being indeed “ $x$ -horizontal” within the  $xy$ -frame in  $a$ .

Continue to fix the pseudoplane  $a$ , let  $E_a$  be some subset of vertices of  $\mathcal{A}(\Gamma_a)$ , and let  $G_a = (\Gamma_a, E_a)$  denote the graph whose vertices are the pseudolines in  $\Gamma_a$  and whose edges are the pairs that form the vertices of  $E_a$ . A **diamond** in  $G_a$  is two pairs  $\{\gamma_b, \gamma_c\}, \{\gamma_d, \gamma_e\}$  of curves of  $\Gamma_a$  on  $a$ , both pairs belonging to  $E_a$ , with all four pseudoplanes  $b, c, d, e$  distinct, such that  $p_{a,b,c} = \gamma_b \cap \gamma_c \in W_{d,e}^a$  and  $p_{a,d,e} = \gamma_d \cap \gamma_e \in W_{b,c}^a$ . See Figure 7(2.i) and 7(2.ii).



**Figure 7** 1. The  $x$ -horizontal wedge  $W_{b,c}^a$  on the pseudoplane  $a$ . The regions  $a_b^+$  and  $a_c^+$  lie above (in the  $y$ -direction) the respective curves  $\gamma_b$  and  $\gamma_c$ , and they both contain  $z_a$ . 2. The two pairs  $\{\gamma_b, \gamma_c\}, \{\gamma_d, \gamma_e\}$  on  $a$  form a diamond in  $G_a$ . (i) The blue area is the  $x$ -horizontal wedge  $W_{b,c}^a$ , and it contains  $p_{a,d,e} = \gamma_d \cap \gamma_e$ . (ii) The red area is the  $x$ -horizontal wedge  $W_{d,e}^a$ , and it contains  $p_{a,b,c} = \gamma_b \cap \gamma_c$ .

The following is our version of an extension of the dual version of Euler’s formula for planar maps, derived in Tamaki and Tokuyama [18], for the case of pseudolines in  $\mathbb{R}^2$ :

► **Lemma 13** (Tamaki and Tokuyama [18]). *For a pseudoplane  $a \in \Lambda$ , let  $G_a$  be as defined above, with  $|\Gamma_a| > 3$ . If  $\Gamma_a$  is diamond-free, then  $G_a$  is planar, and so  $|E_a| \leq 3|\Gamma_a| - 6$ .*

As a corollary of Lemma 13, we have:

► **Lemma 14** (Generalized dual version of the Crossing Lemma). *Let  $\Gamma_a$  and  $G_a$  be as above, so that  $|E_a| > 4|\Gamma_a|$ . The number of diamonds in  $G_a$  is at least  $\frac{|E_a|^3}{64|\Gamma_a|^2}$ .*

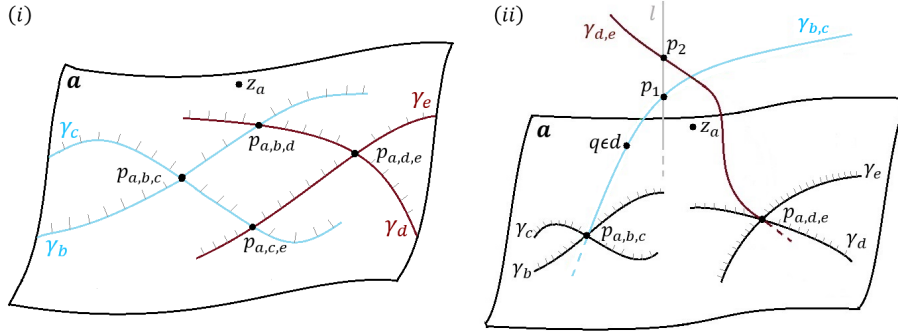
The proof follows more or less the standard probabilistic proof of the Crossing Lemma. That is, Lemma 13 easily implies that the number of diamonds in  $G_a$  is at least  $|E_a| - 3|\Gamma_a| + 6 > |E_a| - 3|\Gamma_a|$ . One then draws a random sample  $G'_a$  of  $G_a$ , by picking each curve  $\gamma_b \in \Gamma_a$  independently with probability  $p = 4|\Gamma_a|/|E_a|$ , and applies the above bootstrapping bound to  $G'_a$ , to obtain the improved bound in the lemma. This works, as in the planar case, provided that  $|E_a| > 4|\Gamma_a|$ , as the lemma assumes.

We now specialize this result to our context. For each  $a \in \Lambda$ , consider the set  $E_a^k = \{p_{a,b,c} = \gamma_b \cap \gamma_c \mid \gamma_b, \gamma_c \in \Gamma_a, C_{a,b,c} \in C^k\}$ , and the graph  $G_a^k = (\Gamma_a, E_a^k)$  defined as above. Lemma 14 implies the following:

► **Lemma 15.** *The number  $X^k$  of ordered pairs of immersed  $k$ -corridors in the arrangement  $\mathcal{A}(\Lambda)$  is at least  $\frac{|C^k|^3}{64n^4} - n^2$ .*

**Proof.** Let  $a \in \Lambda$ ,  $G_a^k = (\Gamma_a, E_a^k)$  be as above, and define  $\Delta_a$  as the number of diamonds in  $G_a^k$ . Let  $\{\gamma_b, \gamma_c\}, \{\gamma_d, \gamma_e\}$  be a pair that form a diamond. Since the pseudoplanes in  $\Lambda$  satisfy property (iv) and are in general position, the  $xy$ -projections of the curves  $\gamma_{b,c}$  and  $\gamma_{d,e}$  have exactly one intersection point (but  $\gamma_{b,c}$  and  $\gamma_{d,e}$  do not intersect in 3-space). Moreover, since  $b, c$  are the graphs of total bivariate functions, the projection of their intersection curve  $\gamma_{b,c}$  on  $a$  is fully contained in the region  $\{p_{a,b,c}\} \cup \{a_b^+ \cap a_c^+\} \cup \{a_b^- \cap a_c^-\}$ . That is, the projection of  $\gamma_{b,c}$  is disjoint from the interior of  $W_{b,c}^a$ . Similarly, the projection of the intersection curve  $\gamma_{d,e}$  on  $a$  is fully contained in the region  $\{p_{a,d,e}\} \cup \{a_d^+ \cap a_e^+\} \cup \{a_d^- \cap a_e^-\}$ , and is disjoint from  $W_{d,e}^a$ . In addition, the portion of  $\gamma_{b,c}$  that projects to  $a_b^+ \cap a_c^+$  lies above  $a$  and the portion projecting to  $a_b^- \cap a_c^-$  lies below  $a$ . A similar property holds for  $\gamma_{d,e}$ .

Assume without loss of generality that the pseudoplanes  $b, c, d, e$  intersect  $a$  as in Figure 8(i); that is,  $p_{a,b,c}$  is contained in  $a_d^- \cap a_e^+$  and  $p_{a,d,e}$  is contained in  $a_b^- \cap a_c^+$ . Since  $\{\gamma_b, \gamma_c\}$  and  $\{\gamma_d, \gamma_e\}$  form a diamond and each pair of curves on  $a$  intersects exactly once, the intersection of the boundary of  $a_d^+ \cap a_e^+$  and the boundary of  $a_b^- \cap a_c^-$  is empty. Indeed, the interior of the arc  $p_{a,b,c}p_{a,d,e}$  is fully contained in  $a_b^- \cap a_c^+$ , and the half-curve of  $\gamma_d$  emanating from  $p_{a,b,d}$  and not containing  $p_{a,d,e}$ , is fully contained in  $a_b^+$  (otherwise,  $\gamma_d$  would intersect  $\gamma_b$  more than once). On the other hand, the half-curve of  $\gamma_e$  emanating from  $p_{a,d,e}$  and not containing  $p_{a,c,e}$ , is fully contained in  $a_c^+$ , since  $\gamma_c$  already intersects the other half-curve of  $\gamma_e$ . These two observations establish our claim. The regions  $a_b^- \cap a_c^-$  and  $a_d^+ \cap a_e^+$  are not contained in one another, and therefore their intersection is empty. Similarly, The intersection of  $a_b^+ \cap a_c^+$  and  $a_d^- \cap a_e^-$  is empty.



**Figure 8** The two pairs  $\{\gamma_b, \gamma_c\}, \{\gamma_d, \gamma_e\}$  on  $a$  form a diamond in  $G_a^k$ . (i) The diamond, where  $p_{a,b,c}$  is contained in  $a_d^- \cap a_e^+$  and  $p_{a,d,e}$  is contained in  $a_b^- \cap a_c^+$ . (ii) The intersection curve  $\gamma_{b,c}$  is below the intersection curve  $\gamma_{d,e}$ . The pseudoplane  $d$  (resp.,  $e$ ) meets  $\gamma_{b,c}$  at a point  $q$  between  $p_{a,b,c}$  and  $p_1$  (resp., at a point, not drawn, outside this arc).

Assume without loss of generality that  $\gamma_{b,c}$  passes below  $\gamma_{d,e}$ . That is, letting  $l$  denote the unique  $z$ -vertical line that meets both  $\gamma_{b,c}, \gamma_{d,e}$ , the points  $p_1 = l \cap \gamma_{b,c}, p_2 = l \cap \gamma_{d,e}$  satisfy  $z_{p_1} < z_{p_2}$ . Assume without loss of generality that  $l$  intersects  $a$  in the region on  $a$  that is the intersection of  $a_b^+, a_c^+, a_d^+, a_e^+$  (the regions on  $a$  induced by the curves  $\gamma_b, \gamma_c, \gamma_d, \gamma_e$  and containing  $z_a$ ). The case where  $l$  intersects  $a$  in the region on  $a$  that is the intersection of  $a_b^-, a_c^-, a_d^-, a_e^-$ , is handled symmetrically. These are the only two possibilities, since the intersection of  $a_b^+, a_c^+, a_d^-, a_e^-$  is empty, and so is the intersection of  $a_b^-, a_c^-, a_d^+, a_e^+$ .

Since  $\gamma_{d,e}$  is above  $p_1$ , it follows that both  $d$  and  $e$  themselves are above  $p_1$ . Moreover, since  $p_{a,b,c}$  lies in  $a_d^-$ ,  $d$  must lie below  $p_{a,b,c}$ . Hence  $d$  must intersect  $\gamma_{b,c}$  at some point  $q$  between  $p_{a,b,c}$  and  $p_1$ . Moreover, since  $e$  satisfies  $z_e > z_a$  and  $p_{a,b,c}$  lies in  $a_e^+$ , as in Figure 8(ii),  $e$  is above  $p_{a,b,c}$ . Since  $e$  is also above  $p_1$  and  $e$  is the graph of a bivariate continuous function, its single intersection point with  $\gamma_{b,c}$  must be outside the arc  $p_{a,b,c}p_1$  of  $\gamma_{b,c}$ .

We claim that  $\gamma_{b,c} \subseteq C_{a,d,e}$ . Indeed,  $\gamma_{b,c}$  is fully above the lower envelope of  $\{a, d, e\}$ : the half-curve of  $\gamma_{b,c}$  that emanates from  $p_{a,b,c}$  and contains  $p_1$  lies above the pseudoplane  $a$  ( $\gamma_{b,c}$  intersects  $a$  at  $p_{a,b,c}$ ), and the complementary half-curve lies above  $d$ , because the intersection point  $q$  of  $\gamma_{b,c}$  and  $d$  lies between  $p_{a,b,c}$  and  $p_1$ . The intersection curve  $\gamma_{b,c}$  also lies fully below the upper envelope of  $\{a, d, e\}$ . That is because (i) the half-curve of  $\gamma_{b,c}$  that emanates from the intersection point  $q$  of  $\gamma_{b,c}$  and  $d$ , and contains  $p_1$ , lies below the pseudoplane  $d$ , since  $p_2 \in d$  is higher than  $p_1$ ; (ii) the half-curve of  $\gamma_{b,c}$  that emanates from  $p_{a,b,c}$  and does not contain  $p_1$ , lies below the pseudoplane  $a$  (again,  $\gamma_{b,c}$  intersects  $a$  at  $p_{a,b,c}$ ); and (iii) the arc  $p_{a,b,c}q$  is below  $e$ , since  $e$  is above both  $p_{a,b,c}, p_1$  and therefore must be above the complete arc  $p_{a,b,c}p_1$ , and in particular  $e$  is above the smaller arc  $p_{a,b,c}q$ . The other cases behave similarly and lead to similar conclusions.

Thus for each pair  $\{\gamma_b, \gamma_c\}, \{\gamma_d, \gamma_e\}$  that form a diamond, either  $\gamma_{b,c} \subseteq C_{a,d,e}$ , or  $\gamma_{d,e} \subseteq C_{a,b,c}$ . Either way, one of the corridors  $C_{a,b,c}, C_{a,d,e}$  is immersed in the other one. Notice that every diamond in  $\{G_a^k\}_{a \in \Lambda}$  yields a distinct ordered pair of immersed  $k$ -corridors, because for each  $k$ -corridor  $C_{a,b,c}$ , the intersection point  $p_{a,b,c} = a \cap b \cap c$  represents an edge of only the graph associated with the pseudoplane with the lowest intersection point with the  $z$ -axis. Hence, by the dual version of the Crossing Lemma, namely Lemma 14, we have  $X^k \geq \sum_a \frac{|E_a^k|^3}{64|\Gamma_a|^2}$ , where the sum is over all those  $a$  for which  $|E_a^k| \geq 4|\Gamma_a|$ . Any other pseudoplane  $a$  satisfies  $\frac{|E_a^k|^3}{64|\Gamma_a|^2} \leq |\Gamma_a|$ , which implies the somewhat weaker lower bound

$$X^k \geq \sum_{a \in \Lambda} \left( \frac{|E_a^k|^3}{64|\Gamma_a|^2} - |\Gamma_a| \right). \tag{1}$$

By the definition of  $G_a^k$ , and as just noted, each  $k$ -corridor  $C_{a,b,c}$  in  $\mathcal{A}(\Lambda)$  appears in exactly one of  $E_a^k, E_b^k, E_c^k$  (in the graph of the pseudoplane that intersects the  $z$ -axis at the lowest point among the three). Thus,  $\sum_{a \in \Lambda} |E_a^k| = |C^k|$ . The number of curves in  $\Gamma_a$  is at most  $n - 1$ . Therefore, using (1) and Hölder’s inequality, we get the following lower bound

$$X^k \geq \sum_{a \in \Lambda} \left( \frac{|E_a^k|^3}{64|\Gamma_a|^2} - |\Gamma_a| \right) \geq \frac{1}{64n^2} \sum_{a \in \Lambda} |E_a^k|^3 - n^2 \geq \frac{1}{64n^2} \cdot \frac{\left( \sum_{a \in \Lambda} |E_a^k| \right)^3}{n^2} - n^2 = \frac{|C^k|^3}{64n^4} - n^2. \blacktriangleleft$$

### 2.3 The complexity of the $k$ -level of $\mathcal{A}(\Lambda)$

We are now ready to obtain the upper bound on the complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$ .

► **Lemma 16.** *The complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$  is  $O(n^{8/3})$ .*

**Proof.** We compare the upper bound in Lemma 10 and the lower bound in Lemma 15 for the number  $X^k$  of ordered pairs of immersed  $k$ -corridors in  $\mathcal{A}(\Lambda)$ , and get:

$$\frac{3n^4}{4} \geq X^k \geq \frac{|C^k|^3}{64n^4} - n^2.$$

Hence we get that  $|C^k|^3 \leq 48n^8 + 64n^6$ , which implies that  $|C^k| = O(n^{8/3})$ . The number of  $k$ -corridors is the number of vertices of  $\mathcal{A}(\Lambda)$  at level  $k$ , which implies that the complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$  is  $O(n^{8/3})$ . ◀

Combining the upper bound in Lemma 16 with the general technique of [1], we get the following  $k$ -sensitive result, whose proof is reviewed in the full version [17].

► **Theorem 17.** *The complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$  is  $O(nk^{5/3})$ .*

## 3 Discussion

In this paper we have shown that, for any set  $\Lambda$  of  $n$  surfaces in  $\mathbb{R}^3$  that form a family of pseudoplanes, in the sense of satisfying properties (i)–(iv) of Section 2, the complexity of the  $k$ -level of  $\mathcal{A}(\Lambda)$  is  $O(nk^{5/3})$ . Our analysis is based on ingredients from the technique of [15], for the primal version of bounding the number of  $k$ -sets in a set of  $n$  points in  $\mathbb{R}^3$ . The upper bound established in [15] is  $O(nk^{3/2})$ , and is thus better than the bound we obtain here, for the case of (planes and) pseudoplanes (see full version [17]). The main reason for following this weaker analysis is the availability of the result of Tamaki and Tokuyama [18] on

diamond-free graphs in arrangements of pseudolines, which leads to an extended dual version of the Crossing Lemma. It is definitely an intriguing hopefully not too difficult, challenge to extend, to the case of pseudoplanes, a dual version of the sharper analysis in [15].

Another line of research is to relax one or more of properties (i)–(iv), defining a family of pseudoplanes, as described in Section 2, with the goal of extending our analysis and obtaining nontrivial bounds for the complexity of the  $k$ -level in arrangements of more general surfaces. Property (iv) seems to be the most restrictive property among the four, namely requiring the  $xy$ -projections of all intersection curves from  $\Lambda$  to form a family of pseudolines in the plane (although it trivially holds for the case of planes). The main use of this property in our analysis is in proving a generalized dual version of the Lovász Lemma (Lemma 9), as it (a) facilitates the applicability of topological sweeping, and (b) allows us to exploit the extended notion of antipodality, as in Lemma 7. It is an interesting challenge to find refined techniques that can extend this analysis to situations where the arrangement within the curtain is not an arrangement of pseudolines. One open direction is to find a different proof technique of the Lovász Lemma that is not based on sweeping. This would also be very interesting for the original case of planes (or of lines in the plane).

In studying the complexity of a level in an arrangement of more general surfaces, how far can we relax the constraints that these surfaces must satisfy in order to enable us to obtain sharp (significantly subcubic) bounds on the complexity of a level?

Finally, can our technique be extended to higher dimensions? For example, can we obtain a sharp bound in four dimensions, similar to the bound in Sharir [14] for  $k$ -sets in  $\mathbb{R}^4$ ?

---


## References

- 1 P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete Comput. Geom.*, 19:315–331, 1998.
- 2 P. K. Agarwal and M. Sharir. *Davenport-Schinzel Sequences and Their Geometric Applications*, pages 216–217. Cambridge University Press, NY, USA, 1995.
- 3 P. K. Agarwal and M. Sharir. Pseudo-line arrangements: duality, algorithms, and applications. *SIAM J. Comput.*, pages 34:526–552, 2005.
- 4 M. Ajtai, V. Chvátal, M. M. Newborn, and E. Szemerédi. *Crossing-free subgraphs*, pages 9–12. North-Holland Mathematics Studies, Amsterdam, 1982.
- 5 I. Bárány, Z. Füredi, and L. Lovász. On the number of halving planes. *Combinatorica*, 10:175–183, 1990.
- 6 T. M. Chan. On the bichromatic  $k$ -set problem. *ACM Transactions on Algorithms*, 6(4):62:1–62:20, 2010.
- 7 T. M. Chan. On levels in arrangements of surfaces in three dimensions. *Discrete Comput. Geom.*, 48:1–18, 2012.
- 8 T. K. Dey. Improved bounds on planar  $k$ -sets and related problems. *Discrete Comput. Geom.*, 19:373–382, 1998.
- 9 T. K. Dey and H. Edelsbrunner. Counting triangle crossing and halving planes. *Discrete Comput. Geom.*, 12:281–289, 1994.
- 10 H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, pages 271–291. Springer Verlag, Heidelberg, 1987.
- 11 H. Edelsbrunner and L. Guibas. Topologically sweeping an arrangement. *J. Computer Systems Sci.*, 38:165–194, 1989.
- 12 J. Hershberger and J. Snoeyink. Sweeping arrangements of curves. *Proc. 5th ACM Sympos. on Computational Geometry*, pages 354–363, 1989.
- 13 G. Nivasch. An improved, simple construction of many halving edges. *Contemporary Mathematics*, 453:299–306, 2008.

- 14 M. Sharir. An improved bound for  $k$ -sets in four dimensions. *Combinatorics, Probability and Computing*, 20:119–129, 2011.
- 15 M. Sharir, S. Smorodinsky, and G. Tardos. An improved bound for  $k$ -sets in three dimensions. *Discrete Comput. Geom.*, 26:195–204, 2001.
- 16 M. Sharir and J. Zahl. Cutting algebraic curves into pseudo-segments and applications. *J. Combinat. Theory, Ser. A*, pages 150:37–42, 2017.
- 17 M. Sharir and C. Ziv. On the complexity of the  $k$ -level in arrangements of pseudoplanes, 2019. [arXiv:1903.07196](https://arxiv.org/abs/1903.07196).
- 18 H. Tamaki and T. Tokuyama. A characterization of planar graphs by pseudo-line arrangements. *Algorithmica*, 35:269–285, 2003.
- 19 G. Tóth. Point sets with many  $k$ -sets. *Proc. 16th Annual Sympos. on Computational Geometry*, 35:37–42, 2000.
- 20 C. Ziv. On the Complexity of the  $k$ -Level in Arrangements of Pseudoplanes. Master's thesis, School of Computer Science, Tel Aviv University, 2019.



# Packing Geometric Objects with Optimal Worst-Case Density

**Aaron T. Becker** 


Department of Electrical and Computer Engineering, University of Houston  
Houston, TX 77204-4005 USA  
atbecker@uh.edu

**Sándor P. Fekete** 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
s.fekete@tu-bs.de

**Phillip Keldenich** 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
p.keldenich@tu-bs.de

**Sebastian Morr** 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
sebastian@morr.cc

**Christian Scheffer** 

Department of Computer Science, TU Braunschweig  
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany  
c.scheffer@tu-bs.de

---

## Abstract

We motivate and visualize problems and methods for packing a set of objects into a given container, in particular a set of different-size circles or squares into a square or circular container. Questions of this type have attracted a considerable amount of attention and are known to be notoriously hard. We focus on a particularly simple criterion for deciding whether a set can be packed: comparing the total area  $A$  of all objects to the area  $C$  of the container. The *critical packing density*  $\delta^*$  is the largest value  $A/C$  for which any set of area  $A$  can be packed into a container of area  $C$ . We describe algorithms that establish the critical density of squares in a square ( $\delta^* = 0.5$ ), of circles in a square ( $\delta^* = 0.5390\dots$ ), regular octagons in a square ( $\delta^* = 0.5685\dots$ ), and circles in a circle ( $\delta^* = 0.5$ ).

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Packing and covering problems; Theory of computation  $\rightarrow$  Computational geometry

**Keywords and phrases** Packing, complexity, bounds, packing density

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.63

**Category** Multimedia Exposition

**Related Version** Parts of this contribution describe the conference paper [3], which is part of SoCG 2019; a full version of that paper can be found at <https://arxiv.org/abs/1903.07908>.

**Funding** *Aaron T. Becker*: Supported by the National Science Foundation under Grant No. IIS-1553063.

*Phillip Keldenich*: Supported by the German Research Foundation under Grant No. FE 407/17-2.



© Aaron T. Becker, Sándor P. Fekete, Phillip Keldenich, Sebastian Morr, and Christian Scheffer;

licensed under Creative Commons License CC-BY

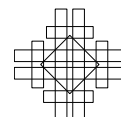
35th International Symposium on Computational Geometry (SoCG 2019).

Editors: Gill Barequet and Yusu Wang; Article No. 63; pp. 63:1–63:6

Leibniz International Proceedings in Informatics



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



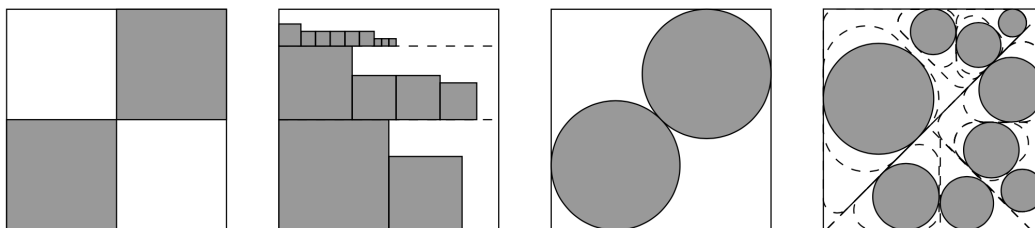


## 1 Introduction

Deciding whether a set of disks can be packed into a given container is a fundamental geometric optimization problem that has attracted considerable attention. Disk packing also has numerous applications in engineering, science, operational research and everyday life, e.g., for packaging cylinders [1, 8], bundling tubes or cables [22, 24], or the cutting industry [23]. Further applications stem from forestry [23] and origami design [13].

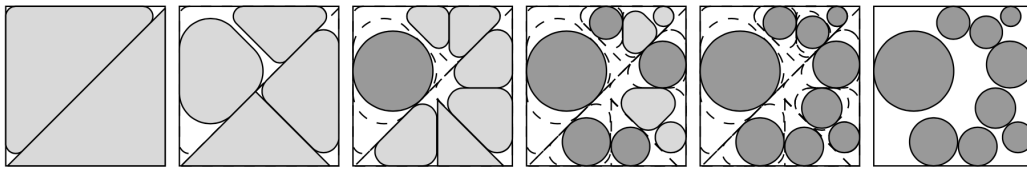
Like many other packing problems, disk packing is typically quite difficult; what is more, the combinatorial hardness is compounded by the geometric complications of dealing with irrational coordinates that arise when packing circular objects. This is reflected by the limitations of provably optimal results for the optimal value for the smallest sufficient disk container (and hence, the densest such disk packing in a disk container), a problem that was discussed by Kravitz [12] in 1967: Even when the input consists of just 13 unit disks, the optimal value for the densest disk-in-disk packing was only established in 2003 [7], while the optimal value for 14 unit disks is still unproven. The enormous challenges of establishing densest disk packings are also illustrated by a long-standing open conjecture by Erdős and Oler from 1961 [19] regarding optimal packings of  $n$  unit disks into an equilateral triangle, which has only been proven up to  $n = 15$ . For other examples of mathematical work on densely packing relatively small numbers of identical disks, see [5, 6, 9, 16], and [10, 15, 20] for related experimental work. Many authors have considered heuristics for circle packing problems, see [11, 23] for overviews of numerous heuristics and optimization methods. The best known solutions for packing equal disks into squares, triangles and other shapes are continuously published on Specht's website <http://packomania.com> [21].

The related problem of packing square objects has also been studied for a long time. The decision problem whether it is possible to pack a given set of squares into the unit square was shown to be strongly NP-complete by Leung et al. [14], using a reduction from 3-PARTITION. Already in 1967, Moon and Moser [17] found a sufficient condition. They proved that it is possible to pack a set of squares into the unit square in a shelf-like manner if their combined area (i.e., the sum over all square areas) does not exceed  $\frac{1}{2}$ . At the same time,  $\frac{1}{2}$  is the *largest upper area bound* one can hope for, because two squares larger than the quarter-squares shown in Fig. 1 cannot be packed. We call the ratio between the largest combined object area that can always be packed and the area of the container the problem's *critical packing density*.

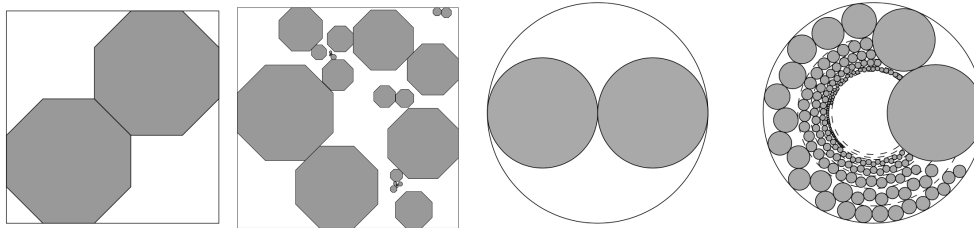


■ **Figure 1** (1) An instance of critical packing density for squares in a square. (2) An example packing produced by Moon and Moser's Shelf Packing. (3) An instance of critical packing density for disks in a square. (4) An example packing produced by Morr's Split Packing.

The equivalent problem of establishing the critical packing density for disks in a square was posed by Demaine, Fekete, and Lang [2] and resolved relatively recently by Morr, Fekete and Scheffer [4, 18]. Making use of Split Packing, a recursive procedure for cutting the



■ **Figure 2** An example run of Split Packing.



■ **Figure 3** (1) An instance of critical packing density for regular octagons in a square. (2) An example packing produced by Split Packing. (3) An instance of critical packing density for circles in a circles. (4) An example packing produced by Boundary/Ring Packing of Fekete, Keldenich, and Scheffer.

container into *hats*, i.e., triangular pieces with rounded corners, they proved that the critical packing density of disks in a square is  $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$ . As illustrated in Fig. 3 (1) and (2), Split Packing can be generalized to also achieve critical packing density for other shapes such as regular octagons.

The analogous question of establishing the critical packing density for disks in a disk has just been resolved by Fekete, Keldenich and Scheffer [3]. As illustrated in Fig. 3 (3) and (4), it is 0.5.

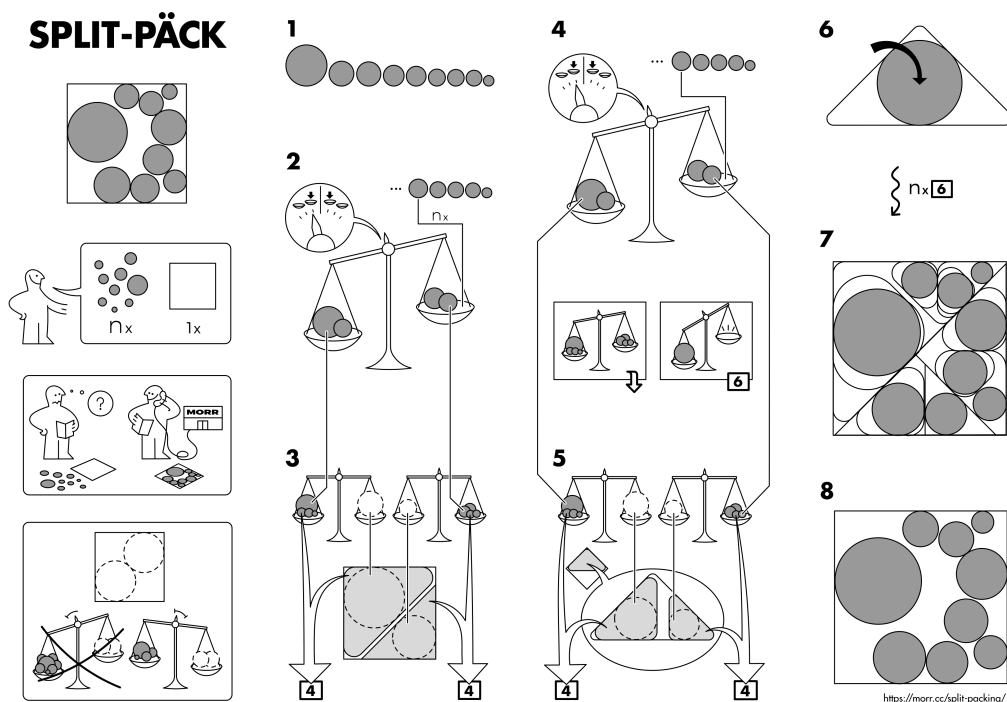
## 2 Squares in a Square: Shelf Packing

The basic idea for Shelf Packing (see Fig. 1 (2)) is to greedily insert the items in order of decreasing size, starting in the lower left corner. Items are placed from left to right with aligned bottoms, until an item no longer fits to the right of its predecessor; when that happens, an item is placed on top of the leftmost item in the previous row. Assuming that this fails to place the last item completely inside the container yields an estimate for the packed area that exceeds  $1/2$ ; this involves relatively simple algebraic transformations, as shown in the video.

## 3 Circles in a Square: Split Packing

The key idea for Split Packing is to recursively subdivide both the set of items and the container in a fashion that yields a similar, balanced split for both. A schematic overview is shown in Fig. 4: After sorting the circles by decreasing size (Step 1), the set of items is greedily split into two subsets, such that the difference between their total area does not exceed the area of the last subset (Step 2); the area split is also applied to the container along a diagonal cut (Step 3). A finer point is to notice that a guaranteed minimum size of circles in a subset makes it sufficient to consider triangular subcontainers with rounded corners (“hats”), as the acute corners will not be required to accommodate bigger circles; as a consequence, asymmetric diagonal cuts leave a bigger piece that can still be treated as such a hat, allowing recursion. This is carried out recursively on both subsets (Steps 4 and 5),

until a subset consists of a single circle, which is then simply packed into its corresponding triangle (Step 6). The resulting subdivision and packing are shown in Steps 7 and 8. In the end, each subcontainer achieves a packing density of at least  $\frac{\pi}{3+2\sqrt{2}} \approx 0.539$ , establishing the critical bound; see Fig. 2 for an example run.



■ **Figure 4** A schematic illustration of Split Packing.

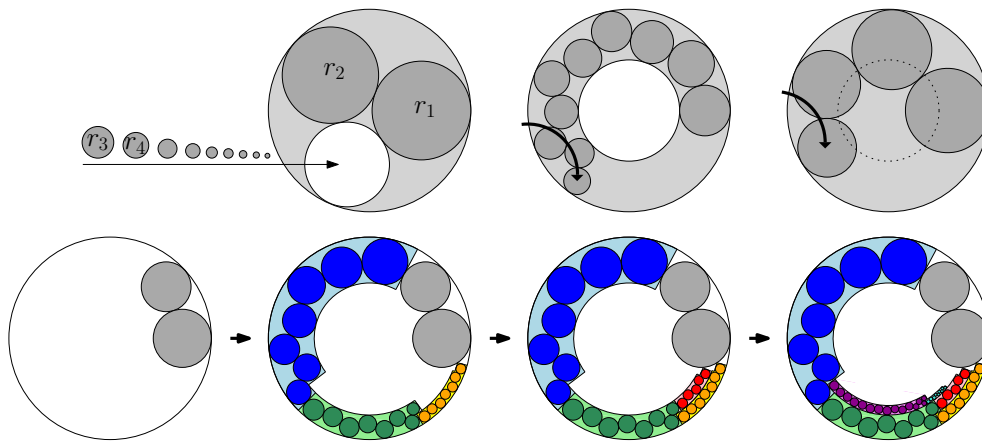
The key insight (a balanced, recursive split into subsets can be used analogously to the container by a diagonal cut) can also be applied to other kinds of objects: See Fig. 3 for the case of regular octagons, for which the critical packing density is  $8(5\sqrt{2} - 7) \approx 0.5685\dots$ . Similar results can be obtained for other shapes and triangular containers, see the paper [4] for details.

#### 4 Circles in a Circle: Boundary/Ring Packing

For the scenario of packing circles into a circle, the curved container boundary requires a different approach. The algorithm of Fekete, Keldenich and Scheffer [3] applies subroutines. The circles we want to pack into a circle container  $\mathcal{C}$  are considered in decreasing order of radius. In the following, we sketch the algorithm of Fekete, Keldenich and Scheffer; see the full paper [3] for details.

First we check whether the two largest circles allow a recursive packing of the remaining circles, see Figure 5 (top-left). If we cannot apply the recursion, two subroutines are used: *Ring Packing* and *Boundary Packing*, see Figures 5 (top-middle) and (top-right).

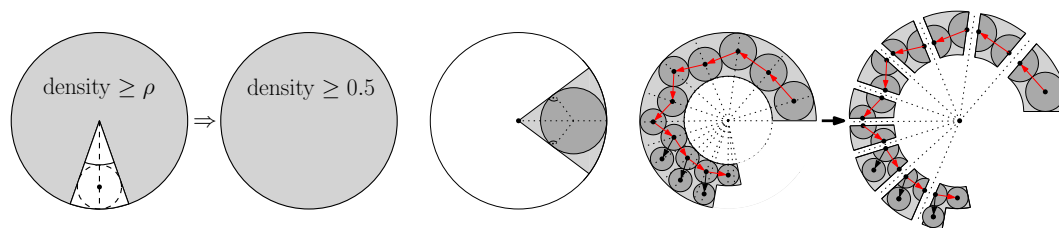
*Ring Packing* packs circles into a ring  $R$ , alternating between touching the inner and outer boundary of the ring. If the current circle and the previously packed circle could pass each other in  $R$ , Ring Packing partitions the ring into two rings and continues by packing into these two rings. Ring Packing continues until the current circle cannot be packed into any open ring, see Figure 5 (bottom).



■ **Figure 5** The approach of Fekete, Keldenich and Scheffer [3]: (Top, left to right) Recursion, Ring Packing and Boundary Packing. (Bottom) A stepwise run of the algorithm.

*Boundary Packing* packs the input circles into the container, touching its boundary and the previously packed circle until the radius of the current circle is below a given threshold.

Using a combination of manual and automated case checking, the analysis ensures a packing density of at least 0.5; see Figure 6 for an illustration and the full paper for details.



■ **Figure 6** The analytic approach for ensuring a packing density of at least 0.5.

## 5 The Video

The video opens with a number of practical illustrations for packing problems, followed by a sketch of NP-hardness proofs and additional geometric difficulties when packing circles. After introducing the concept of critical packing density, we sketch the algorithmic proof by Moon and Moser for the critical packing density of squares in a square. This is followed by an illustrated description of Morr's Split Packing algorithm for circles in a square, along with extensions to other shapes and containers. Finally, we sketch a current result on the critical packing density of circles in a circle.

---

## References

- 1 Ignacio Castillo, Frank J. Kampas, and János D. Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. *European Journal of Operational Research*, 191(3):786–802, 2008.
- 2 Erik D. Demaine, Sándor P. Fekete, and Robert J. Lang. Circle Packing for Origami Design is Hard. In *Origami<sup>5</sup>: 5th International Conference on Origami in Science, Mathematics and Education*, AK Peters/CRC Press, pages 609–626, 2011. [arXiv:1105.0791](https://arxiv.org/abs/1105.0791).

- 3 Sándor P. Fekete, Phillip Keldenich, and Christian Scheffer. Packing Disks into Disks with Optimal Worst-Case Density. In *Proceedings of the 35th Annual Symposium on Computational Geometry (SoCG)*, LIPIcs, pages 35:1–35:19, 2019. These proceedings; full version at <https://arxiv.org/abs/1903.07908>.
- 4 Sándor P. Fekete, Sebastian Morr, and Christian Scheffer. Split Packing: Algorithms for Packing Circles with Optimal Worst-Case Density. *Discrete & Computational Geometry*, page 562–594, 2018. doi:10.1007/s00454-018-0020-2.
- 5 F. Fodor. The Densest Packing of 19 Congruent Circles in a Circle. *Geometriae Dedicata*, 74:139–145, 1999.
- 6 F. Fodor. The Densest Packing of 12 Congruent Circles in a Circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 41:401–409, 2000.
- 7 F. Fodor. The Densest Packing of 13 Congruent Circles in a Circle. *Beiträge zur Algebra und Geometrie (Contributions to Algebra and Geometry)*, 44:431–440, 2003.
- 8 Hamish J. Fraser and John A. George. Integrated container loading software for pulp and paper industry. *European Journal of Operational Research*, 77(3):466–474, 1994.
- 9 M. Goldberg. Packing of 14, 16, 17 and 20 circles in a circle. *Mathematics Magazine*, 44:134–139, 1971.
- 10 R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, and P.R.J. Östergård. Dense Packings of Congruent Circles in a Circle. *Discrete Mathematics*, 181:139–154, 1998.
- 11 Mhand Hifi and Rym M’Hallah. A literature review on circle and sphere packing problems: Models and methodologies. *Advances in Operations Research*, 2009. Article ID 150624.
- 12 S. Kravitz. Packing cylinders into cylindrical containers. *Mathematics Magazine*, 40:65–71, 1967.
- 13 Robert J. Lang. A computational algorithm for origami design. *Proceedings of the Twelfth Annual Symposium on Computational Geometry (SoCG)*, pages 98–105, 1996.
- 14 Joseph Y. T. Leung, Tommy W. Tam, Chin S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.
- 15 B.D. Lubachevsky and R.L. Graham. Curved Hexagonal Packings of Equal Disks in a Circle. *Discrete & Computational Geometry*, 18:179–194, 1997.
- 16 H. Melissen. Densest Packing of Eleven Congruent Circles in a Circle. *Geometriae Dedicata*, 50:15–25, 1994.
- 17 John W. Moon and Leo Moser. Some packing and covering theorems. In *Colloquium Mathematicae*, volume 17, pages 103–110. Institute of Mathematics, Polish Academy of Sciences, 1967.
- 18 Sebastian Morr. Split Packing: An Algorithm for Packing Circles with Optimal Worst-Case Density. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 99–109, 2017.
- 19 Norman Oler. A finite packing problem. *Canadian Mathematical Bulletin*, 4:153–155, 1961.
- 20 G.E. Reis. Dense Packing of Equal Circles within a Circle. *Mathematics Magazine*, issue 48:33–37, 1975.
- 21 Eckard Specht. Packomania, 2015. URL: <http://www.packomania.com/>.
- 22 Kokichi Sugihara, Masayoshi Sawai, Hiroaki Sano, Deok-Soo Kim, and Donguk Kim. Disk packing for the estimation of the size of a wire bundle. *Japan Journal of Industrial and Applied Mathematics*, 21(3):259–278, 2004.
- 23 Péter Gábor Szabó, Mihály Csaba Markót, Tibor Csendes, Eckard Specht, Leocadio G. Casado, and Inmaculada García. *New Approaches to Circle Packing in a Square*. Springer US, 2007.
- 24 Huaiqing Wang, Wenqi Huang, Quan Zhang, and Dongming Xu. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141(2):440–453, September 2002.

# Properties of Minimal-Perimeter Polyominoes

**Gill Barequet**

Technion – Israel Inst. of Technology, Haifa, Israel  
barequet@cs.technion.ac.il

**Gil Ben-Shachar**

Technion – Israel Inst. of Technology, Haifa, Israel  
gilbe@cs.technion.ac.il

---

## Abstract

In this video, we survey some results concerning polyominoes, which are sets of connected cells on the square lattice, and specifically, minimal-perimeter polyominoes, that are polyominoes with the minimal-perimeter from all polyominoes of the same size.

**2012 ACM Subject Classification** Mathematics of computing → Combinatoric problems

**Keywords and phrases** Polyominoes, Perimeter, Minimal-Perimeter

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.64

**Category** Multimedia Exposition

**Supplement Material** <https://www.youtube.com/watch?v=hmMewVI1RIg&rel=0>

**Funding** Work on this paper by both authors has been supported in part by ISF Grant 575/15 and by BSF Grant 2017684.

## 1 Introduction

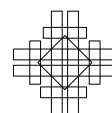
In this video, we discuss the combinatorial and geometric properties of minimal-perimeter polyominoes. A polyomino is a set of edge-connected squares on the square lattice. The size of a polyomino is the number of squares composing it. The problem of counting polyominoes dates back to the 1950s, when it was studied in parallel in the fields of combinatorics [7] and statistical physics [6]. The number of polyominoes of area  $n$  is denoted in the literature by  $A(n)$ . No formula for  $A(n)$  is known as of today, but there exist a few algorithms for computing values of  $A(n)$ , such as those of Redelmeier [12] and Jensen [8]. Using Jensen's algorithm, values of  $A(n)$  were computed up to  $n = 56$ . The existence of the *growth constant* of  $A(n)$ , namely,  $\lambda := \lim_{n \rightarrow \infty} \sqrt[n]{A(n)}$ , was shown by Klarner [9]. More than 30 years later, Madras [11] showed that  $\lim_{n \rightarrow \infty} A(n+1)/A(n)$  exists and, thus, is equal to  $\lambda$ . The best known lower and upper bounds on  $\lambda$  are 4.0025 [5] and 4.6496 [10], respectively. The currently best *estimate* of  $\lambda$  is  $4.0625696 \pm 0.0000005$  [8].

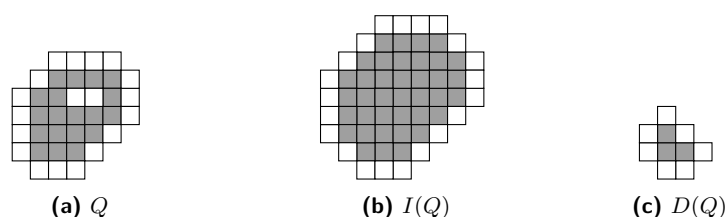
The perimeter of a polyomino is the set of empty squares adjacent to the polyomino. The number of polyominoes with a given size and perimeter size was studied in several papers. Recently, Asinowski et al. [2] provided formulae for the number of polyominoes of a given area and perimeter that is close to the maximum possible perimeter.

In contrast to polyominoes with large perimeter size, a minimal-perimeter polyomino is a polyomino with the minimum possible perimeter for its size. Minimal-perimeter polyominoes were studied by Altshular et al. [1] and by Sieben [13], providing formulae for the minimum perimeter size possible for a given size of a polyomino. Recently, we provided some combinatorial results about the sets of minimal-perimeter polyominoes [3, 4]. In this video, we show some of these results.



© Gill Barequet and Gil Ben-Shachar;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 64; pp. 64:1–64:4  
Leibniz International Proceedings in Informatics  
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** A polyomino  $Q$  and the respective inflated and deflated polyominoes. The gray cells represent polyomino cells, while the white cells represent perimeter cells.

## 2 Results

As mentioned in the introduction, a polyomino, denoted by  $Q$ , is a set of edge-connected cells. The perimeter of a polyomino, denoted by  $\mathcal{P}(Q)$ , is the set of empty cells adjacent to the polyomino.

In this video, we present several results about the properties of minimal-perimeter polyominoes, namely, polyominoes with the minimum number of perimeter cells for their size. A key concept for these results is the inflation operation. Formally, the inflation of a polyomino  $Q$ , denoted by  $I(Q)$ , is defined as  $I(Q) = Q \cup \mathcal{P}(Q)$ , or in words, inflating a polyomino is the operation of expanding a polyomino by its perimeter. Similarly, the deflation of a polyomino  $Q$ , denoted by  $D(Q)$ , is the removal of the cells adjacent to some perimeter cells. Formally, we define the border of  $Q$ , denoted by  $\mathcal{B}(Q)$  as the set of cells adjacent to a perimeter cell. Similarly to inflation, the deflated polyomino is defined as  $D(Q) = Q \setminus \mathcal{B}(Q)$ . Those concepts are depicted in Figure 1.

### 2.1 Inflating Minimal-Perimeter Polyominoes

Using the above concepts, we present in the video the following theorems [3].

► **Theorem 1.** [3, Thm. 3] *If  $Q$  is a minimal-perimeter polyomino, then  $I(Q)$  is a minimal-perimeter polyomino as well.*

► **Theorem 2.** [3] *Let  $Q$  be a polyomino for which  $D(Q)$  is a minimal-perimeter polyomino. Then, for any other minimal-perimeter polyomino  $Q'$  of the same size as  $Q$ , we have that  $D(Q')$  is a minimal-perimeter polyomino as well.*

Combining these results, we obtain the following theorem.

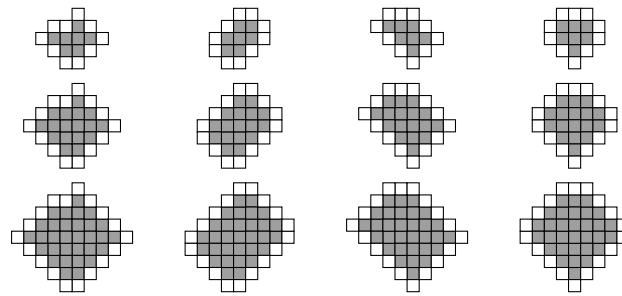
► **Theorem 3.** [3, Thm. 4] *Let  $M_n$  be the set of minimal-perimeter polyominoes of size  $n$ . Then, for  $n \geq 3$ , we have that  $|M_n| = |M_{n+\epsilon(n)}|$ , where  $\epsilon(n)$  is the perimeter size of a minimal-perimeter polyomino of size  $n$ .*

Theorem 3 is demonstrated in Figure 2. Theorem 3 is the main result shown in [3]; we refer the reader to the full paper for more details.

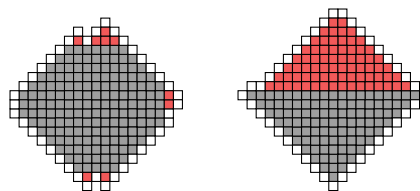
### 2.2 Counting Minimal-Perimeter Polyominoes

The bijection shown in Theorem 3 provides the number of minimal-perimeter polyominoes for any size which is obtained by inflating a set of minimal-perimeter polyominoes of some smaller size  $n$ , for which we know the cardinality of  $M_n$ . However, there are sizes of polyominoes, for which the set of minimal-perimeter polyominoes is not created by inflating the polyominoes





■ **Figure 2** A demonstration of Theorem 3. The first row contains all the minimal-perimeter polyominoes of size 7, the second row contains all the minimal-perimeter polyominoes of size 17, and the last row contains all the minimal-perimeter polyominoes of size 31. Notice that the polyominoes in each row are the inflated versions of the respective polyominoes in the previous row.



■ **Figure 3** An example of the decomposition of two minimal-perimeter polyominoes into their the body (gray) and caps (red).

in another set of minimal-perimeter polyominoes. The properties of those sizes are discussed in [4]. However, due to time constraints, we focus in the video only on the question of how many minimal-perimeter polyominoes there are for a given size.

In order to count the minimal-perimeter polyominoes of a given size, we show that any minimal-perimeter polyomino can be decomposed into an octagonal shaped polyomino, with the same perimeter as the original polyomino (referred to as the “body” of the polyomino), and “caps,” which are added on top of the octagon edges and do not change the perimeter. This decomposition is depicted in Figure 3. Using this decomposition, we were able to devise an efficient algorithm for counting minimal-perimeter polyominoes of a given size. Data we obtained are shown in Figure 4. For more information, we refer the reader to [4].

### 3 Future Work

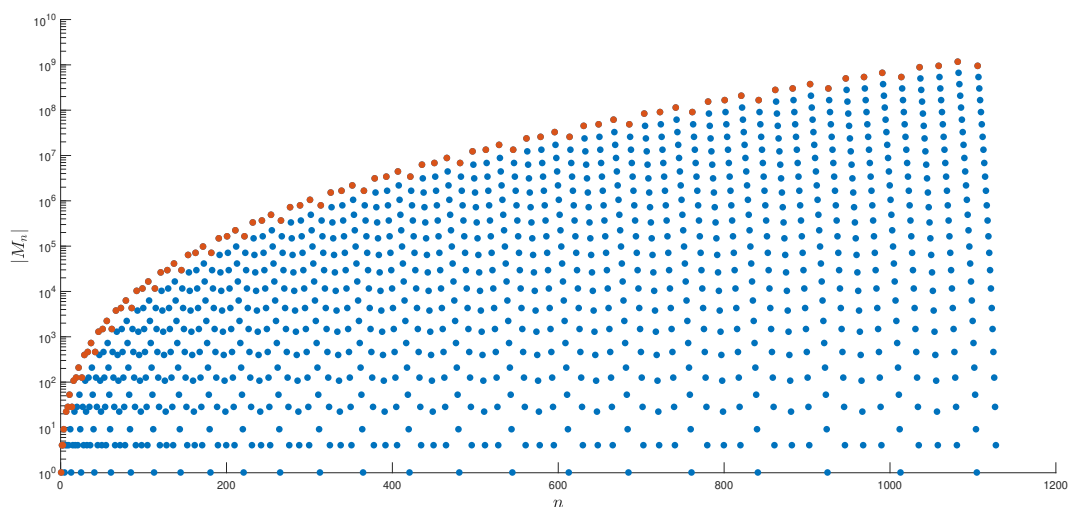
Several questions in this area are yet to be answered. First, how do those results generalize to other lattices, such as polyiamonds on the triangular lattice, polyhexes on the hexagonal lattice, and polycubes on the cubical lattice in higher dimensions?

Second, the results shown in Figure 4 raise some questions about the behavior of the upper envelope of the sequence  $|M_n|$ . It seems that this sequence is exponential, and if it is, we would like to achieve good bounds on its growth rate.

### 4 The Video

The video depicts the above results. It was created mostly using Powtoon. The 3-dimensional animation was created using Maya Autodesk, and the graph animation was created programmatically using the D3.js library.





■ **Figure 4** Values of  $|M_n|$ . The values which are not created by the inflation operation are shown in red.

---

## References

- 1 Y. Altshuler, V. Yanovsky, D. Vainsencher, I.A. Wagner, and A.M. Bruckstein. On minimal perimeter polyominoes. In *Proc. Conf. Discrete Geometry for Computer Imagery*, pages 17–28. Springer, 2006.
- 2 A. Asinowski, G. Barequet, and Y. Zheng. Enumerating polyominoes with fixed perimeter defect. In *Proc. 9th European Conf. on Combinatorics, Graph Theory, and Applications*, volume 61, pages 61–67, Vienna, Austria, August 2017. Elsevier.
- 3 G. Barequet and G. Ben-Shachar. Properties of Minimal-Perimeter polyominoes. In *International Computing and Combinatorics Conference*, Qingdao, China, 2018. Springer.
- 4 G. Barequet and G. Ben-Shachar. Minimal-Perimeter Polyominoes: Chains, Roots, and Algorithms. In *Conference on Algorithms and Discrete Applied Mathematics*, pages 109–123. Springer, 2019.
- 5 G. Barequet, G. Rote, and M. Shalah.  $\lambda > 4$ : An improved lower bound on the growth constant of polyominoes. *Comm. of the ACM*, 59(7):88–95, 2016.
- 6 S.R. Broadbent and J.M. Hammersley. Percolation processes: I. Crystals and Mazes. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 53, pages 629–641. Cambridge University Press, 1957.
- 7 S.W. Golomb. Checker boards and polyominoes. *The American Mathematical Monthly*, 61(10):675–682, 1954.
- 8 I. Jensen. Counting Polyominoes: A Parallel Implementation for Cluster Computing. In Peter M. A. Sloot, David Abramson, Alexander V. Bogdanov, Yuriy E. Gorbachev, Jack J. Dongarra, and Albert Y. Zomaya, editors, *Proc. Int. Conf. on Computational Science*, pages 203–212, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- 9 D.A. Klarner. Cell growth problems. *Canadian J. of Mathematics*, 19:851–863, 1967.
- 10 D.A. Klarner and R.L. Rivest. A procedure for improving the upper bound for the number of n-ominoes. *Canadian J. of Mathematics*, 25(3):585–602, 1973.
- 11 N. Madras. A pattern theorem for lattice clusters. *Annals of Combinatorics*, 3(2):357–384, June 1999.
- 12 D.H. Redelmeier. Counting polyominoes: Yet another attack. *Discrete Mathematics*, 36(2):191–203, 1981.
- 13 N. Sieben. Polyominoes with minimum site-perimeter and full set achievement games. *European J. of Combinatorics*, 29(1):108–117, 2008.

# A Manual Comparison of Convex Hull Algorithms

Maarten Löffler

Dept. of Information and Computing Sciences, Utrecht University, The Netherlands  
m.loffler@uu.nl

---

## Abstract

We have verified experimentally that there is at least one point set on which Andrew’s algorithm (based on Graham’s scan) to compute the convex hull of a set of points in the plane is significantly faster than a brute-force approach, thus supporting existing theoretical analysis with practical evidence. Specifically, we determined that executing Andrew’s algorithm on the point set

$$P = \{(1, 4), (2, 8), (3, 10), (4, 1), (5, 7), (6, 3), (7, 9), (8, 5), (9, 2), (10, 6)\}$$

takes 41 minutes and 18 seconds; the brute-force approach takes 3 hours, 49 minutes, and 5 seconds.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** convex hull, efficiency

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.65

**Category** Multimedia Exposition

**Funding** *Maarten Löffler*: Partially supported by the Netherlands Organisation for Scientific Research (NWO); 614.001.504.

## 1 Introduction

Computational efficiency lies at the heart of the research area of computational geometry, and more broadly at that of algorithm design. As we explain in every single paper, it is of vital importance, since data sets are growing ever larger, and advancing hardware cannot compete with the pitiless mathematics of asymptotic behaviour. The same growth in both data size and computation speed is also making the notion of efficiency more abstract, and more difficult to explain to inhabitants of a world in which instant computation is the norm.

In an attempt to make the concept of efficiency more tangible by scaling both the size of the data set and the speed of the computation to something more human, we report on a manual execution of two different algorithms to compute the convex hull of the set

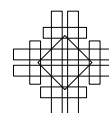
$$P = \{(1, 4), (2, 8), (3, 10), (4, 1), (5, 7), (6, 3), (7, 9), (8, 5), (9, 2), (10, 6)\} \quad (\text{Figure 1})$$

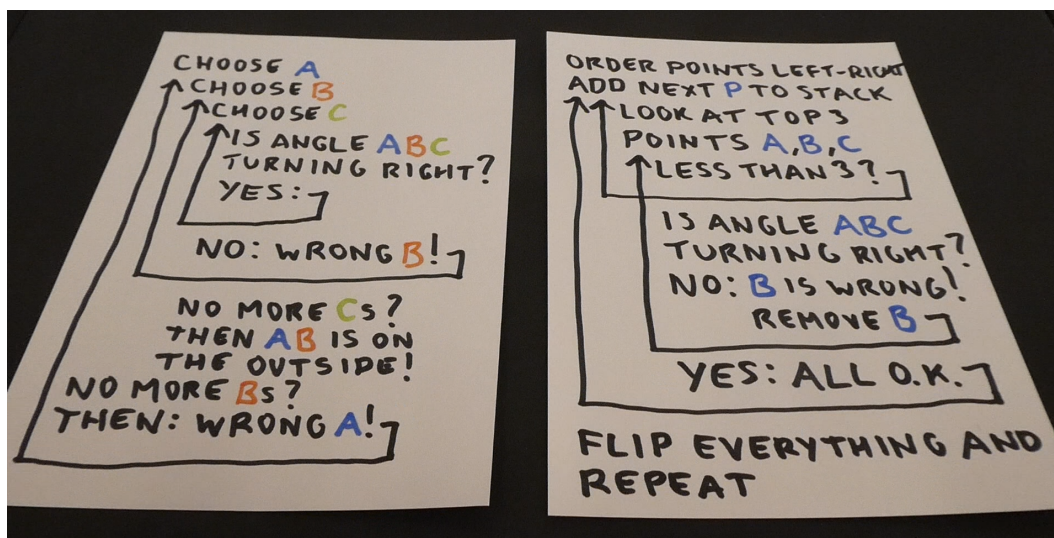
of ten points in  $\mathbb{R}^2$ . The execution has been recorded and timed.



■ **Figure 1** Left: the set  $P$  of 10 points in the plane. Right: the convex hull of  $P$ .

The experiment is inspired by the introductory chapter of the well-known computational geometry textbook of de Berg *et al.* [2], which uses the convex hull problem to illustrate the value of efficient algorithms by comparing a brute-force algorithm which runs in  $O(n^3)$  time to the  $O(n \log n)$  time algorithm by Andrew [1], which is in turn based on Graham’s scan algorithm [3].





■ **Figure 2** Pseudocode of the two algorithms. Left: a cubic-time brute-force algorithm. Right: Andrew's adaptation of Graham's scan algorithm.

## 2 Implementation

We chose two different algorithms to test: a fairly straightforward cubic-time algorithm, and the algorithm of Andrew [1]. We made some minor adaptations out of practical considerations. Pseudocode of the two algorithms is provided in Figure 2.

We implemented the algorithms using light cardboard points and semi-transparent tape for the edges (or potential edges) of the hull. Points are stored in standard postal envelopes, and geometric predicates are tested by (temporarily) embedding the points physically on a 30-centimeter unit grid and sticking tape edges onto the grid. As coordinates range from 1 to 10, this causes some fluctuation in the time required to perform an operation, depending on the walking distance to the respective point.

We expect there is additional bias in the results, due to several environmental factors. In particular, over the course of the execution of the brute-force algorithm, there is a measurable increase in the number of operations per minute.

## 3 Results

The execution of the brute-force algorithm took 3 hours, 49 minutes, and 5 seconds. The execution of Andrew's algorithm took 41 minutes and 18 seconds.

---

### References

- 1 A.M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. doi:10.1016/0020-0190(79)90072-3.
- 2 Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 3rd edition, 2008.
- 3 R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972. doi:10.1016/0020-0190(72)90045-2.

# Fréchet View – A Tool for Exploring Fréchet Distance Algorithms

Peter Schäfer

FernUniversität in Hagen, Germany  
peter.schaefer1@studium.fernuni-hagen.de

---

## Abstract

The **Fréchet-distance** is a similarity measure for geometric shapes. Alt and Godau presented the first algorithm for computing the Fréchet-distance and introduced a key concept, the **free-space diagram**. Since then, numerous variants of the Fréchet-distance have been studied.

We present here an interactive, graphical tool for exploring some Fréchet-distance algorithms. Given two curves, users can experiment with the free-space diagram and compute the Fréchet-distance. The Fréchet-distance can be computed for two important classes of shapes: for polygonal curves in the plane, and for simple polygonal surfaces.

Finally, we demonstrate an implementation of a very recent concept, the ***k*-Fréchet-distance**.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Fréchet distance, free-space diagram, polygonal curves, simple polygons

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2019.66

**Category** Multimedia Exposition

**Supplement Material** <https://hrimfaxi.bitbucket.io/fv>

- Binaries (for Windows, Linux, macOS), sample data and source code
- Brief user guide
- Video presentation

**Acknowledgements** This program was developed as part of my master thesis at the chair of Prof. Dr. André Schulz, supervised by Dr. Lena Schlipf.

## 1 The Fréchet-distance

The Fréchet-distance is a metric for describing similarity between two continuous shapes. The need for comparing shapes arises in many fields of applications, like computer vision, handwriting recognition, proteome matching, map construction, analysis of movement data, and more. The Fréchet-distance is similar to the Hausdorff-metric, but it better captures the notion of similarity. It is defined as the minimum bottleneck-cost over all possible homeomorphisms.

► **Definition 1.** Let  $P$  and  $Q$  be two given curves in a metric space  $S$ . A reparametrization  $\alpha$  of  $[0, 1]$  is a continuous, non-decreasing surjection  $\alpha : [0, 1] \mapsto [0, 1]$ .

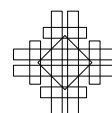
The **Fréchet-distance** between  $P$  and  $Q$  is the infimum over all reparametrizations  $\alpha$  and  $\beta$  of the maximum over all  $t \in [0, 1]$  of the distance between  $P(\alpha(t))$  and  $Q(\beta(t))$ :

$$\delta_F(P, Q) = \inf_{\alpha, \beta} \max_{t \in [0, 1]} \|P(\alpha(t)) - Q(\beta(t))\|,$$

where  $\|\cdot\|$  denotes the distance function of  $S$ . In the following sections we will always assume the Euclidean norm in  $\mathbb{R}^2$ , and we will only consider polygonal curves and shapes.

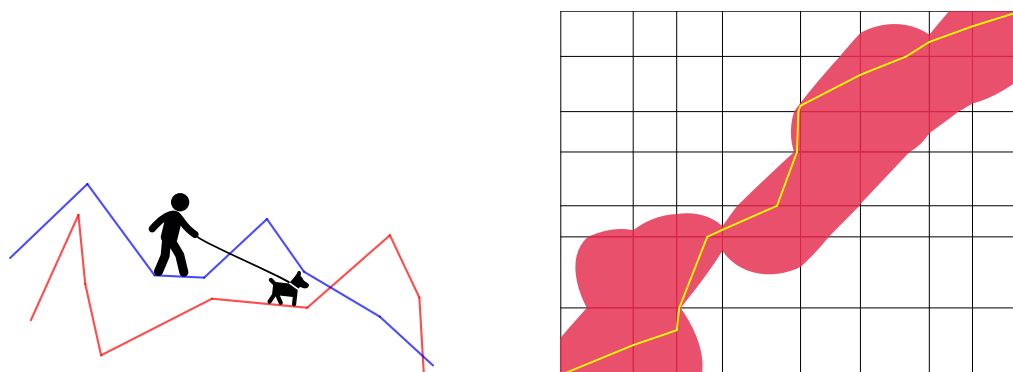


© Peter Schäfer;  
licensed under Creative Commons License CC-BY  
35th International Symposium on Computational Geometry (SoCG 2019).  
Editors: Gill Barequet and Yusu Wang; Article No. 66; pp. 66:1–66:5  
Leibniz International Proceedings in Informatics  
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



An intuitive explanation of the Fréchet-distance is this: imagine a man walking his dog on a leash. The man walks on one curve, while the dog follows the other curve. Both the man and the dog must move forward. The *weak* Fréchet-distance would allow for backward movement, also. We will come back to it in section 3.

- the *decision problem* asks: is it possible to walk the curves with a leash of given length  $\varepsilon$ ?
- the *optimization problem* asks for the shortest possible leash.



■ **Figure 1** Two curves, and a dog on a leash. Right: the corresponding free-space diagram with a feasible path.

Alt and Godau [2] presented an algorithm for computing the Fréchet-distance for polygonal curves in the plane. The key concept of this algorithm is the **free-space diagram**: one curve is mapped to the  $x$ -axis of the diagram, the second curve is mapped to the  $y$ -axis, spanning a two-dimensional grid, or configuration space.

The **free-space** represents all pairs of points whose distance is less than or equal to  $\varepsilon$ . In Figure 1 the free-space is indicated by the shaded area (which is composed of numerous ellipse sections).

We have  $d_F(P, Q) \leq \varepsilon$ , if there is a monotone path in the free-space diagram starting at the lower left corner  $(0, 0)$  and ending in the upper right corner  $(p, q)$  (where  $p$  and  $q$  denote the number of vertices in the curves). We call that a **feasible path**.

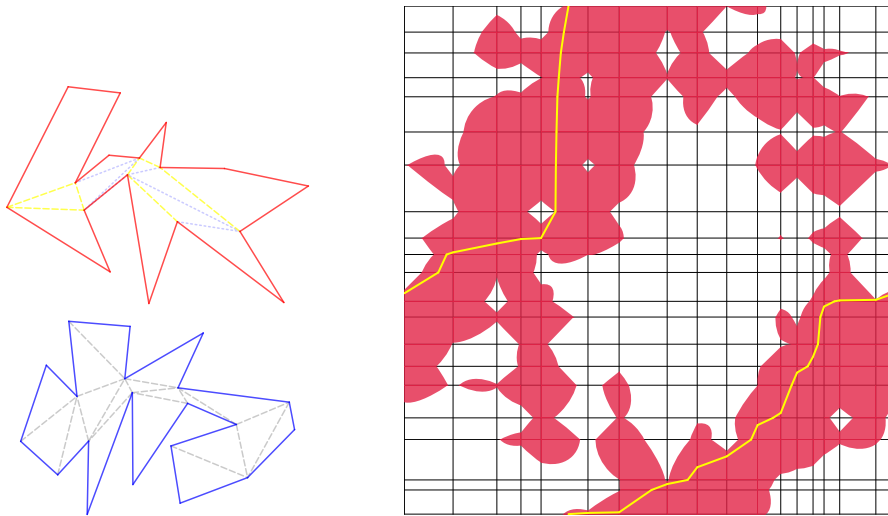
For *closed* curves, the left and right edges of the free-space diagram coincide. Just imagine the diagram to wrap around at the edges, creating a cylinder (or a torus, though the latter is not relevant for our algorithms).<sup>1</sup> The starting point of a feasible path may be located anywhere on the  $x$ -axis. Alt and Godau's [2] algorithm finds such a path in  $O(pq \log pq)$  time. We can thus solve the decision problem.

The optimization problem is solved by applying the algorithm repeatedly on a set of critical values, i. e. values of  $\varepsilon$  for which the structure of the free-space diagram changes significantly. **Fréchet View** is a useful tool for examining the structure of free-space diagrams.

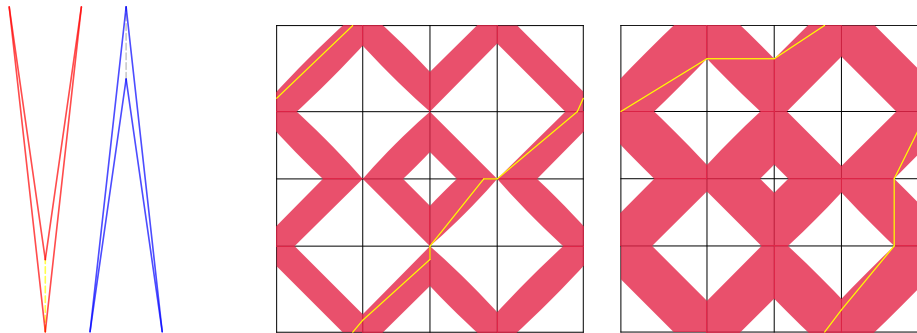
## 2 The Fréchet-distance for Simple Polygons

The Fréchet-distance easily extends to higher dimensions. As it turns out, computation becomes significantly harder, already for two-dimensional surfaces. Buchin et al. [3] presented a polynomial-time algorithm for a natural subset of surfaces, namely **simple polygons** (i. e.

<sup>1</sup> In the literature this is often referred to as a *double-free-space diagram*



■ **Figure 2** Triangulated polygons and their free-space diagram.



■ **Figure 3** The Fréchet-distance of the polygonal surfaces differs from the Fréchet-distance of their boundary curves.

polygons without holes and self-intersection). Basically, their algorithm is able to break down the problem to a *convex decomposition* and *triangulation* of the polygons (see Figure 2). It is sufficient to consider certain diagonals, which are mapped to *shortest paths* in the other polygon, with additional conditions applying. To the best of our knowledge, ours is the first practical implementation of this algorithm.

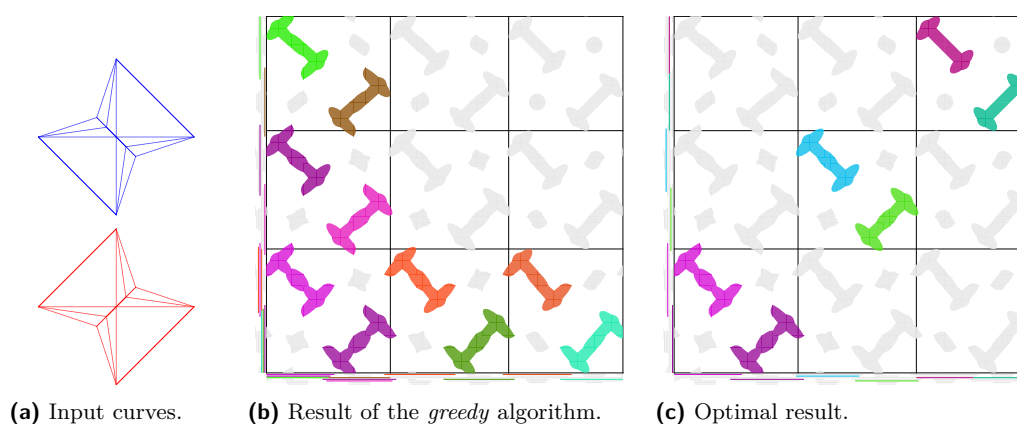
In many cases, the Fréchet-distance for simple polygons is just identical to the Fréchet-distance of their boundary curves. Figure 3 shows an example where they actually differ.

### 3 The $k$ -Fréchet-distance

Buchin and Ryvkin [4, 1] recently introduced a new concept, the  $k$ -Fréchet-distance. It represents a connecting bridge between the Hausdorff-metric and the well-known *weak* Fréchet-distance. Roughly speaking, the input curves are cut into parts and each part is mapped using the weak Fréchet-distance. With the weak Fréchet-distance, reparametrizations are not required to be monotone; man and dog are allowed to move backwards.

The question is: can we find a mapping with at most  $k$  parts? This adds a strong combinatorial aspect to the problem. In fact, the problem was shown to be NP-complete by Buchin and Ryvkin [4, 1].

We present implementations of two algorithms: an exponential-time “*brute-force*” exact algorithm, and a *greedy* approximation algorithm.



■ **Figure 4**  $k$ -Fréchet-distance: greedy vs. optimal results. The approximation factor can be shown to approach 2.

Many questions regarding the  $k$ -Fréchet-distance are still open to research. For one, the exact approximation factor of the greedy algorithm was not yet known. With the help of Fréchet View we were able to construct curves whose approximation ratio comes arbitrarily close to 2 (see Figure 4).

#### 4 Fréchet View – the Implementation

**Fréchet View** is a tool for exploring various aspects of the Fréchet-distance algorithms. It expects two polygonal curves as input data. These curves are stored in vector graphics files (in SVG, or IPE format) and can be edited with any usual vector graphics editor, like Inkscape, or IPE. In addition, we provide an implementation to assemble curves from straightforward JavaScript code.

Both curves and the corresponding free-space diagram are displayed on screen. The user can manipulate the parameter  $\varepsilon$  and watch the changes to the free-space diagram. Whenever a feasible path is found, it is drawn within the free-space diagram. Moving the mouse pointer over the path displays the mapped sections on the input curves (and vice versa).

Graphics produced by Fréchet View can be stored to disk. For example, all figures in this document have been created using Fréchet View.

The program is written in C++, making use of the Computational Geometry Algorithms Library [5], and a number of development frameworks like Qt, Boost, TBB, and more. It can be run from the command line with additional options for multi-core and GPGPU-support. However, these features are not part of our presentation.

---

#### References

- 1 Hugo A. Akitaya, Maike Buchin, Leonie Ryvkin, and Jérôme Urhausen. The  $k$ -Fréchet distance revisited and extended. In *35th European Workshop on Computational Geometry*, 2019. URL: <http://www.eurocg2019.uu.nl/papers/41.pdf>.
- 2 Helmut Alt and Michael Godau. Computing the Fréchet Distance between two Polygonal Curves. *International Journal of Computational Geometry and Applications*, 5(1-2):75–91, 1995. doi:10.1142/S0218195995000064.



- 3 Kevin Buchin, Maïke Buchin, and Carola Wenk. Computing the Fréchet Distance Between Simple Polygons in Polynomial Time. In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry, SCG '06*, pages 80–87, 2006. doi:10.1145/1137856.1137870.
- 4 Maïke Buchin and Leonie Ryvkin. The  $k$ -Fréchet distance of polygonal curves. In *34th European Workshop on Computational Geometry*, 2018. URL: [https://conference.imp.fu-berlin.de/eurocg18/download/paper\\_43.pdf](https://conference.imp.fu-berlin.de/eurocg18/download/paper_43.pdf).
- 5 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.12 edition, 2018. URL: <https://doc.cgal.org/4.12/Manual/packages.html>.



