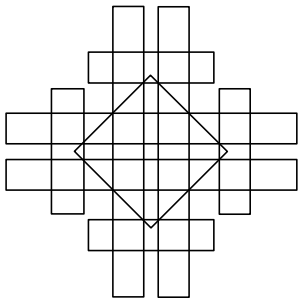
32nd International Symposium on Computational Geometry

SoCG'16, June 14–17, 2016, Boston, MA, USA

Edited by

Sándor Fekete

Anna Lubiw



LIPICS

Editors

Sándor Fekete	Anna Lubiw
TU Braunschweig	University of Waterloo
Braunschweig	Waterloo
Germany	Canada
s.fekete@tu-bs.de	alubiw@uwaterloo.ca

ACM Classification 1998
F.2.2 [Analysis of Algorithms and Problem Complexity] Nonnumerical Algorithms and Problems – Geometrical problems and computations, G.2.1 [Discrete Mathematics] Combinatorics, I.3.5 [Computer Graphics] Computational Geometry and Object Modeling

ISBN 978-3-95977-009-5

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-009-5.

Publication date
June, 2016

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 http://dnb.d-nb.de.

License
This work is licensed under a Creative Commons Attribution 3.0 Unported license (CC-BY 3.0): http://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.2016.i

ISBN 978-3-95977-009-5 **ISSN 1868-8969** **http://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

- Susanne Albers (TU München)
- Chris Hankin (Imperial College London)
- Deepak Kapur (University of New Mexico)
- Michael Mitzenmacher (Harvard University)
- Madhavan Mukund (Chennai Mathematical Institute)
- Catuscia Palamidessi (INRIA)
- Wolfgang Thomas (*Chair*, RWTH Aachen)
- Pascal Weil (CNRS and University Bordeaux)
- Reinhard Wilhelm (Saarland University)

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

Invited Talks

Regular Papers

32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Multimedia Contributions

Foreword

For many years, the premier annual event of the Computational Geometry community has been the International Symposium on Computational Geometry (SoCG). The 32nd SoCG was held as part of CG Week 2016 at Tufts University in Boston, USA, June 14–17, 2016. These proceedings consist of the contributions that were selected for SoCG'16: abstracts of invited talks, research papers, and video and multimedia presentations.

There were 161 papers submitted to SoCG'16, of which the program committee selected 61 for presentation and inclusion in the proceedings. The thorough review process was conducted using 249 external reviewers, and each paper received at least three reviews. Online submission and reviews were conducted using EasyChair. Selected papers have been invited to the special issues of *Discrete & Computational Geometry* and the *Journal of Computational Geometry* dedicated to the best papers of SoCG'16.

The Best Paper Award went to the paper "The Number of Holes in the Union of Translates of a Convex Set in Three Dimensions" by Boris Aronov, Otfried Cheong, Michael Gene Dobbins and Xavier Goaoc. The Best Student Presentation Award was determined and announced at the symposium, based on input of the attendees.

In addition to the technical papers, there were nine submissions received in response to the Call for Multimedia. All nine were reviewed and accepted for presentation. The extended abstracts that describe the accepted submissions are included in these proceedings. The final versions of the multimedia content are archived at `http://www.computational-geometry.org`. Also included in the proceedings are the abstracts of the two invited talks by Daniela Rus, "Toward Pervasive Robots", and by Jacob Fox, "Discrete Geometry, Algebra, and Combinatorics".

We thank the authors of submitted papers, videos and multimedia presentations. Many other people generously devoted time and effort to the quality and success of SoCG and CG Week. We especially thank the local organizers, the external reviewers, and the members of the SoCG Program Committee, the Multimedia Committee, and the Workshops and YRF Committees.

Sándor Fekete
Program Committee co-chair
TU Braunschweig

Anna Lubiw
Program Committee co-chair
University of Waterloo

Maarten Löffler
Multimedia Committee chair
Utrecht University

32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

SoCG Program Committee

Sándor Fekete *(co-chair, TU Braunschweig, Germany)*
Anna Lubiw *(co-chair, University of Waterloo, Canada)*
Mohammad Ali Abam *(Sharif University, Iran)*
Nina Amenta *(University of California at Davis, USA)*
Ulrich Bauer *(TU Munich, Germany)*
Sergio Cabello *(University of Ljubljana, Slovenia)*
Jean Cardinal *(Université Libre de Bruxelles, Belgium)*
Éric Colin de Verdière *(École normale supérieure, Paris and CNRS, France)*
Marc Glisse *(Inria, France)*
David Gu *(Stony Brook University, USA)*
Matias Korman *(Tohoku University, Japan)*
Wolfgang Mulzer *(FU Berlin, Germany)*
Joseph O'Rourke *(Smith College, USA)*
Jeff M. Phillips *(University of Utah, USA)*
Micha Sharir *(Tel Aviv University, Israel)*
Takeshi Tokuyama *(Tohuku University, Japan)*
Géza Tóth *(Rényi Institute, Hungary)*
Kevin Verbeek *(TU Eindhoven, Netherlands)*
Yusu Wang *(Ohio State University, USA)*
Emo Welzl *(ETH Zurich, Switzerland)*
Chee Yap *(New York University, USA)*

Multimedia Program Committee

Maarten Löffler *(chair, Utrecht University, The Netherlands)*
Martin Demaine *(MIT, USA)*
William Evans *(University of British Columbia, Canada)*
Michael Hoffmann *(ETH Zürich, Switzerland)*
Irina Kostitsyna *(TU Eindhoven, the Netherlands)*
Martin Nöllenburg *(TU Wien, Austria)*
Don Sheehy *(University of Connecticut, USA)*
Birgit Vogtenhuber *(TU Graz, Austria)*

Workshop Program Committee

Yusu Wang *(chair, Ohio State University, USA)*
Suresh Venkatasubramanian *(University of Utah, USA)*

32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw

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

Young Researchers Forum Program Committee

Erin Chambers *(chair, Saint Louis University, USA)*
Therese Biedl *(University of Waterloo, Canada)*
Vin de Silva *(Pomona College, USA)*
Vida Dujmović *(University of Ottawa, Canada)*
Jeff Phillips *(University of Utah, USA)*
Donald Sheehy *(University of Connecticut, USA)*
Mikael Vejdemo-Johansson *(Stockholm, Sweden)*
Carola Wenk *(Tulane University, USA)*

Local Organizers

Greg Aloupis *(chair, Tufts University, USA)*
Hee-Kap Ahn *(Pohang University of Science and Technology, South Korea)*
Diane Souvaine *(Tufts University, USA)*
Csaba Tóth *(California State University Northridge, USA)*

Steering Committee (2013–2016)

Jeff Erickson *(chair, University of Illinois at Urbana-Champaign, USA)*
David Eppstein *(secretary, University of California, Irvine, USA)*
Mark de Berg *(TU Eindhoven, the Netherlands)*
Joseph S. B. Mitchell *(Stony Brook University, USA)*
Günter Rote *(Freie Universität Berlin, Germany)*

Steering Committee (2016–)

Monique Teillaud *(chair, INRIA Nancy - Grand Est, France)*
Danny Halperin *(secretary, Tel Aviv University, Israel)*
Joseph S. B. Mitchell *(treasurer, Stony Brook University, USA)*
Erin Chambers *(Saint Louis University, USA)*
Marc van Kreveld *(Utrecht University, the Netherlands)*
David Mount *(University of Maryland, USA)*

External Reviewers

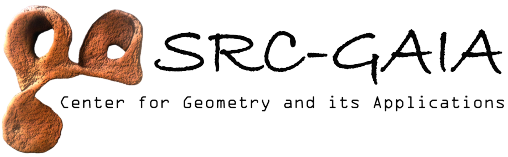
Zachary Abel
Michal Adamaszek
Peyman Afshani
Oswin Aichholzer
Noga Alon
Helmut Alt
Victor Alvarez
Nima Anari
Alexandr Andoni
Patrizio Angelini
Chidambaram Annamalai
Sang Won Bae
Jean-Francois Baffier
Aritra Banik
Bahareh Banyassady
János Barát
Luis Barba
Yair Bartal
Abdul Basit
Alexander Barvinok
Paul Bendich
Huxley Bennett
Thomas Bläsius
Cecilia Bohler
Jean-Daniel Boissonnat
Miklós Bóna
Steffen Borgwardt
Prosenjit Bose
Peter Brass
David Bremner
Srecko Brlek
Peter Bubenik
Mickaël Buchet
Kevin Buchin
Maike Buchin
Michael Burr
Norbert Bus
Imre Bárány
Paz Carmi
Hamish Carr
Erin Chambers
Amit Chattopadhyay
Frédéric Chazal
Chao Chen
Siu-Wing Cheng
Yi-Jen Chiang
Markus Chimani
Man Kwun Chiu
Serafino Cicerone
Vincent Cohen-Addad
Justin Curry
Marco Cuturi
Mark de Berg
Jean-Lou De Carufel
Arnaud De Mesmay
Frank de Zeeuw
Olivier Devillers
Tamal Dey
Giuseppe Antonio Di Luna
Walter Didimo
Yago Diez Donoso
Michael Gene Dobbins
Anne Driemel
Vida Dujmović
Adrian Dumitrescu
Laurent Dupont
Jérôme Durand-Lose
Ramsay Dyer
Herbert Edelsbrunner
Alon Efrat
Ronen Eldan
Ioannis Emiris
David Eppstein
Omid Etesami
Esther Ezra
Martin Farach-Colton
Mohammad Farshi
Stefan Felsner
Vissarion Fisikopoulos
Kyle Fox
Fabrizio Frati
Florian Frick
Radoslav Fulek
Natalia García-Colín
Cyril Gavoille
Panos Giannopoulos
Matt Gibson
Joachim Giesen
Xavier Goaoc
Joachim Gudmundsson
Anna Gundert
Bernd Gärtner
Mohammadtaghi Hajiaghayi
Dan Halperin
Sariel Har-Peled
Matthias Henze
John Hershberger
Michael Hoffmann
Udo Hoffmann
Takashi Horiyama
Alfredo Hubard
Geoffrey Irving
Martin Jaggi
Bart M. P. Jansen
Tibor Jordán
Michael Joswig
Matthew Kahle
Volker Kaibel
Iyad Kanj
Haim Kaplan
Matthew Katz
Michael Kaufmann
Akitoshi Kawamura
Michael Kerber
Heuna Kim
David Kirkpatrick
Rolf Klein
Boris Klemz
Christian Knauer
Kevin Knudson
Klaus Kriegel
Roland Kwitt
Stefan Langerman
Francis Lazarus
Jyh-Ming Lien
Andre Lieutier

32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Giuseppe Liotta
Frank H. Lutz
Maarten Löffler
Isaac Mabillard
Sepideh Mahabadi
Clément Maria
Benjamin Matschke
Pooran Memari
Quentin Merigot
Wouter Meulemans
Piotr Micek
Malte Milatz
Victor Milenkovic
Joseph Mitchell
Bojan Mohar
Morteza Monemizadeh
Pat Morin
Guillaume Moroz
David Mount
Elizabeth Munch
Nabil Mustafa
Torsten Mütze
Amir Nayyeri
Eran Nevo
Aleksandar Nikolov
Mostafa Nouri Baygi
Jerri Nummenpalo
Martin Nöllenburg
Yoshio Okamoto
Yota Otachi
Steve Oudot
Shayan Oveis Gharan
Kenta Ozeki
Evanthia Papadopoulou
Salman Parsa
Amit Patel
Florian Pausinger
Petar Pavešić
Michael Payne
Jose Perea
Vincent Pilaud
Alexander Pilz
Marc Pouget
Simon Pratt
Kirk Pruhs
Pablo Pérez-Lantero
Sharath Raghvendra
Zahed Rahmati
Benjamin Raichel
Pedro Ramos
Marcel Roeloffzen
Günter Rote
Ignaz Rutter
Ankan Saha
Raman Sanyal
Maria Saumell
Marcus Schaefer
Nadja Scharf
Stefan Schirra
Jean-Marc Schlenker
Lena Schlipf
Christiane Schmidt
Patrick Schnider
André Schulz
Paul Seiferth
C. Seshadhri
Vikram Sharma
Don Sheehy
Adam Sheffer
Jonathan Shewchuk
Akiyoshi Shioura
Anastasios Sidiropoulos
Rodrigo Silveira
Primoz Skraba
Shakhar Smorodinsky
Christian Sohler
Jozsef Solymosi
Wanbin Son
Bettina Speckmann
Jonathan Spreer
Frank Staals
Yannik Stein
András Stipsicz
Subhash Suri
Konrad Swanepoel
Toshie Takata
Martin Tancer
Yufei Tao
Gabor Tardos
Monique Teillaud
Stephan Tillmann
Csaba Tóth
Elias Tsigaridas
Mimi Tsuruga
Hemant Tyagi
Ryuhei Uehara
Pavel Valtr
Marc Van Kreveld
André van Renssen
Kasturi Varadarajan
Mikael Vejdemo-Johansson
Suresh Venkatasubramanian
Sander Verdonschot
Giovanni Viglietta
Antoine Vigneron
Birgit Vogtenhuber
Hubert Wagner
Uli Wagner
Bei Wang
Haitao Wang
Larry Wasserman
Rephael Wenger
Carola Wenk
Manuel Wettstein
Mathijs Wintraecken
Steve Wismath
Christian Wulff-Nilsen
Yukiko Yamauchi
Hai Yu
Xiang Yu
Alireza Zarei
Günter Ziegler

Sponsors

We gratefully acknowledge the financial support received from the sponsors of CG Week 2016: National Science Foundation (NSF), The Fields Institute for Research in Mathematical Sciences, Tufts University, Princeton University, SRC-GAIA (The Center for Geometry and its Applications), MITRE, Mentor Graphics, Madalgo (Center for Massive Data Algorithmics) and MIT Lincoln Laboratory.



32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Toward Pervasive Robots

Daniela Rus

CSAIL, MIT, Cambridge, USA
rus@csail.mit.edu

Abstract

The digitization of practically everything coupled with the mobile Internet, the automation of knowledge work, and advanced robotics promises a future with democratized use of machines and wide-spread use of robots and customization. However, pervasive use of robots remains a hard problem. Where are the gaps that we need to address in order to advance toward a future where robots are common in the world and they help reliably with physical tasks? What is the role of geometric reasoning along this trajectory?

In this talk I will discuss challenges toward pervasive use of robots and recent developments in geometric algorithms for customizing robots. I will focus on a suite of geometric algorithms for automatically designing, fabricating, and tasking robots using a print-and-fold approach. I will also describe how geometric reasoning can play a role in creating robots more capable of reasoning in the world. By enabling on-demand creation of programmable robots, we can begin to imagine a world with one robot for every physical task.

1998 ACM Subject Classification I.2.9 Robotics

Keywords and phrases Pervasive robots, geometric algorithms, print-and-fold

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.1

Category Invited Talk

© Daniela Rus;
licensed under Creative Commons License CC-BY
32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw; Article No. 1; pp. 1:1–1:1

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

Discrete Geometry, Algebra, and Combinatorics

Jacob Fox

Department of Mathematics, Stanford University, Stanford, USA
jacobfox@stanford.edu

Abstract

Many problems in discrete and computational geometry can be viewed as finding patterns in graphs or hypergraphs which arise from geometry or algebra. Famous Ramsey, Turán, and Szemerédi-type results prove the existence of certain patterns in graphs and hypergraphs under mild assumptions. We survey recent results which show much stronger/larger patterns for graphs and hypergraphs that arise from geometry or algebra. We further discuss whether the stronger results in these settings are due to geometric, algebraic, combinatorial, or topological properties of the graphs.

1998 ACM Subject Classification G.2 Discrete Mathematics, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Discrete geometry, extremal combinatorics, regularity lemmas, Ramsey theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.2

Category Invited Talk

© Jacob Fox;
licensed under Creative Commons License CC-BY
32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw; Article No. 2; pp. 2:1–2:1

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

Who Needs Crossings? Hardness of Plane Graph Rigidity

Zachary Abel[*1], **Erik D. Demaine**[2], **Martin L. Demaine**[3], **Sarah Eisenstat**[4], **Jayson Lynch**[5], **and Tao B. Schardl**[6]

1 **MIT Department of Mathematics, 77 Massachusetts Ave., Cambridge, MA 02139, USA**
zabel@math.mit.edu
2 **MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, USA**
edemaine@mit.edu
3 **MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, USA**
mdemaine@mit.edu
4 **MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, USA**
seisenst@mit.edu
5 **MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, USA**
jaysonl@mit.edu
6 **MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, USA**
neboat@mit.edu

Abstract

We exactly settle the complexity of graph realization, graph rigidity, and graph global rigidity as applied to three types of graphs: "globally noncrossing" graphs, which avoid crossings in all of their configurations; matchstick graphs, with unit-length edges and where only noncrossing configurations are considered; and unrestricted graphs (crossings allowed) with unit edge lengths (or in the global rigidity case, edge lengths in $\{1, 2\}$). We show that all nine of these questions are complete for the class $\exists\mathbb{R}$, defined by the Existential Theory of the Reals, or its complement $\forall\mathbb{R}$; in particular, each problem is (co)NP-hard.

One of these nine results – that realization of unit-distance graphs is $\exists\mathbb{R}$-complete – was shown previously by Schaefer (2013), but the other eight are new. We strengthen several prior results. Matchstick graph realization was known to be NP-hard (Eades & Wormald 1990, or Cabello et al. 2007), but its membership in NP remained open; we show it is complete for the (possibly) larger class $\exists\mathbb{R}$. Global rigidity of graphs with edge lengths in $\{1, 2\}$ was known to be coNP-hard (Saxe 1979); we show it is $\forall\mathbb{R}$-complete.

The majority of the paper is devoted to proving an analog of Kempe's Universality Theorem – informally, "there is a linkage to sign your name" – for globally noncrossing linkages. In particular, we show that any polynomial curve $\varphi(x, y) = 0$ can be traced by a noncrossing linkage, settling an open problem from 2004. More generally, we show that the nontrivial regions in the plane that may be traced by a noncrossing linkage are precisely the compact semialgebraic regions. Thus, no drawing power is lost by restricting to noncrossing linkages. We prove analogous results for matchstick linkages and unit-distance linkages as well.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Graph Drawing, Graph Rigidity Theory, Graph Global Rigidity, Linkages, Complexity Theory, Computational Geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.3

[*] Partially supported by an NSF Graduate Research Fellowship.

© Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B. Schardl;
licensed under Creative Commons License CC-BY

Table 1 Summary of our results (bold) compared with old results (cited). The rows give the special types of graphs considered. The middle three columns give complexity results for the three natural decision problems about graph embedding; all completeness results are strong. The rightmost column gives the exact characterization of drawable sets.

Graph type	Realization	Rigidity	Global rigidity	Universality
General	$\exists\mathbb{R}$-complete [17]	$\forall\mathbb{R}$-complete [17]	**$\forall\mathbb{R}$-complete** (CoNP-hard [16])	Compact semi-algebraic [12]
Globally noncrossing *(no configs. cross)*	**$\exists\mathbb{R}$-complete**	**$\forall\mathbb{R}$-complete**	**$\forall\mathbb{R}$-complete**	**Compact semialgebraic**
Matchstick graph *(unit + noncrossing)*	**$\exists\mathbb{R}$-complete** (NP-hard [6])	**$\forall\mathbb{R}$-complete**	**$\forall\mathbb{R}$-complete**	**Bounded semialgebraic**
Unit edge lengths *(allowing crossings)*	$\exists\mathbb{R}$-complete [17]	**$\forall\mathbb{R}$-complete**	Open (do they even exist?)	**Compact semialgebraic**
Edge lengths in $\{1, 2\}$ *(allowing crossings)*	$\exists\mathbb{R}$-complete [17]	**$\forall\mathbb{R}$-complete**	**$\forall\mathbb{R}$-complete** (CoNP-hard [16])	**Compact semialgebraic**

1 Introduction

The rise of the steam engine in the mid-1700s led to an active study of *mechanical linkages*, typically made from rigid bars connected together at hinges. For example, steam engines need to convert the linear motion of a piston into the circular motion of a wheel, a problem solved approximately by Watt's parallel motion (1784) and exactly by Peaucellier's inversor (1864) [5, Section 3.1]. These and other linkages are featured in an 1877 book called *How to Draw a Straight Line* [10] by Alfred Bray Kempe – a barrister and amateur mathematician in London, perhaps most famous for his false "proof" of the Four-Color Theorem [11] that nonetheless introduced key ideas used in the correct proofs of today [2, 15].

Kempe's Universality Theorem

Kempe wondered far beyond drawing a straight line by turning a circular crank. In 1876, he claimed a universality result, now known as Kempe's Universality Theorem: every polynomial curve $\varphi(x, y) = 0$ can be traced by a vertex of a 2D linkage [9]. Unfortunately, his "proof" was again flawed: the linkage he constructs indeed traces the intended curve, but also traces finitely many unintended additional curves. Fortunately, his idea was spot on.

Many researchers have since solidified and/or strengthened Kempe's Universality Theorem [8, 7, 12, 1, 17]. In particular, small modifications to Kempe's gadgets lead to a working proof [1, 5, Section 3.2]. Furthermore, the regions of the plane drawable by a 2D linkage (other than the entire plane $\mathbb{R}^2$) are exactly compact semialgebraic regions[1] [12, 1]. By carefully constructing these linkages to have rational coordinates, Abbott et al. [1] showed how to reduce the problem of testing isolatedness of a point in an algebraic set to testing rigidity of a linkage. Isolatedness was proved coNP-hard [13] and then $\forall\mathbb{R}$-complete[2] [17]; thus linkage rigidity is $\forall\mathbb{R}$-complete.

[1] A compact planar region is **semialgebraic** if it can be obtained by intersecting and/or unioning finitely many basic sets defined by polynomial inequalities $p(x, y) \geq 0$.

[2] The class $\forall\mathbb{R}$ = co-$\exists\mathbb{R}$ consists of decision problems whose complement (inverting yes/no instances) belong to $\exists\mathbb{R}$. The class $\exists\mathbb{R}$ refers to the problems (Karp) reducible to the **existential theory of the reals** ($\exists x_1 : \cdots \exists x_n : \pi(x_1, \ldots, x_n)$ for a Boolean function $\pi : \mathbb{R} \to \{0, 1\}$), which is somewhere between NP and PSPACE (by [4]). The classic example of an $\exists\mathbb{R}$-complete problem is pseudoline stretchability [14].

Our results: no crossings

See Table 1 for a summary of our results in comparison to past results. Notably, all known linkage constructions for Kempe's Universality Theorem (and its various strengthenings) critically need to allow the bars to cross each other. In practice, certain crossings can be made physically possible, by placing bars in multiple parallel planes and constructing vertices as vertical pins. Without extreme care, however, bars can still be blocked by other pins, and it seems difficult to guarantee crossing avoidance for complex linkages. Beyond these practical issues, it is natural to wonder whether allowing bars to cross is necessary to achieve linkage universality. Don Shimamoto first posed this problem in April 2004, and it was highlighted as a key open problem in the first chapter of *Geometric Folding Algorithms* [5].

We solve this open problem by strengthening most of the results mentioned above to work for **globally noncrossing graphs**, that is, graphs plus edge-length constraints that alone force all configurations to be (strictly) noncrossing.[3] In particular, we prove the following universality and complexity results:

1. The planar regions drawable by globally noncrossing linkages are exactly the compact semialgebraic regions (and $\mathbb{R}^2$), settling Shimamoto's 2004 open problem.
2. Testing whether a globally noncrossing graph has any valid configurations is $\exists\mathbb{R}$-complete.
3. Testing rigidity is strongly $\forall\mathbb{R}$-complete even for globally noncrossing linkages drawn with polynomially bounded integer vertex coordinates and constant-sized integer edge lengths.
4. Testing global rigidity (uniqueness of a given embedding) is strongly $\forall\mathbb{R}$-complete even for globally noncrossing linkages drawn with polynomially bounded integer vertex coordinates and constant-sized integer edge lengths.

Our techniques are quite general and give us results for two other restricted forms of graphs as well. First, **matchstick graphs** are graphs with *unit* edge-length constraints, and where only (strictly) noncrossing configurations are considered valid. We prove the following universality and complexity results:

5. The planar regions drawable by matchstick graphs are exactly the bounded semialgebraic regions (and $\mathbb{R}^2$). Notably, unlike all other models considered, matchstick graphs enable the representation of open boundaries in addition to closed (compact) boundaries.
6. Recognizing matchstick graphs is (strongly) $\exists\mathbb{R}$-complete. This result strengthens a 25-year-old NP-hardness result [6, 3], and settles an open question of [17].
7. Testing rigidity or global rigidity of a matchstick graph is strongly $\forall\mathbb{R}$-complete.

Second, we consider restrictions on edge lengths to be either all equal (unit) or all in $\{1, 2\}$, but at the price of allowing crossing configurations. Recognizing unit-distance graphs is already known to be $\exists\mathbb{R}$-complete [17]. We prove the following additional universality and complexity results:

8. The planar regions drawable by unit-edge-length linkages are exactly the compact semialgebraic regions (and $\mathbb{R}^2$), proving a conjecture of Schaefer [17].
9. Testing rigidity of unit-edge-length linkages is strongly $\forall\mathbb{R}$-complete, proving a conjecture of Schaefer [17].
10. Testing global rigidity of linkages with edge lengths in $\{1, 2\}$ is strongly $\forall\mathbb{R}$-complete. This result strengthens a 35-year-old strong-coNP-hardness result for the same scenario

[3] Thus, the noncrossing constraint can be thought of as being "required" or not of a configuration; in either case, the configurations (even those reachable by discontinuous motions) will be noncrossing.

[16]. While it would be nice to strengthen this result to unit edge lengths, we have been unable to find even a single globally rigid equilateral linkage larger than a triangle.

We introduce several techniques to make noncrossing linkages manageable in this setting. In Section 4.1 we define *extended linkages* to allow additional joint types, in particular, requiring angles between pairs of bars to stay within specified intervals. Section 4.2 then shows how to draw a polynomial curve and obtain Kempe's Universality Theorem with these powerful linkages while avoiding crossings, by following the spirit of Kempe's original construction but with specially designed modular gadgets to guarantee no crossings between (or within) the gadgets. We simulate extended linkage with linkages that have chosen subgraphs marked as rigid. In turn, in Section 3, we simulate these "partially rigidified" linkages with the three desired linkage types: globally noncrossing, unit-distance or $\{1, 2\}$-distance, and matchstick.

2 Description of the Main Construction

The heart of this paper is a single, somewhat intricate linkage construction. In this section, we describe and discuss the properties of this construction in detail, after building up the necessary terminology.

2.1 Linkages and Graphs

Unless otherwise specified, all graphs $G = (V(G), E(G), \ell_G)$ in this text are connected, edge-weighted with positive edge lengths $\ell_G(e) > 0$, and contain no self-loops.

We use standard definitions for **abstract and configured linkages/graphs** the **configuration space** of a linkage/graph, and rigidity and global rigidity of a linkage/graph. (For concrete notation, a linkage is specified by a weighted graph G together with a choice of **pin locations** $P(w) \in \mathbb{R}^2$ for vertices w in a chosen subset $W \subset V(G)$ of **pinned vertices**.)

A configuration is called **noncrossing** if it has no edge intersections in the plane (other than common endpoints of adjacent edges); a linkage all of whose configurations are noncrossing is called **globally noncrossing**. For such a linkage, the **global minimum feature size** is defined as the infimum of the minimum feature size of the configurations.

A **combinatorial embedding** σ for a graph G consists of a cyclic ordering σ_v of v's incident edges for each vertex $v \in V(G)$, and a configuration C **agrees with** σ if v's edges are arranged counterclockwise around point $C(v)$ in order σ_v. Whenever edge (v, u) is followed by (v, w) in σ_v, the two-edge path $\Lambda = (u, v, w)$ is an **angle chain** of σ at v. (See the full paper for complete definitions.)

2.2 Constrained Linkages

We will make use of a number of special-purpose "constraints" or "annotations" that may be attached to linkages to artificially modify their behavior, such as "rigid constraints" that "rigidify" a subgraph into a chosen configuration while allowing the rest of the linkage to move freely. These annotations do not affect the linkage itself; instead, they merely indicate which configurations of the linkage they consider acceptable. The language of constraints allows us to separate a desired *effect* from the *implementation* or *construction* that enforces that effect.

▶ **Definition 2.1.** A **constraint** Con on an abstract linkage $\mathcal{L}$ is specified by a subset of the configuration space, $\text{Con} \subseteq \text{Conf}(\mathcal{L})$, and we say the configurations $C \in \text{Con}$ **satisfy**

constraint Con. A **constrained linkage** $\mathcal{L}$ is an abstract linkage $\mathcal{L}_0$ together with a finite set K of constraints on $\mathcal{L}_0$, and the **constrained configuration space** is defined as $\mathrm{Conf}(\mathcal{L}) := \mathrm{Conf}(\mathcal{L}_0) \cap \bigcap_{\mathrm{Con}\in K} \mathrm{Con}$. In other words, constrained linkage $\mathcal{L}$ simply ignores any configurations of $\mathcal{L}_0$ that don't satisfy all of its constraints.

All terms discussed in Section 2.1 – realizability, rigidity, global rigidity, etc. – apply to equally well to constrained linkages via their *constrained* configuration space.

▶ **Definition 2.2.** A **rigid constraint** $\mathrm{RigidCon}_{\mathcal{L}}(H, C_H)$ on a linkage $\mathcal{L} = (G, W, P)$ is specified by a connected subgraph $H \subseteq G$[4] together with a configuration C_H of H. A configuration $C \in \mathrm{Conf}(\mathcal{L})$ satisfies the rigid constraint when C induces a configuration $C|_H$ on H that is congruent to the given C_H, i.e., differs only by a (possibly orientation-reversing) Euclidean transformation. When a constrained linkage $\mathcal{M}$ contains constraint $\mathrm{RigidCon}_{\mathcal{M}}(H, C_H)$, we say (H, C_H) is a **rigidified subgraph** of $\mathcal{M}$. A constrained linkage all of whose constraints are rigid constraints is called a **partially rigidified linkage**.

2.3 Drawing with Linkages and Graphs

▶ **Definition 2.3** (Linkage Trace and Drawing)**.** For a linkage $\mathcal{L}$ and a tuple $X = (v_1, \ldots, v_k)$ of distinct vertices of $\mathcal{L}$, the **trace** of X is defined as the image $\pi_X(\mathrm{Conf}(\mathcal{L})) \subset (\mathbb{R}^2)^k$, where π_X is the projection map sending $C \in \mathrm{Conf}(\mathcal{L})$ to $\pi_X(C) := (C(v_1), \ldots, C(v_k))$. A linkage $(\mathcal{L}, X)$ is said to **draw**[5] its trace, and a set $R \subseteq (\mathbb{R}^2)^k$ is **drawable (by a linkage)** if it can be expressed as the trace of some k vertices of a linkage.

We single out some drawings as particularly nice:

▶ **Definition 2.4** (Rigid Drawing)**.** Say $(\mathcal{L}, X)$ draws its trace **rigidly** if the map π_X has finite fibers, i.e., for any $p \in \pi_X(\mathrm{Conf}(\mathcal{L}))$, there are only finitely many configurations $C \in \mathrm{Conf}(\mathcal{L})$ with $\pi_X(C) = p$.

In particular, if p is isolated in $\pi_X(\mathrm{Conf}(\mathcal{L}))$, then any configuration C with $\pi_X(C) = p$ is rigid, because the discrete set $\pi_X^{-1}(p)$ contains no nonconstant continuous paths.

▶ **Definition 2.5** (Continuous Drawing)**.** Say $(\mathcal{L}, X)$ draws its trace **continuously** if the map π_X has the **path lifting property**: for any configuration $C \in \mathrm{Conf}(\mathcal{L})$ and path $\gamma : [0, 1] \to \pi_X(\mathrm{Conf}(\mathcal{L}))$ in the trace starting at $\pi_X(C)$, there is a path $\gamma' : [0, 1] \to \mathrm{Conf}(\mathcal{L})$ starting at $\gamma'(0) = C$ and lifting γ, i.e., $\gamma = \pi_X \circ \gamma'$.

In particular, if a point $p \in \pi_X(\mathrm{Conf}(\mathcal{L}))$ is *not* isolated, then any configuration C with $\pi_X(C) = p$ is *not* rigid, because a nontrivial continuous path beginning at p can be lifted to a nontrivial path beginning at C. We are especially interested in cases where $(\mathcal{L}, X)$ draws both continuously and rigidly; these concepts were introduced in [1] for their usefulness in proving computational hardness of linkage rigidity, and we rely on them for similar purposes. We make use of an even stronger notion as well:

▶ **Definition 2.6** (Perfect Drawing)**.** If the map π_X is a *homeomorphism* between $\mathrm{Conf}(\mathcal{L})$ and the trace, we say $(\mathcal{L}, X)$ draws **perfectly**.

▶ **Definition 2.7** (Linkage Simulation)**.** When $(\mathcal{L}, X)$ draws precisely the full configuration space $\mathrm{Conf}(\mathcal{M})$ of another linkage $\mathcal{M}$, we say that $(\mathcal{L}, X)$ **simulates $\mathcal{M}$**. It may **continuously, rigidly, or perfectly simulate $\mathcal{M}$** if it draws $\mathrm{Conf}(\mathcal{M})$ in this manner.

[4] Our notion of "subgraph" requires the edge-lengths of H to agree with those in G.
[5] This "drawing" need not be continuous. For example, the linkage may have a disconnected trace.

2.4 Specification of Main Theorem

For a collection $F = \{f_1, \ldots, f_s\}$ of polynomials in $\mathbb{R}[x_1, y_1, \ldots, x_m, y_m] = \mathbb{R}[\overrightarrow{xy}]$, the **algebraic set** defined by F is the set of common zeros,

$$Z(F) := \{\overrightarrow{xy} \in \mathbb{R}^{2m} \mid f_1(\overrightarrow{xy}) = \cdots = f_s(\overrightarrow{xy}) = 0\}.$$

The primary technical construction in this paper builds a globally noncrossing, partially rigidified linkage $\mathcal{L}(F)$ that draws precisely the algebraic set $Z(f_1, \ldots, f_s) \subseteq \mathbb{R}^{2m}$, or at least a bounded piece thereof, up to a translation of $\mathbb{R}^{2m}$. Why is the translation necessary? Without it, some algebraic sets would require the drawing vertices in X to collocate at some or all of the linkage's configuration space[6], precluding the possibility of global noncrossing.

We are now prepared to precisely specify the properties of this construction, from which the results listed in Table 1 follow as corollaries. We thoroughly detail these properties here, so that the corollaries may be derived solely from Theorem 2.8's statement without referring to the specifics of its proof (with one small exception, discussed in Section 4.3). This also allows for maximal reuse: the commonalities in our arguments for our three linkage contexts – unconstrained globally noncrossing linkages in Section 3.1, unit-distance linkage in Section 3.2, and matchstick linkages in Section 3.3 – have been unified and generalized into Theorem 2.8, so only features unique to each context need be discussed in Sections 3.1–3.3.

The Main Theorem is divided into three parts because it must be used in subtly different ways by the four types of results we seek. Hardness of realizability requires a polynomial-time construction of an abstract linkage that draws $Z(f_1, \ldots, f_s)$ *without knowing whether the resulting configuration space is empty*, whereas proving hardness of rigidity and global rigidity requires the polynomial-time construction of a linkage *together with a known configuration*. We thus separate these into different Parts of Theorem 2.8 with slightly different assumptions about the input polynomials f_i (Part II for realizability, Part III for rigidity and global rigidity). When proving universality, we must prove *existence* of a linkage to draw any compact semialgebraic set, but the coefficients of the polynomials defining this set may be non-rational or non-algebraic, as might the edge-lengths and coordinates of the resulting linkage, so we isolate this in Part I, away from algorithmic and efficiency concerns.

We measure the "size" of polynomials naïvely: a polynomial $f \in \mathbb{R}[x_1, y_1, \ldots, x_m, y_m]$ with total degree d is specified by $\#\text{Coeffs}(f) := \binom{2m+d}{d} = \text{poly}(m^d, d^d)$ real coefficients, using dense representation.[7] If f's coefficients are integers with maximum magnitude M, we record its size as $\text{Size}(f) := M \cdot \#\text{Coeffs}(f) = \text{poly}(m^d, d^d, M)$ unary digits (not binary!). For a set $F = \{f_1, \ldots, f_s\}$ of s polynomials in $\mathbb{R}[x_1, y_1, \ldots, x_m, y_m]$ with maximum total degree d, we set $\#\text{Coeffs}(F) := s \cdot M \cdot \binom{2m+d}{d} = \text{poly}(m^d, d^d, s)$, and if these coefficients are integers with maximum magnitude M, then $\text{Size}(F) := s \cdot M \cdot \binom{2m+d}{d} = \text{poly}(s, M, m^d, d^d)$.

▶ **Theorem 2.8.**

Part I. *Take as input a collection of polynomials $F = \{f_1, \ldots, f_s\}$, each in $\mathbb{R}[x_1, y_1, \ldots, x_m, y_m]$. Then we may construct a partially rigidified linkage $\mathcal{L} = \mathcal{L}(F)$ that draws, up to translation, a bounded portion of the algebraic set $Z(F)$: specifically, there is a translation T on $\mathbb{R}^{2m}$ and a subset X of m vertices of $\mathcal{L}$ such that*

$$T(Z(F) \cap [-1, 1]^{2m}) \subseteq \pi_X(\text{Conf}(\mathcal{L})) \subseteq T(Z(F)).$$

[6] For example, two distinct vertices that draw the trace $\{((0,0),(0,0))\} \subset (\mathbb{R}^2)^2$ must meet.

[7] The measure $\#\text{Coeffs}(f)$ does *not* count the nonzero coefficients of f; instead, it counts the total number of monomials that have total degree at most that of f.

Furthermore,

1. *Vertices X draw this trace rigidly and continuously.*
2. *The number of vertices and edges in $\mathcal{L}$ is* $\text{poly}(\#\text{Coeffs}(F))$.
3. *Each edge of $\mathcal{L}$ has length $\Omega(1)$, and $\mathcal{L}$ is globally noncrossing with global minimum feature size $\Omega(1)$.*
4. *For each constraint* $\text{RigidCon}_{\mathcal{L}}(H, C_H)$ *on $\mathcal{L}$, H is a tree that connects to $G \setminus H$ precisely at leaves of H, and configuration C_H has all edges parallel to the x- or y-axes. Each edge of G is contained in at most one rigidified subgraph (H, C_H).*
5. *There is a combinatorial embedding σ of G such that every configuration $C \in \text{Conf}(\mathcal{L})$ agrees with σ. Furthermore, if v is not an internal vertex of any constrained tree H, then for each angle chain Λ at v, angle $\angle C(\Lambda)$ lies strictly between $60°$ and $240°$.*
6. *Linkage $\mathcal{L}$ has precisely $|P| = 3$ pinned vertices, which belong to one of the rigidified trees (H, C_H) and are not collinear in C_H.*

Part II. *If polynomials f_i have integer coefficients, we may bound the complexity of $\mathcal{L}$ as follows:*

7. *All edge-lengths in $\mathcal{L}$ are rational, with numerators bounded by* $\text{poly}(\text{Size}(F))$ *and denominators bounded by $O(1)$.*
8. *Constrained linkage $\mathcal{L}$, the set X of vertices, translation T, and combinatorial embedding σ may be constructed from F deterministically in time* $\text{poly}(\text{Size}(F))$.

Part III. *Finally, if the polynomials f_i each satisfy $f_i(\overrightarrow{0}) = 0$, we may additionally compute an initial configuration C_0 satisfying:*

9. *All coordinates of C_0 are rational numbers with magnitude bounded by* $\text{poly}(\text{Size}(F))$ *and with $O(1)$ denominators.*
10. *C_0 is the* only *configuration of $\mathcal{L}$ that projects to $T(\overrightarrow{0}) \in \pi_X(\text{Conf}(\mathcal{L}))$.*
11. *C_0 may also be computed deterministically in time* $\text{poly}(\text{Size}(F))$.

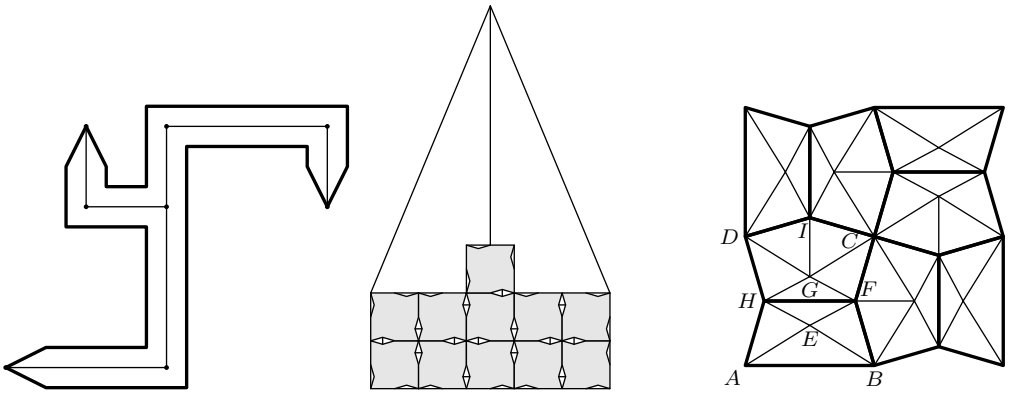
3 Using the Main Theorem: Three Linkage Models

We apply Theorem 2.8 in three separate contexts: for globally noncrossing linkages (those designed to make crossing impossible), for unit-distance or $\{1, 2\}$-distance linkages (where crossing is allowed), and for matchstick linkages (which have unit-length edges, and crossing configurations are ignored). To this end, we show that the partially rigidified linkage $\mathcal{L}(F)$ resulting from Theorem 2.8 may be simulated by each type of linkage. Primarily, this simulation is achieved by "implementing" rigidified orthogonal trees in each context: for globally noncrossing and matchstick linkages, we show that each rigid constraint in $\mathcal{L}(F)$ may be replaced by a sufficiently narrow rigid assembly. The unit-distance linkages are easier, as crossings may be ignored, so we can use standard techniques.

3.1 Globally Noncrossing Linkages

For each rigidified orthogonal tree in $\mathcal{L}(F)$ (from Theorem 2.8), we may draw a polygon P that slightly thickens the tree, and then construct a globally rigid graph G whose outer boundary is P. This graph G may be constructed with both integer coordinates and integer, $O(1)$ edge-lengths (when scaled appropriately), as illustrated in Figure 1. Replacing each tree in this way, and making sure that each tree is thickened by less than half of $\mathcal{L}(F)$'s global minimum feature size (as guaranteed by property 3), results in the desired globally noncrossing linkage simulating $\mathcal{L}(F)$.

We may now sketch proofs for hardness of realizability, hardness of rigidity and global rigidity, and universality for globally noncrossing linkages.



(a) The thickened outline.

(b) Graph G is built as a polyomino with extra bars at sharp corners.

(c) Each globally rigid monomino has integer coordinates *and* integer edge-lengths: $AB = 720$, $AE = 424$, $EF = 289$, $GI = 329$, and $HF = 510$. (A, E, F are not quite collinear.)

Figure 1 Each partially rigidified orthogonal tree may be simulated with a globally rigid graph G that slightly thickens the tree.

▶ **Theorem** (Hardness of Globally Noncrossing Realizability)**.** *Deciding whether a given abstract weighted graph $\mathcal{G}$ is realizable, even when $\mathcal{G}$ is promised to be globally noncrossing and has integer edge-lengths of size $O(1)$, is strongly $\exists\mathbb{R}$-complete.*

Proof Sketch. We reduce from the CommonZero problem, which asks whether a collection of polynomials $F = \{f_1, \ldots, f_s\}$, each in $\mathbb{Z}[x_1, y_1, \ldots, x_m, y_m]$, has a common zero. This problem is $\exists\mathbb{R}$-complete, even when the polynomials have constant total degree and constant coefficients, and furthermore, all common zeroes are promised to lie in the box $[-1, 1]^{2m}$. By simulating the Main Theorem (Part II) as described above, we may construct a globally noncrossing linkage $\mathcal{M}$ that draws a translation of $Z(F)$, which means $\mathcal{M}$ is realizable exactly when $Z(F)$ is nonempty. ◀

▶ **Theorem** (Hardness of Noncrossing Rigidity and Global Rigidity)**.** *Deciding whether a given configured weighted graph $(\mathcal{G}, C_0)$ is rigid, when $\mathcal{G}$ is promised to be globally noncrossing (so in particular, C_0 is noncrossing) and C_0 has integer coordinates and constant-sized integer edge-lengths, is strongly $\forall\mathbb{R}$-complete. It remains $\forall\mathbb{R}$-complete if "rigid" is replaced by "globally rigid".*

Proof Sketch. As in [1, 17], we reduce from the complement of the $\mathsf{H_2N}$ problem, which asks whether a given set of *homogeneous* polynomials F in $\mathbb{Z}[x_1, y_1, \ldots, x_m, y_m]$ has a nonzero common root. This problem is $\exists\mathbb{R}$-hard even when the given polynomials have constant total degree and constant coefficients. By simulating the Main Theorem (Part III) as above, we may construct a configured globally noncrossing linkage $\mathcal{L}$ that continuously and rigidly draws a neighborhood of $0 \in Z(F)$ (up to translation). Then the initial configuration of $\mathcal{L}$ is flexible if and only if 0 is not isolated in $Z(F)$, which by homogeneity happens precisely when $Z(F)$ has any nonzero point. On the other hand, if $Z(F)$ contains only 0, then by property 10 of Theorem 2.8, $\mathcal{L}$ is in fact globally rigid. ◀

▶ **Theorem** (Universality of Globally Noncrossing Linkages)**.** *For any compact semialgebraic set $R \subset \mathbb{R}^2$, there is a globally noncrossing linkage $\mathcal{L}$ that draws R.*

Proof Sketch. The set R may be written as a coordinate projection of some compact *algebraic* set $R' = Z(F) \subset \mathbb{R}^{2m}$, so it suffices to draw a translation of R'. By scaling, we may assume $R' \subseteq [-1, 1]^{2m}$. Simulating the Main Theorem (Part I) as above provides the desired globally noncrossing linkage that draws a translation of R', and hence draws R. ◀

3.2 Unit- and $\{1, 2\}$-Distance Linkages

In this context, we do not use the globally noncrossing properties of the Main Construction at all. A simpler, though still careful construction is likely possible, but it does not seem that the results in this section follow straightforwardly from prior constructions, such as [1, 12, 17]. Indeed, we rely crucially on polynomially bounded integer *edge lengths*, not just coordinates. So Theorem 2.8 may be slightly overpowered for this purpose, but if all you have is a hammer...

To simulate Theorem 2.8, we begin by simulating each edge of integer length n by a **reinforced bar** graph, which is formed by adjoining $n-1$ degenerate $\{1, 1, 2\}$-sided triangles along their unit edges (as in [16]). For each rigidified orthogonal tree T, we construct a rigid integer grid, by combining many reinforced bars, and inserting one more reinforced bar along the hypotenuse of a 3-4-5 right triangle to preserve orthogonality. This grid stands in for tree T, connecting at the grid nodes corresponding to T's leaves. This is sufficient to prove hardness of global rigidity for $\{1, 2\}$-distance graphs, as in the previous section.

Finally, as described in [17], length-2 edges may be simulated continuously and rigidly by unit-distance graphs (by combining two copies of Moser's Spindle), and so Theorem 2.8 itself may be simulated continuously and rigidly by unit-distance linkages.

Proofs of hardness proceed analogously to the arguments in Section 3.1, but restricting to unit edge lengths offers a noteworthy challenge for universality. To illustrate, the circle $C = \{(x, y) \mid x^2 + y^2 = r^2\}$ (where $r > 0$ is any uncomputable number, such as Chaitin's constant) can be drawn easily by a linkage (using an edge of length r), but simulating such an edge with a unit-distance graph is impossible. As a workaround, we instead rely on *pins* to introduce non-algebraic values. Indeed, we may slightly generalize curve C by introducing new variables (a, b) and considering the modified curve

$$C' = \{((x, y), (a, b)) \in \mathbb{R}^4 \mid x^2 + y^2 = a^2\}.$$

As C' is now defined by polynomials with *integer* coefficients, the Main Theorem (Part II) applies and may be simulated by a unit distance linkage as above. Finally, with one pin, we may fix the values $a = r$ and $b = 0$, which recovers the desired circle C. Suitably generalized, this argument can be made to work for arbitrary compact semialgebraic sets.

3.3 Matchstick Linkages

▶ **Definition 3.1** (Noncrossing Constraint)**.** We define a **noncrossing constraint**, $\mathrm{NXCon}_{\mathcal{L}}$, on a linkage $\mathcal{L}$ by declaring that $\mathrm{NXCon}_{\mathcal{L}}$ is only satisfied by noncrossing configurations; in other words, the constrained configuration space $\mathrm{Conf}(\mathcal{L}, \{\mathrm{NXCon}_{\mathcal{L}}\})$ is, by definition, $\mathrm{NXConf}(\mathcal{L})$. We refer to a constrained linkage with a noncrossing constraint (and no other constraints) as an **NX-constrained linkage**.

NX-constrained linkages may seem similar to globally noncrossing linkages, but there is an important distinction. Very few linkages are globally noncrossing – this is a stringent, intrinsic *property* that the linkage must satisfy. By contrast, *any* linkage can be annotated with an NX-constraint, which does not change the fact that crossing configurations may exist,

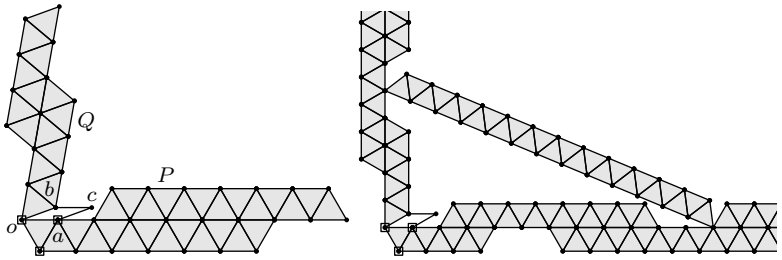


Figure 2 Left: Edge polyiamonds used to simulate edges of integer length. Right: Edge polyiamonds braced at 90°.

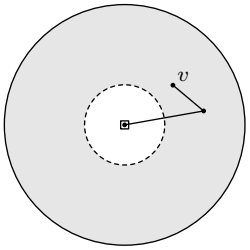


Figure 3 The trace of an NX-constrained linkage need not be closed.

but instead simply tells the observer to ignore them. For example, if $\mathcal{G}$ is the unit-distance graph with 5 edges forming two abutting equilateral triangles, then $\mathcal{G}$ is neither globally noncrossing nor globally rigid, since the triangles may be "folded" on top of each other. The *constrained* linkage $\mathcal{M} = (\mathcal{G}, \{\text{NXCon}_{\mathcal{G}}\})$, on the other hand, *is* globally rigid: $\mathcal{M}$ has only one configuration, because the folded (crossing) configuration of $\mathcal{G}$ is rejected by $\mathcal{M}$'s constraint. We have not changed the structure of the linkage, only the lens through which it is viewed.

▶ **Definition 3.2** (Matchstick Linkages)**.** We define an **abstract matchstick linkage** as an NX-constrained abstract unit-distance linkage; a **configured matchstick linkage** additionally comes with a (necessarily unit-edge-length and noncrossing) configuration.

To simulate Theorem 2.8 with matchstick linkages, we first simulate the integer-length edges with **edge polyiamonds**, with **wing edges** inserted along each angle chain, as shown in Figure 2. These wing edges enforce consistent orientation of the edge polyiamonds. To rigidify orthogonal trees, we may brace selected angle chains at 90° with a 5-12-13 right triangle as shown. As in Section 3.1, these assemblies are narrow enough to avoid unplanned crossings.

Matchstick linkages (more generally, NX-constrained linkages) are unique among the three contexts because their traces are not always closed. The simple NX-constrained linkage $\mathcal{A}$ in Figure 3 draws an annulus with one open boundary, because the configurations of the underlying linkage that have v on the inner boundary are crossing and hence excluded by $\mathcal{A}$'s constraint. We prove a stronger universality result: matchstick linkages can draw every bounded semialgebraic set in $\mathbb{R}^2$. Our argument, however, involves our *proof* of Theorem 2.8, not just its statement; we discuss how in Section 4.3.

4 Extended Linkages and the Main Construction

4.1 Defining Extended Linkages

For convenience and clarity, we define and use **extended linkages**, which are constrained linkages whose constraints are tailored for the specifics of our construction. The first of these constraints, the cyclic constraint, specifies a preferred arrangement of edges around each vertex.

▶ **Definition 4.1** (Cyclic Constraint)**.** For an abstract linkage $\mathcal{L}$ with combinatorial embedding σ, a configuration C of $\mathcal{L}$ satisfies the **cyclic constraint** $\text{CyclicCon}_{\mathcal{L}}(\sigma)$ if, for each vertex v with $\sigma_v = [e_1, \ldots, e_{\deg(v)}]$, segments $C(e_1), \ldots, C(e_{\deg(v)})$ intersect only at $C(v)$ and are arranged counterclockwise around $C(v)$ in this order.

▶ **Definition 4.2** (Sliceform Constraint)**.** For a constrained abstract linkage $\mathcal{L}$ possessing a cyclic constraint $\mathrm{CyclicCon}_{\mathcal{L}}(\sigma)$, a **Sliceform Constraint**, $\mathrm{SliceCon}_{\mathcal{L}}(S)$, is specified by a subset $S \subset V(G)$ of (some or all of the) vertices of degree 4. A configuration $C \in \mathrm{Conf}(\mathcal{L})$ (necessarily satisfying $\mathrm{CyclicCon}_{\mathcal{L}}(\sigma)$) satisfies the sliceform constraint $\mathrm{SliceCon}_{\mathcal{L}}(S)$ if, for each **sliceform vertex** $v \in S$, segments $C(e_1)$ and $C(e_3)$ are collinear and $C(e_2)$ and $C(e_4)$ are collinear, where $\sigma_v = [e_1, e_2, e_3, e_4]$.

Sliceforms allow a limited form of "nonplanar" interaction while still being simulatable without crossings (c.f. Figure 5h), so they are our primary tool in circumventing the difficulties of planarity.

▶ **Definition 4.3** (Angle Constraint)**.** If linkage $\mathcal{L}$ has a cyclic constraint $\mathrm{CyclicCon}_{\mathcal{L}}(\sigma)$, an **angle constraint**, $\mathrm{AngleCon}_{\mathcal{L}}(A, \Delta)$, is specified by an assignment of an angle $0 \le A(\Lambda) \le 2\pi$ and an angle tolerance $\Delta(\Lambda) \ge 0$ to each angle chain Λ of $\mathcal{L}$, with the condition that A assigns a total of 2π to the angle chains around each vertex.

A configuration $C \in \mathrm{Conf}(\mathcal{L})$ (necessarily satisfying $\mathrm{CyclicCon}_{\mathcal{L}}(\sigma)$) satisfies the angle constraint $\mathrm{AngleCon}_{\mathcal{L}}(A, \Delta)$ if, for each angle chain Λ, angle $\angle C(\Lambda)$ lies in the closed interval

$$[A(\Lambda) - \Delta(\Lambda), A(\Lambda) + \Delta(\Lambda)].$$

In particular, any angle chain with $\Delta(\Lambda) = 0$ is rigid: its angle in C must be exactly $A(\Lambda)$.

▶ **Definition 4.4** (Extended Linkage)**.** An **(ε, δ)-extended linkage** where $0 < \delta < \varepsilon < \pi/4$ is defined as a constrained linkage $\mathcal{L}$ whose constraints K have the form

$$K = \{\mathrm{CyclicCon}_{\mathcal{L}}(\sigma), \mathrm{SliceCon}_{\mathcal{L}}(S), \mathrm{AngleCon}_{\mathcal{L}}(A, \Delta)\},$$

where at each angle chain Λ of $\mathcal{L}$, $A(\Lambda) \in \{90°, 180°, 270°, 360°\}$ and $\Delta(\Lambda) \in \{0, \delta, \varepsilon\}$. We will call $\mathcal{L}$ simply an **extended linkage** when ε and δ are clear from context.

4.2 Detailed Overview of Strategy

Suppose we are given a finite set F of polynomials in $\mathbb{R}[x_1, y_1, \ldots, x_m, y_m]$. In this section, we discuss how to construct an extended linkage that draws a bounded portion of the common zero set $Z(F)$, i.e., something between $Z(F) \cap [-1, 1]^{2m}$ and $Z(F)$, up to a translation. Our construction uses a transformation to polar coordinates similar to the one used in Kempe's original argument [9] and the corrected construction of Abbott et al. [1]: in place of rectangular coordinates (x_j, y_j), we use angles (α_j, β_j) related by $(x_j, y_j) = 2r \cdot \mathrm{Rect}(\alpha_j, \beta_j)$, where

$$\mathrm{Rect}(\alpha, \beta) := (\cos\alpha, \sin\alpha) + (-\sin\beta, \cos\beta) - (1, 1), \tag{1}$$

where radius $2r$ is carefully chosen. Note that $\mathrm{Rect}(0, 0) = (0, 0)$. We may write this equivalently as

$$x_j = r\left(e^{i\alpha_j} + e^{-i\alpha_j} + ie^{i\beta_j} - ie^{-i\beta_j} - 2\right) \quad y_j = r\left(-ie^{i\alpha_j} + ie^{-i\alpha_j} + e^{i\beta_j} + e^{-i\beta_j} - 2\right). \tag{2}$$

By making this latter substitution into each polynomial $f \in F$, we arrive at a representation of the form

$$f(\overrightarrow{xy}(\overrightarrow{\alpha\beta})) = f(0) + \sum_{u=0}^{3} \sum_{I \in \mathrm{Coeffs}(2m,d)} i^u \cdot d_{u,I} \cdot \left(e^{i\cdot(I\cdot\overrightarrow{\alpha\beta})} - 1\right), \tag{3}$$

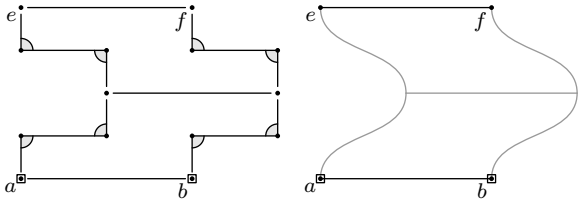


Figure 4 Left: The Parallel Gadget allows e to move freely in a neighborhood of its initial position while forcing ef to remain parallel to ab. Right: a schematic representation of the same gadget.

where $\overrightarrow{\alpha\beta} := (\alpha_1, \beta_1, \ldots, \alpha_m, \beta_m)$, each $d_{u,I}$ is a nonnegative real number, and

$$\text{Coeffs}(2m, d) := \{(a_1, \ldots, a_{2m}) \in \mathbb{Z}^{2m} \mid |a_1| + \cdots + |a_{2m}| \leq d\}.$$

Even though $f(\overrightarrow{xy}(\overrightarrow{\alpha\beta}))$ is real, this complex representation proves more useful for computations below.

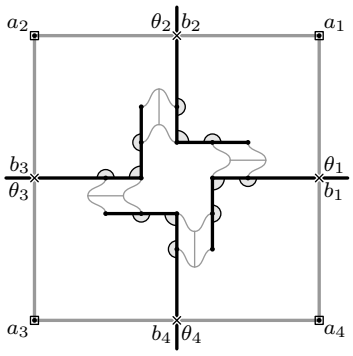
We use this polar representation as a template to compute each polynomial f in the linkage. Indeed, much like in the strategies referenced above, we provide gadgets for the following tasks.

- The Start Gadget (Figure 5d) converts from rectangular position (x_j, y_j) to polar angles (α_j, β_j).
- The Angle Average Gadget (Figure 5c) allows adding and subtracting angles to construct all of the $I \cdot \overrightarrow{\alpha\beta}$ values.
- The Vector Creation Gadget (Figure 5e) and Vector Rotation Gadget (Figure 5f) compute the vectors $i^u \cdot d_{u,I} \cdot e^{i(I \cdot \overrightarrow{\alpha\beta})}$.
- The Vector Average Gadget (Figure 6) allows adding vectors to compute the values $f(\overrightarrow{xy}(\overrightarrow{\alpha\beta})) - f(0)$ for each $f \in F$.
- The End Gadget (Figure 5g) constrains these values to equal $-f(0)$.

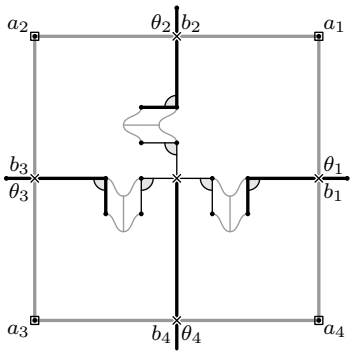
We employ several new ideas to ensure the resulting extended linkage $\mathcal{E}(F)$ is noncrossing. First, we construct a rigid grid of large square cells (with side-length $10R$). Each gadget is isolated in one or a constant number of these cells, and information is passed between gadgets/cells only using sliceform vertices along grid edges. In this way, these modular gadgets may be analyzed individually, as there is no possibility for distinct gadgets to intersect each other. We therefore rely on the Copy Gadget (Figure 5a) to copy angles and propagate them along paths of cells to distant gadgets in the grid. The Crossover Gadget (Figure 5b) allows these paths to cross, so we are not restricted to planar communication between gadgets. These gadgets make frequent use of the Parallel Gadget in Figure 4, which (with pins removed) keeps segments parallel without otherwise restricting motion. Figure 7 shows an example of the gadgets working together.

The linkage $\mathcal{E}(F)$ is an (ε, δ)-extended linkage, where ε and δ (the angle tolerances in $\text{AngleCon}_{\mathcal{L}}(A, \Delta)$) are used in the following manner. The parameter ε constrains bar movement enough to protect against crossings and to ensure uniqueness. By contrast, δ serves (morally) as a lower bound: in each gadget we construct, we ensure that every angle chain with tolerance δ can in fact realize any offset in the entire interval $[-\delta, \delta]$ – this is how we ensure we can draw a large enough portion of $Z(F)$.

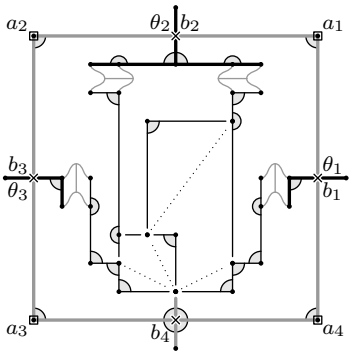
Finally, we simulate linkage $\mathcal{E}(F)$ with a partially rigidified linkage $\mathcal{L}(F)$, in two steps. First, by replacing a vicinity of each sliceform vertex in $\mathcal{E}(F)$ with the Sliceform Gadget (Figure 5h), we construct an extended linkage $\mathcal{E}'(F)$ that perfectly simulates $\mathcal{E}(F)$ but has no



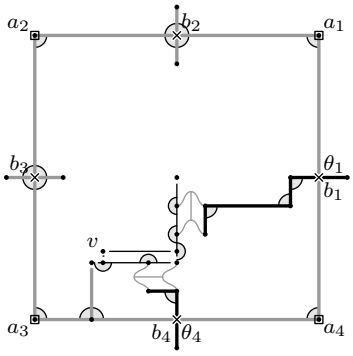
(a) The Copy Gadget forces $\theta_1 = \cdots = \theta_4$.



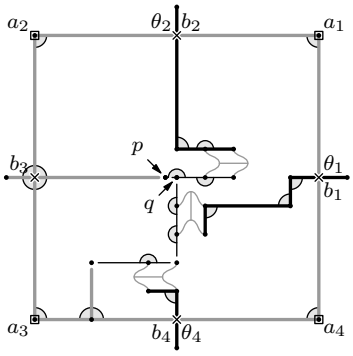
(b) The Crossover Gadget forces $\theta_1 = \theta_3$ and $\theta_2 = \theta_4$.



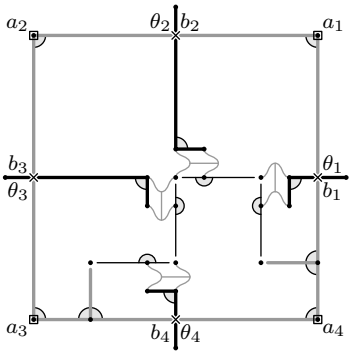
(c) The Angle Average Gadget forces $\theta_2 = (\theta_1 + \theta_3)/2$.



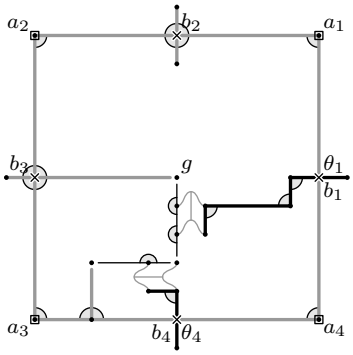
(d) The Start Gadget forces $v = 2r \cdot \mathrm{Rect}(\theta_4 - \frac{\pi}{2}, \theta_1 - \frac{\pi}{2})$.



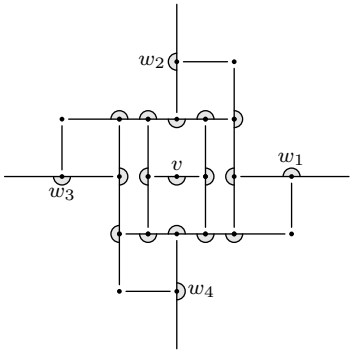
(e) The Vector Creation Gadget forces $R \cdot \mathrm{Rect}(\theta_4 - \frac{\pi}{2}, \theta_1 - \frac{\pi}{2}) = d \cdot (e^{i\theta_2} - 1)$.



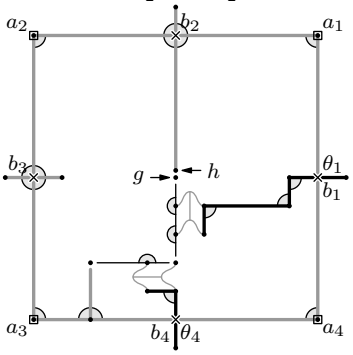
(f) The Vector Rotation Gadget forces $\mathrm{Rect}(\theta_4 - \frac{\pi}{2}, \theta_3 - \frac{\pi}{2}) = i \cdot \mathrm{Rect}(\theta_1 - \frac{\pi}{2}, \theta_2 - \frac{\pi}{2})$.



(g) The End Gadget forces $g = 0$, i.e., $\theta_1 = \theta_4 = \frac{\pi}{2}$.



(h) The Sliceform gadget keeps w_1, v, w_3 and w_2, v, w_4 collinear.



(i) The Crossing End Gadget creates a crossing at $g = h$ when $\theta_1 = \theta_4 = \frac{\pi}{2}$.

Figure 5 Gadgets used in the Main Construction. Angle chains Λ marked with a solid gray sector have $\Delta(\Lambda) = 0$; angle chains at midpoints of cell edges have $\Delta(\Lambda) = \delta$; and the rest have $\Delta(\Lambda) = \varepsilon$ unless otherwise specified. Vertices surrounded by squares are pinned, and those marked with an "x" are sliceform vertices. The pins shown here at the vertices a_i are for clarification only; in the overall construction, these nodes are forbidden from moving by other means, so these explicit pins are unnecessary.

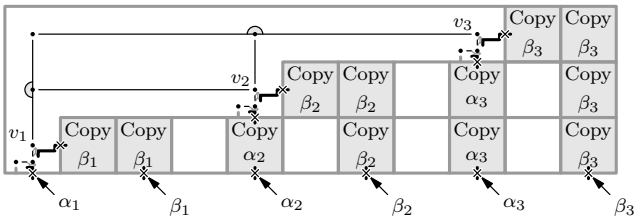


Figure 6 The Vector Average Gadget forces $v_2 = (v_1 + v_3)/2$, i.e., $\text{Rect}(\alpha_2 - \frac{\pi}{2}, \beta_2 - \frac{\pi}{2}) = (\text{Rect}(\alpha_1 - \frac{\pi}{2}, \beta_1 - \frac{\pi}{2}) + \text{Rect}(\alpha_3 - \frac{\pi}{2}, \beta_3 - \frac{\pi}{2}))/2$.

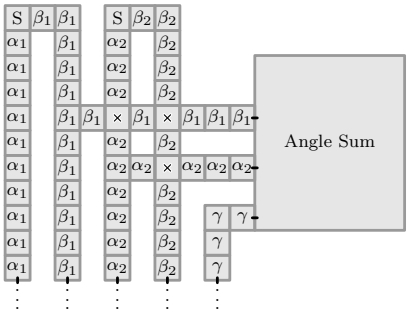


Figure 7 Computing the sum $\gamma = \beta_1 + \alpha_2$. Cells with "S" are start gadgets; those with "x" are crossover gadgets; and those with α_j, β_j, or γ are copy gadgets.

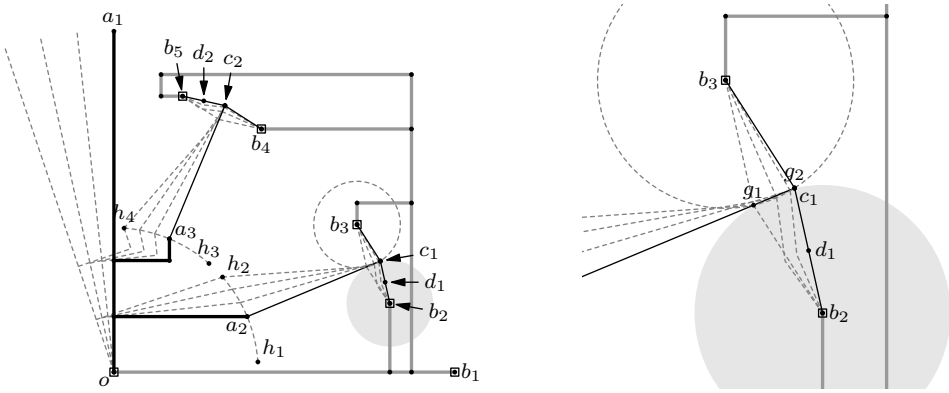


Figure 8 Angle Restrictor Gadget, $\mathcal{L}_{\angle\text{restrict}}$, shown in full (left) and closeup (right).

sliceforms. Then, we replace each edge of $\mathcal{E}'(F)$ with a rigidified orthogonal tree, connected to neighboring edges with the Angle Restrictor Gadget (Figure 8), which exactly enforces the cyclic constraint and the angle constraints.

4.3 Modifications for Strong Matchstick Universality

We may subtly modify the above proof of Theorem 2.8 to prove that the nontrivial subsets of $\mathbb{R}^2$ drawn by matchstick linkages are *exactly* the bounded semialgebraic sets. We use one extra cell gadget when constructing extended linkage $\mathcal{E}(F)$, the Crossing End Gadget (Figure 5i), which is used to create a crossing precisely when $g(\overrightarrow{xy}) = 0$ for a given polynomial g. When linkage $\mathcal{E}(F)$ is simulated by a matchstick linkage $\mathcal{M}(F)$ as described in Section 3.3, all of $\mathcal{E}(F)$'s noncrossing configurations transfer to $\mathcal{M}(F)$, i.e., thickening does not introduce unintended crossings. This allows us to draw semialgebraic sets of the form

$$\{\overrightarrow{x} \in \mathbb{R}^k \in \mathbb{R}^2 \mid f_1(\overrightarrow{x}) = \cdots = f_s(\overrightarrow{x}) = 0, g_1(\overrightarrow{x}) \neq 0, \ldots, g_r(\overrightarrow{x}) \neq 0\},$$

as well as coordinate projections thereof. This is sufficient to draw any bounded semialgebraic set in the plane.

References

1. Timothy Good Abbott. Generalizations of Kempe's Universality Theorem. Master's thesis, Massachusetts Institute of Technology, June 2008. Joint work with Reid W. Barton and Erik D. Demaine. URL: http://web.mit.edu/tabbott/www/papers/mthesis.pdf.
2. Kenneth Appel and Wolfgang Haken. Every planar map is four colorable, Part I: Discharging. *Illinois Journal of Mathematics*, 21:429–490, 1977.
3. Sergio Cabello, Erik D. Demaine, and Günter Rote. Planar embeddings of graphs with specified edge lengths. *Journal of Graph Algorithms and Applications*, 11(1):259–276, 2007.
4. John Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 460–469, Chicago, Illinois, May 1988. URL: http://www.acm.org/pubs/citations/proceedings/stoc/62212/p460-canny/.
5. Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007.
6. Peter Eades and Nicholas C. Wormald. Fixed edge-length graph drawing is NP-hard. *Discrete Applied Mathematics*, 28(2):111–134, August 1990. doi:10.1016/0166-218X(90)90110-X.
7. D. Jordan and M. Steiner. Configuration spaces of mechanical linkages. *Discrete & Computational Geometry*, 22:297–315, 1999.
8. Michael Kapovich and John J. Millson. Universality theorems for configuration spaces of planar linkages. *Topology*, 41(6):1051–1107, 2002. URL: http://arxiv.org/abs/math.AG/9803150.
9. A. B. Kempe. On a general method of describing plane curves of the n^{th} degree by linkwork. *Proceedings of the London Mathematical Society*, 7:213–216, 1876.
10. A. B. Kempe. *How to Draw a Straight Line: A Lecture on Linkages*. Macmillan and co., London, 1877.
11. A. B. Kempe. On the geographical problem of the four colours. *American Journal of Mathematics*, 2:183–200, 1879.
12. Henry C. King. Planar linkages and algebraic sets. *Turkish Journal of Mathematics*, 23(1):33–56, 1999. URL: http://arxiv.org/abs/math.AG/9807023.
13. Pascal Koiran. The complexity of local dimensions for constructible sets. *Journal of Complexity*, 16(1):311–323, March 2000. doi:10.1006/jcom.1999.0536.
14. N. E. Mnev. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In Oleg Viro and Anatoly Vershik, editors, *Topology and Geometry — Rohlin Seminar*, volume 1346 of *Lecture Notes in Mathematics*, pages 527–543. Springer, 1988. doi:10.1007/BFb0082792.
15. Neil Robertson, Daniel Sanders, and Paul Seymour. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70:2–44, 1997. doi:10.1006/jctb.1997.1750.
16. James B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Proceedings of the 17th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, October 1979. URL: https://www.cs.duke.edu/brd/Teaching/Bio/asmb/current/Readings3/saxe-embeddability.pdf.
17. Marcus Schaefer. Realizability of graphs and linkages. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, chapter 23. Springer, 2013. URL: http://ovid.cs.depaul.edu/documents/realizability.pdf.

Finding the Maximum Subset with Bounded Convex Curvature

Mikkel Abrahamsen[*1] **and Mikkel Thorup**[†2]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk

2 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
mikkel2thorup@gmail.com

Abstract

We describe an algorithm for solving an important geometric problem arising in computer-aided manufacturing. When machining a pocket in a solid piece of material such as steel using a rough tool in a milling machine, sharp convex corners of the pocket cannot be done properly, but have to be left for finer tools that are more expensive to use. We want to determine a tool path that maximizes the use of the rough tool. Mathematically, this boils down to the following problem. Given a simply-connected set of points P in the plane such that the boundary ∂P is a curvilinear polygon consisting of n line segments and circular arcs of arbitrary radii, compute the maximum subset $Q \subseteq P$ consisting of simply-connected sets where the boundary of each set is a curve with bounded convex curvature. A closed curve has bounded convex curvature if, when traversed in counterclockwise direction, it turns to the left with curvature at most 1. There is no bound on the curvature where it turns to the right. The difference in the requirement to left- and right-curvature is a natural consequence of different conditions when machining convex and concave areas of the pocket. We devise an algorithm to compute the unique maximum such set Q. The algorithm runs in $O(n \log n)$ time and uses $O(n)$ space.

For the correctness of our algorithm, we prove a new generalization of the Pestov-Ionin Theorem. This is needed to show that the output Q of our algorithm is indeed maximum in the sense that if Q' is any subset of P with a boundary of bounded convex curvature, then $Q' \subseteq Q$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases plane geometry, bounded curvature, pocket machining

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.4

1 Introduction

The motivation for our work comes from the generation of toolpaths for pocket machining. Pocket machining is the process of cutting some specified pocket in a piece of material – in our case most likely a piece of metal – using a milling machine. We first describe the clean mathematical problem that we solve and afterwards explain how it relates to pocket machining.

* Research partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.
† Research partly supported by an Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.

© Mikkel Abrahamsen and Mikkel Thorup;
licensed under Creative Commons License CC-BY

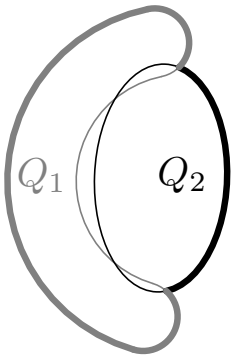


Figure 1

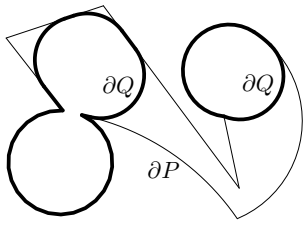


Figure 2

Consider a simply-connected closed subset of the plane and the weakly simple, closed curve around its boundary which we traverse in counter-clockwise direction. We say that the curve is convex where it turns left and concave where it turns right. We say it has *bounded convex curvature* if it turns left with curvature at most 1. There is no bound on the right-curvature, and we may even have sharp concave corners. We say that a set in the plane has bounded convex curvature if all the connected components are simply-connected and the weakly simple, closed curve around the boundary of each connected component has bounded convex curvature. Similarly, we say that a closed curve has *bounded curvature in general* if it turns to the left and to the right with curvature at most 1.

A nice composition property of sets of bounded convex curvature is that if we take two such sets Q_1 and Q_2, unite them, and fill out any holes, then the resulting set, denoted $Q_1 \uplus Q_2$, has bounded convex curvature. See Figure 1, where the boundary of $Q_1 \uplus Q_2$ is the thick curve. This composition property does not hold if we demand that the curvature is bounded in general.

The input to our problem is a simply-connected closed set of points P in the plane such that the boundary ∂P is a curvilinear polygon consisting of n line segments and circular arcs of arbitrary radii. We present an algorithm that in $O(n \log n)$ time finds the unique maximum subset $Q \subseteq P$ of bounded convex curvature, that is, Q contains any other $Q' \subseteq P$ of bounded convex curvature. See Figure 2 for an example.

We note that the uniqueness of a maximal subset Q of bounded convex curvature follows from the composition property; for if there was another $Q' \subseteq P$ of bounded convex curvature that was not contained in Q, then $Q \uplus Q'$ would also be contained in P and have bounded convex curvature, contradicting the maximality of Q.

The boundary of Q will be a curvilinear polygon consisting of $O(n)$ line segments and circular arcs. A very useful property of the boundary of Q is that all concave arcs and vertices are also on the boundary of P. Indeed, it is easy to verify that if there is a concave arc or vertex on the boundary of Q which is not on the boundary of P, then Q is not maximal. A similar reasoning implies that if the boundary of P is a simple curve, then so is the boundary of Q.

We now describe different contexts in which this problem appears naturally. The general problem is that we are given an area S of the plane whose boundary is represented as a curvilinear polygon. There is a thin layer of material in S close to the boundary ∂S of S. The goal is to remove that layer without removing anything from outside S. We are given a rough tool, and we want to remove as much as possible of the thin layer, leaving as little as possible for finer tools that are more expensive to use. The output is a toolpath for the

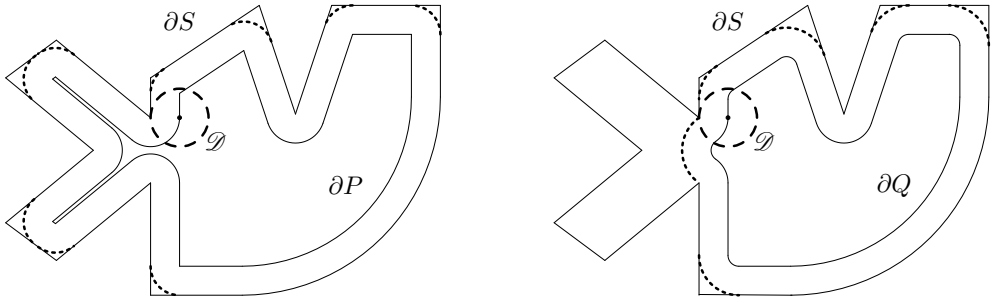


Figure 3 A pocket bounded by ∂S. To the left is shown the boundary ∂P of the inwards offset of S by r. To the right is shown the boundary ∂Q of the maximum subset with bounded convex curvature of P. The dotted arcs in the corners show the boundary of the material in S that cannot be removed by $\mathcal{D}$ using the two toolpaths.

rough tool consisting of one or more curvilinear polygons. The tool is a disk $\mathcal{D}$ of radius r, where r is bigger than the width of the layer we wish to remove. The toolpath is the path which the center of $\mathcal{D}$ is following and the material cut away is the area swept by $\mathcal{D}$ as its center follows the toolpath. The reason that we only have to handle a thin layer close to the boundary of S is because the area farther from the boundary is removed beforehand by tools that are less precise since they do not get close to the boundary. Thus we may assume that the material at all points at some distance $\delta \leq r$ from the boundary have been removed. Some of the points closer to the boundary may also have been removed, but this only makes it easier for our tool to move. With this in mind, when the tool follows a weakly simple closed curve, we think of it as removing the interior of the curve plus every point at distance at most r to the curve.

Let P be the inwards offset of S by r, that is, P is the subset of S of points with distance at least r to the pocket boundary ∂S. See Figure 3. P is the set of all allowed positions of the center of $\mathcal{D}$. If we had complete control over the tool, then we would be able to remove the material in all of P and at all points with distance at most r from P by letting the tool center traverse the boundary ∂P. However, there are restrictions on what tool paths we can follow, e.g., we cannot count on following a toolpath with sharp corners in a precise way.

We are now ready to describe the first application where we want to compute the maximum subset of bounded convex curvature. Assume that the tool can only follow a path which has bounded curvature in general. Assume furthermore that the tool can turn at least as sharply as its own boundary, that is, $r \geq 1$.

Using our algorithm for bounded convex curvature, we are able to identify the maximum area that can be removed by the tool such that the toolpath has bounded curvature in general. First we compute the above set P which is the inwards offset of S by r. The boundary ∂P can be computed from the Voronoi diagram of ∂S [7]. Clearly the toolpath has to stay inside P. We now note that every concave part of ∂P has curvature at most $1/r \leq 1$. Next we use our algorithm to find the maximum subset $Q \subseteq P$ of bounded convex curvature, see Figure 3. The area cut away as $\mathcal{D}$ follows ∂Q is the unique maximum subset that can be cut out of S using a tool with radius $r \geq 1$ and such that the toolpath has bounded convex curvature. However, all concave arcs and vertices on ∂Q stem from P which has concave curvature at most $1/r \leq 1$. It follows that using the toolpath ∂Q, we cut out the maximum subset of S under the condition that the toolpath has bounded curvature in general. This

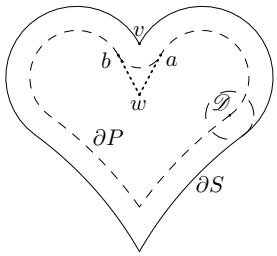


Figure 4

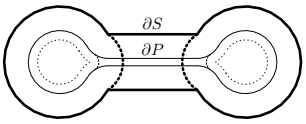


Figure 5

also implies that we remove the maximum subset of the thin layer of material close to the pocket boundary ∂S.

In the above example, the set P had bounded concave curvature. In particular, all concave arcs on ∂P have radius at least r, and for each concave arc A on ∂P of radius r and center v, there is an associated concave vertex v of ∂S. Let a and b be the first and last point on A. When the tool follows A, the corner v will be on the tool boundary $\partial \mathcal{D}$, and the slightest imprecision will blunt the corner v. A recommended alternative [10] is that we substitute A with two line segments aw and wb tangential to A at a and b, respectively, thus creating a sharp concave corner w on the toolpath, see Figure 4. Using this technique, the corner v will be cut much sharper and more precise. One can think of various variations of this technique, since we can "casually" stop and turn the tool at any point on its way to w because the remaining toolpath already ensures that all material will be cut away. This shows that we cannot in general assume that there is any bound on the concave curvature of the input toolpath. We also note the asymmetry with convex corners and arcs, where overshooting a convex corner implies an illegal cut through the boundary of S.

We shall now provide a completely different explanation for the need for bounded convex curvature. The point is that it is often preferable for the surface quality of the resulting part that the tool moves with a constant speed. Recall that the tool is only removing a thin layer close to the pocket boundary. The width of this layer is typically a deliberately chosen small fraction of the tool radius r. When moving at constant speed, a convex turn implies a higher engagement of the tool in the sense of the amount of material removed per time unit. In concave turns the engagement is only decreased. A too high engagement could break the tool, and therefore we must bound the convex curvature of the tool path.

These and other issues related to the machining of corners have been extensively studied in the technical literature on pocket machining. See for instance the papers [5, 6, 9, 11, 14]. There are several previous papers suggesting methods to get bounded convex curvature, but none of them guarantees an optimal solution like ours. One idea for how to handle convex corners is to replace each of them by a convex circular arc as deep in the corner as possible. This is suggested and studied in the papers [5, 9, 11]. However, in all the papers it is assumed that every corner is formed by two line segments which are sufficiently long (relative to the angle between them) that a tangential corner-rounding arc of sufficient size can be placed inside the wedge they form. As can be seen in Figure 3, this is not always the case, and rounding a toolpath can require more complicated modifications.

One heuristic used to obtain a non-trivial subset of bounded convex curvature is the *double offset method*, where we offset P inwards by 1 and then offset the result outwards by 1 and use that as Q. This can be computed in $O(n \log n)$ time using Voronoi diagrams [13]. However, the method does not in general result in the maximum subset of bounded convex

curvature. See for instance Figure 5, where P has bounded convex curvature and all material in S will be machined using ∂P as the toolpath. ∂S is the thick solid curve and ∂P is the thin solid curve. The innermost dotted curves are the boundary of the inwards offset of P by 1. The area of S between the thick dashed arcs will not be machined if the double offset method is used.

In Section 2 we describe an algorithm that computes the maximum subset of bounded convex curvature of P. In Section 3 we describe how to implement the algorithm so that it uses $O(n)$ space and runs in $O(n \log n)$ time. Many proofs and some definitions are omitted due to limited space. We refer to the full version [1].

1.1 General notation and conventions

If M is a set of points in the plane, ∂M denotes the boundary of M and $\overline{M}$ denotes the closure of M. Let γ be a simple curve and x and y two points on γ. Then $\gamma[x, y]$ is the portion of γ between x and y including x and y. Similarly, $\gamma(x, y)$, $\gamma[x, y)$, and $\gamma(x, y]$ are used in the obvious way when none or one of x, y is included. If γ is a closed curve, $\gamma[x, y]$ denotes the portion of γ from x to y in the counterclockwise direction. As before, round parenthesis can be used to exclude one or both endpoints. If γ is not closed, the order of x and y does not matter. We say that a point $z \in \gamma$ is an *inner point* of γ if z is not an endpoint of γ.

1.2 Mathematical foundation: The theorem of Pestov and Ionin

An oriented, simple curve γ is in the class $\mathcal{L}^+$ if γ turns to the left with curvature at most 1, but γ is allowed to turn to the right arbitrarily sharply. There can be corners on γ where γ is not differentiable, but in such corners γ has to turn to the right. These notions are defined rigorously in [1]. Note that a portion of a closed curve with bounded convex curvature traversed in counterclockwise direction is a curve in $\mathcal{L}^+$.

The correctness of the algorithm presented in this paper depends on the following generalization of the Peston-Ionin Theorem. We refer to [1] for a proof.

▶ **Theorem 1.** *Consider an open disk D of arbitrary radius and a curve γ in $\mathcal{L}^+$ from a to b such that $\gamma \cap \overline{D} = \{a, b\}$. Let R be the region bounded by $\partial D[b, a]$ and γ. If R contains D, then R contains an open unit disk U such that there exists a point $c \in \partial U \cap \gamma(a, b)$.*

▶ **Corollary 2.** *Every closed curve γ with bounded convex curvature contains an open unit-disk in its interior.*

The original theorem by Pestov and Ionin [12] was similar to Corollary 2, but γ was assumed to have bounded curvature in general. Howard and Treibergs [8] proved Corollary 2 for closed curves γ with bounded convex curvature for a more restricted class of curves. Ahn et al. [2] proved a theorem similar to Theorem 1, but assuming that γ has bounded curvature in general. Hence, the version of the theorem presented here generalizes all previous versions known to the authors of the present paper.

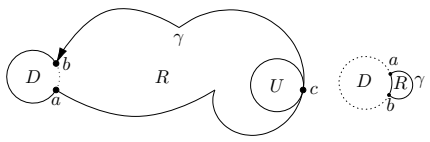


Figure 6 Cases where Theorem 1 does and does not apply.

2 Algorithm

2.1 Preliminaries

We assume that a simply-connected region P_1 in the plane is given such that the boundary ∂P_1 consists of a finite number of line segments and circular arcs of arbitrary radii. For ease of presentation, we assume that ∂P_1 is a simple curvilinear polygon, but our algorithm works as long as ∂P_1 is weakly simple. We can therefore without loss of generality assume that P_1 is an open set. We set $P \leftarrow P_1$ and our algorithm keeps removing parts of P while maintaining the invariant that $\overline{P}$ contains every subset of $\overline{P_1}$ of bounded convex curvature. In the end, $\overline{P}$ itself has bounded convex curvature and it follows that $\overline{P}$ is the unique maximum subset of $\overline{P_1}$ of bounded convex curvature.

The region P is always a collection of disjoint, open, simply-connected sets each of which is bounded by a simple curvilinear polygon. P is represented by its boundary ∂P, which is a collection of disjoint, closed, simple curves where no curve is contained in the interior of another. The input P_1 is defined by one such curve. The open region enclosed by each curve is one connected component of P. ∂P is represented as a set of points known as the *vertices* of ∂P and a set of line segments and circular arcs known as the *arcs* of ∂P. We think of line segments as circular arcs with infinite radius and therefore in most cases use the word *arcs* for both circular arcs and line segments. Depending on the context, we may consider a vertex as a point or a set containing a single point. An *object* of ∂P is a vertex or an arc. We use the convention that an arc includes its endpoints. Every two arcs of ∂P are disjoint except possibly at the endpoints, and for each vertex there are two arcs having an endpoint at that vertex. This way, the arcs form the closed curves bounding P. We always use n to denote the number of vertices of the input ∂P_1.

The boundary of each connected component of P is oriented in counterclockwise direction and we say that a point moving on ∂P is moving *forward* (resp. *backward*) if it is following (resp. not following) the orientation of the boundary. Similarly, we orient every arc following the orientation of the boundary of the component containing it. We denote the endpoints of an arc A as $s(A)$ and $t(A)$, so that a point moving forward along A moves from $s(A)$ to $t(A)$. An arc which turns to the left when traversed in forward direction is called a *convex arc*. A *concave arc* is defined analogously. Line segments are regarded as both concave and convex arcs at the same time. A vertex v is *convex* if the interior angle of ∂P at v is strictly less than π. If the angle is strictly more than π, v is *concave*.

Let A be an arc of ∂P and $a \in A$ a point on A. Then $\mathbf{n}_A(a)$ is the unit-normal of A at a which points to the left relative to the orientation of A. We say that two arcs A and B are *tangential* if $t(A) = s(B)$ and $\mathbf{n}_A(t(A)) = \mathbf{n}_B(s(B))$. Note that A and B are tangential if and only if the vertex $t(A)$ is neither convex nor concave.

2.2 High-level description of the algorithm

A high-level description of the algorithm is given as Algorithm 1. The basic format is to maintain a stack Σ of pointers to objects of ∂P causing the convex curvature condition on ∂P to be violated. An object $\sigma \in \Sigma$ can be a convex vertex, a convex arc of radius less than 1, or a special *cut arc* of radius 1, where one or both of the endpoints might be convex vertices. In each iteration of the loop at line 3, we test if the object σ on the top of Σ is still on ∂P (it might have been removed in another iteration) and if so, we eliminate it by removing from P a subset $\mathcal{V} \subseteq P$. The object σ appears on the boundary $\partial\mathcal{V}$ and not on the boundary of $P \setminus \mathcal{V}$. By *performing a cut* or simply a *cut*, we mean the process of removing $\mathcal{V}$

Algorithm 1: `SubsetOfBoundedConvexCurvature`(P)

```
1 Add all convex arcs of ∂P with a radius less than 1 to Σ.
2 Add all convex vertices of ∂P to Σ.
3 while Σ ≠ ∅
4     Let σ be the topmost element on Σ and remove σ from Σ.
5     if σ ⊂ ∂P and σ is not a perfect cut arc
6         Cut away V from P and let C_j, j = 1, ..., t, be the new cut arcs after the cut.
7         Add each new cut arc C_j to Σ.
8 return P
```

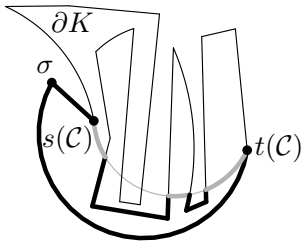


Figure 7

from P. It is important to choose $\mathcal{V}$ such that $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature. Theorem 1 will be used to prove this.

It is possible that a cut splits P into more components. The algorithm will then keep working on each component separately. A cut can introduce new unit-radius cut arcs on ∂P, the endpoints of which can be convex vertices. Therefore, these new cut arcs are added to Σ so that the new convex vertices are eventually removed. The algorithm terminates when Σ is empty, which means that P has bounded convex curvature.

Let P_i be the set P in the beginning of iteration $i = 1, 2, \ldots$ of the loop at line 3 in Algorithm 1. We shall prove that the algorithm terminates after $k = O(n)$ iterations. Hence we have $P_1 \supseteq P_2 \supseteq \cdots \supseteq P_k$. The challenge is to define $\mathcal{V}$ in each iteration so that $k = O(n)$, $P \leftarrow P \setminus \mathcal{V}$ can be computed efficiently, and $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature.

2.3 Specifying a cut using an arc $\mathcal{C}$

Let σ be an object from Σ that is to be eliminated in a certain iteration of Algorithm 1, and let K be the connected component of P such that $\sigma \subset \partial K$. If we conclude that K does not contain any non-empty set of bounded convex curvature, we set $\mathcal{V} = K$, so that all of the component K is removed from P. Otherwise, we specify $\mathcal{V}$ using a circular arc $\mathcal{C}$ of unit radius. See Figure 7, where $\mathcal{C}$ is grey and $\mathcal{V}$ is the area enclosed by the thick closed curve. Assume for now that we have defined $\mathcal{C}$. Let $s(\mathcal{C})$ and $t(\mathcal{C})$ be the points where $\mathcal{C}$ starts and ends in counterclockwise direction, respectively. The endpoints are included in $\mathcal{C}$ and are points on ∂K. They will be defined so that $\sigma \subseteq \partial K[s(\mathcal{C}), t(\mathcal{C})]$. It will follow from the definition of $\mathcal{C}$ that there will be at most two vertices on $\partial K(s(\mathcal{C}), t(\mathcal{C}))$. For efficiency, the algorithm may choose an arc $\mathcal{C}$ that intersects $\partial K(s(\mathcal{C}), t(\mathcal{C}))$.

The arc $\mathcal{C}$ divides K into open regions $R_1, \ldots, R_r$, which are the connected components of $K \setminus \mathcal{C}$. Assume without loss of generality that among all these regions, exactly the regions

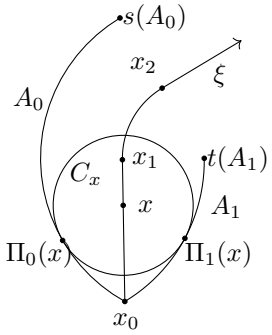


Figure 8

$R_1, \ldots, R_s$, $s \leq r$, contain a part of $\partial K(s(\mathcal{C}), t(\mathcal{C}))$ on their boundary. We note that $s = 1$ if $\mathcal{C}$ does not intersect $\partial K(s(\mathcal{C}), t(\mathcal{C}))$, as is the case in Figure 7. Define $\mathcal{V}$ as the union $\bigcup_{i=1}^{s} K \cap \overline{R_i}$. Then $K \setminus \mathcal{V}$ is open and hence so is our new $P \leftarrow P \setminus \mathcal{V}$. The arc $\mathcal{C}$ will be carefully chosen so that $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature.

▶ **Lemma 3.** *$\mathcal{V}$ is connected and for each $i = 1, 2, \ldots$, the boundary ∂K of each connected component K of P_i is a simple curvilinear polygon.*

Consider the case where we remove a proper subset $\mathcal{V}$ from a connected component K of P_i to obtain P_{i+1}. One or more arcs $\mathcal{C}_1, \ldots, \mathcal{C}_t$ of ∂P_{i+1} are subsets of $\mathcal{C}$ and not arcs of ∂P_i. We denote the arcs $\mathcal{C}_1, \ldots, \mathcal{C}_t$ as *new cut arcs* of ∂P_{i+1}. An arc of ∂P_{i+1} which is a subset of a new cut arc of ∂P_j, $j \leq i$, is not a new cut arc of ∂P_{i+1}. An arc of ∂P_{i+1} is a *cut arc* if it is the subset of a new cut arc of ∂P_j for some $j \leq i + 1$. We say that a cut arc $\mathcal{C}$ of ∂P is *perfect* if none of the endpoints of $\mathcal{C}$ are convex vertices.

▶ **Lemma 4.** *Every cut arc of ∂P is convex.*

2.4 The bisector curve ξ

A specific type of bisector curve, illustrated in Figure 8, turns out to be useful when describing how to define the arc $\mathcal{C}$ specifying a cut and when proving the correctness of our algorithm. The curve is somewhat related to the medial axis of ∂P. Before introducing the curve, we need the concept of an *osculating circle*.

Let A be an arc of ∂P. A circle C *osculates* A at $a \in A$ if the interior of C is disjoint from A, C and A have the point a in common, and the center x of C is a point $a + \mathbf{n}_A(a) \cdot c$ where $c \geq 0$. Recall that $\mathbf{n}_A(a)$ is the unit-normal to A in a pointing to the left. If $c = 0$, C is a single point. Note that a is the point on A closest to x. A special case happens when A is an arc on C, in which case A is a convex arc and C osculates A at every point on A.

▶ **Observation 5.** *Let A be an arc of ∂P and x an arbitrary point. There exists at most one circle C with center x that osculates A. If such a circle C exists, the radius of C is the distance to the closest point a on A from x and C osculates A in a. If A is a convex arc, then the radius of C is at most the radius of A.*

We denote the bisector curve as $\xi = \xi(A_0, A_1, x_0)$. A_0 and A_1 are arcs of ∂P and x_0 is the point where ξ begins. ξ consists of at most three intervals to be defined below, each of which is part of a conic section. Depending on x_0, ξ might start in the second or third interval. From the context where ξ is used, it will be clear in which interval ξ starts.

Often, but not always, A_0 and A_1 are neighboring arcs and x_0 is their common endpoint, i.e., $x_0 = t(A_0) = s(A_1)$. In this case, x_0 will be a convex vertex of ∂P (as in Figure 8) or A_0 and A_1 are tangential. If A_0 and A_1 are tangential, one of A_0 and A_1 will be a cut arc.

For each point $x \in \xi$, we are going to define a circle C_x and points $\Pi_0(x) \in A_0 \cap C_x$ and $\Pi_1(x) \in A_1 \cap C_x$. The circle C_x has its center at x. The functions Π_0 and Π_1 are the *projections* associated to ξ and C_x is the *clearance circle* associated to ξ at x.

Interval **(1)** consists of points x such that C_x osculates both A_0 and A_1 at points $\Pi_0(x)$ and $\Pi_1(x)$, respectively. Interval **(2)** consists of points x where $s(A_0) = \Pi_0(x)$ and C_x osculates A_1 at a point $\Pi_1(x)$ or $t(A_1) = \Pi_1(x)$ and C_x osculates A_0 at a point $\Pi_0(x)$. Interval **(3)** consists of points x such that $s(A_0) = \Pi_0(x)$ and $t(A_1) = \Pi_1(x)$.

Assume that x_0 is contained in interval **(1)**. In the following, we think of x as a point that traverses ξ from x_0.

Interval (1): ξ consists of the points x such that there is a circle C_x with center x that osculates A_0 and A_1 at points $\Pi_0(x)$ and $\Pi_1(x)$, respectively. Hence, this interval consists of points x on ξ such that the distance to the closest points on A_0 and A_1 is the same. In the general case, as x traverses ξ, $\Pi_0(x)$ moves continuously backward and $\Pi_1(x)$ moves continuously forward. We eventually reach a point x_1 where C_{x_1} contains $s(A_0)$ or $t(A_1)$, and ξ continues with the interval defined in **(2)**. A special case occurs if $t(A_0) = s(A_1)$ and A_0 and A_1 are tangential. Then x moves in the direction $\mathbf{n}_{A_0}(t(A_0))$ until x is the center x_1 of A_0 or A_1. This happens since one of the arcs is a cut arc. Until x reaches x_1, the projections are constant, $\Pi_0(x) = \Pi_1(x) = t(A_0)$. If x_1 is the center of A_0, we define $\Pi_0(x_1) = s(A_0)$. Similarly, if x_1 is the center of A_1, we define $\Pi_1(x_1) = t(A_1)$.

Interval (2): ξ has reached a point x_1 such that C_{x_1} contains $s(A_0)$ or $t(A_1)$. If $s(A_0) = t(A_1)$, ξ stops here. Otherwise, if C_{x_1} contains both $s(A_0)$ and $t(A_1)$, interval **(2)** is degenerate, $x_2 = x_1$, and ξ continues with the interval described in **(3)**. Otherwise, assume without loss of generality that C_x contains $s(A_0)$ and osculates A_1. From here, ξ consists of points x such that there is a circle C_x with center x that contains $s(A_0) = \Pi_0(x)$ and osculates A_1 at a point $\Pi_1(x)$. When x moves along ξ, $\Pi_1(x)$ moves continuously forward. At some point, we reach a point x_2 where C_{x_2} contains $s(A_0)$ and $t(A_1)$, and ξ continues with the interval defined in **(3)**.

Interval (3): At the last point x_2 of interval **(2)**, C_{x_2} contains both of the points $s(A_0)$ and $t(A_1)$. Now, ξ consists of the points x such that there is a circle C_x with center x containing both $\Pi_0(x) = s(A_0)$ and $\Pi_1(x) = t(A_1)$, i.e., ξ follows a half-line. We define the direction of the half-line to be the counterclockwise rotation by an angle of $\pi/2$ of the vector $t(A_1) - s(A_0)$.

When defining the arc $\mathcal{C}$ that specifies the cut we want to perform, we are often searching for the first point x_3 on ξ where C_{x_3} has radius 1. We shall then define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$. Yap [13] showed that each of the intervals **(1)**–**(3)** is a part of a conic section. Using elementary geometry we can decide for each interval in $O(1)$ time if it contains a point x_3 such that C_{x_3} has radius 1.

Note that when $x_0 = t(A_0) = s(A_1)$ and x_0 is a convex vertex of ∂P, then a portion of ξ beginning at x_0 is a subset of the medial axis of ∂P. However, x_3 (as defined above) need not be on the medial axis and can even be outside P, also when $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$ becomes a perfect cut arc. See Figure 9 for an example.

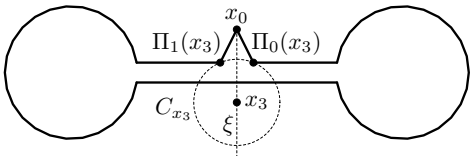


Figure 9

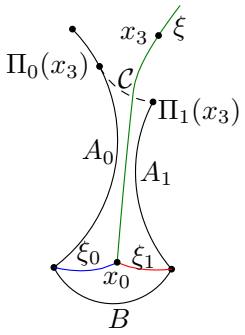


Figure 10

▶ **Observation 6.** *If A_k is a convex arc, $k = 0, 1$, and $x_0 = t(A_0) = s(A_1)$, then for each $a \in A_k$, there exists $x \in \xi$ such that C_x osculates A_k at a, and by Observation 5, the radius of C_x is at most the radius of A_k.*

2.5 Defining the arc $\mathcal{C}$ specifying a cut

The objects in Σ are either **(a)** convex arcs of ∂P_1 with a radius less than 1, **(b)** convex vertices of ∂P_1, or **(c)** cut arcs. Below, we describe how to define $\mathcal{C}$ when the object σ picked by Algorithm 1 is of each of these types.

Case (a): σ is a convex arc with radius less than 1. Consider the curve $\xi = \xi(\sigma, \sigma, x_0)$, where x_0 is the center of σ, which begins in interval **(3)**. Choose the first point $x_3 \in \xi$ such that the associated clearance circle C_{x_3} has radius 1 and define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)] = C_{x_3}[s(\sigma), t(\sigma)]$. Said in another way, we define $\mathcal{C}$ to be the unit-radius arc counterclockwise from $s(\sigma)$ to $t(\sigma)$ which spans an angle of less than π.

Case (b): σ is a convex vertex of ∂P. Let A_0 be the arc of ∂P before σ and A_1 the arc after σ. Consider the curve $\xi = \xi(A_0, A_1, x_0)$, where $x_0 = \sigma$. Find the first point $x_3 \in \xi$ where C_{x_3} has a radius of 1. If such a point does not exist, we are in the case where $s(A_0) = t(A_1)$, and we let $\mathcal{V}$ be the complete component K of P where σ appears on the boundary (in this case, the boundary of K only has two arcs, A_0 and A_1). Otherwise, we define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$.

Case (c): σ is a cut arc B. We assume that one of the endpoints of B is a convex vertex, since otherwise B is perfect and hence ignored by the algorithm. If one of the endpoints, say $s(B)$, is a concave vertex, then $t(B)$ is a convex vertex, and we do as if $\sigma = t(B)$ as described in case **(b)**. We now assume that none of the endpoints of B are concave vertices. See Figure 10. Let A_0 and A_1 be the arcs preceding and succeeding B on ∂P, respectively. Let $\xi_0 = \xi(A_0, B, s(B))$ and $\xi_1 = \xi(B, A_1, t(B))$. For each $b \in B$, follow the ray starting at b with direction $\mathbf{n}_B(b)$ and define $d_k(b)$ to be the distance from b to the first intersection point with ξ_k, $k = 0, 1$. Since $d_0(s(B)) = 0 < d_1(s(B))$ and $d_1(t(B)) = 0 < d_0(t(B))$, the continuity of ξ_k implies there must be an intersection point x_0 between ξ_0 and ξ_1. By Observation 6, we know that the distance from x_0 to B is at most 1. We now consider the curve $\xi = \xi(A_0, A_1, x_0)$ with associated projections Π_0, Π_1, and clearance circle C_x. The clearance circle C_{x_0} of ξ at x_0 is the same as for ξ_0 and ξ_1. If C_{x_0} osculates A_0 and A_1, ξ begins in interval **(1)**. If C_{x_0} osculates one of A_0, A_1 and contains an endpoint of the other,

ξ begins in interval **(2)**. Otherwise, C_{x_0} contains $s(A_0)$ and $t(A_1)$, and ξ begins in interval **(3)**. We find the first point $x_3 \in \xi$ where C_{x_3} has radius 1. If such a point does not exist, we are in the case where $s(A_0) = t(A_1)$, and we let $\mathcal{V}$ be the complete component K of P where σ appears on the boundary. Otherwise, we define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$.

One may object to the algorithm that since a connected component of P can split into more components by a cut, the arcs on Σ are not necessarily well-defined, since subsets of the arc originally added to Σ can be on the boundary of many different connected components of P. However, since the arcs are convex arcs of radius at most 1, the following lemma shows that it is not a problem.

▶ **Lemma 7.** *Let A be an arc of ∂P_1 which is a convex arc of any radius or a concave arc with radius at least 1. For any $i \geq 1$, there exists at most one connected component K of P_i such that (a subset of) A is an arc of ∂K.*

2.6 Proof of correctness of Algorithm 1

In this section, we prove that the set $\mathcal{V}$ to be cut away from P is disjoint from every subset of $\overline{P_1}$ of bounded convex curvature. The proof is by contradiction. The idea is that if there is a subset of $\overline{P_1}$ of bounded convex curvature that overlaps $\mathcal{V}$, then Theorem 1 gives the existence of a unit disk in $\mathcal{V}$ which cannot be there. In general, $\mathcal{V}$ can have a complicated shape and is not easy directly to reason about. Therefore, we describe a superset of $\mathcal{V}$ consisting of *simplices*, which are sufficiently simple that the argument goes through.

A *simplex* S is a closed region and has a boundary ∂S which is a closed, simple curve consisting of two, three, or four circular arcs. One of the arcs D on the boundary is the *door*. The door D has a radius of at most 1 and is clockwise when ∂S is traversed in counterclockwise order. We denote the following as the *simplex condition*, which every simplex is required to satisfy: Let U be the closed disk such that ∂U contains the door D. Then $U \cup S$ does not contain any unit-disk unless U has radius 1, in which case U is the only unit-disk contained in $U \cup S$.

▶ **Lemma 8.** *There exists a set $\mathfrak{S}$ of at most five simplices such that if $\mathcal{S} = \bigcup_{S \in \mathfrak{S}} S$, then $\mathcal{V} \subset \mathcal{S}$ and $\partial\mathcal{S} \subseteq \partial P[s(\mathcal{C}), t(\mathcal{C})] \cup \mathcal{C}$. There is a tree $\mathfrak{T}$ with the nodes being $\mathfrak{S}$ with the following properties. The root of $\mathfrak{T}$ is a simplex with the door $\mathcal{C}$. If a curve $\phi \subset \overline{P}$ exits a simplex $S_0 \in \mathfrak{S}$, it is either through the door of S_0 or through an arc which is the door of a child of S_0 in $\mathfrak{T}$, and in the latter case, ϕ enters the child. If S_0 has a parent in $\mathfrak{T}$ and ϕ exits S_0 through the door of S_0, then ϕ enters the parent.*

An example of the tree $\mathfrak{T}$ is given in Figure 11 for case **(c)** from Section 2.5. The following lemma is the heart of our proof of correctness. Recall that the curves in $\mathcal{L}^+$ are open, oriented curves that turn to the left with curvature at most 1.

▶ **Lemma 9.** *Let γ be a curve in $\mathcal{L}^+$ contained in $\overline{P}$. Let $\mathcal{C}$ be an arc specifying an area $\mathcal{V}$ to be removed from P and suppose that γ starts at a point $a \in \mathcal{C}$. If γ leaves $\overline{\mathcal{V}}$ again, it is through a point on $\mathcal{C}[s(\mathcal{C}), a]$.*

Proof. Assume for contradiction that γ leaves $\overline{\mathcal{V}}$ through a point not as stated in the lemma. Since the boundary of $\mathcal{V}$ is contained in $\partial P \cup \mathcal{C}$, γ leaves $\overline{\mathcal{V}}$ from a point on $\mathcal{C}(a, t(\mathcal{C})]$. Consider the tree $\mathfrak{T}$ as described in Lemma 8. Let S be a simplex with maximum distance in $\mathfrak{T}$ to the root such that γ enters S. Let D be the door of S and $s(D)$ and $t(D)$ be the beginning and end of D in counterclockwise order, respectively. Let $\mathcal{D}$ be the open disk such that D is an arc on the circle $\partial\mathcal{D}$. It follows from Lemma 8 that the first time γ enters S,

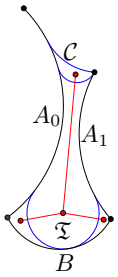


Figure 11

it is through a point on D. Let $e \in D$ be the point closest to $s(D)$ such that γ eventually enters S through e. After e, γ must leave S through a point f on $D(e, t(D)]$, since otherwise γ enters the interior of the closed, simple curve defined by $\gamma[a, e]$, $D[s(D), e]$, $\partial P[s(\mathcal{C}), s(D)]$, and $\mathcal{C}[s(\mathcal{C}), a]$, which it cannot exit again by assumption. Hence, Theorem 1 implies that the region bounded by $\partial\mathcal{D}[f, e]$ and $\gamma[e, f]$ contain a unit disk different from $\mathcal{D}$. That contradicts the simplex condition of S. ◀

By the following lemma, we conclude that when the algorithm terminates, $\overline{P}$ is the maximum subset of $\overline{P_1}$ of bounded convex curvature. The lemma follows easily from Lemma 9.

▶ **Lemma 10.** *For every $i = 1, 2, \ldots$, $\overline{P_i}$ contains Q for every set $Q \subset \overline{P_1}$ of bounded convex curvature.*

2.7 Proof that Algorithm 1 performs a linear number of iterations

Since the set $\mathcal{V}$ that we remove from P is contained in one connected component of P, the algorithm only makes changes to one connected component of P at a time. A vertex v of ∂P is *improper* if v is a convex vertex and v is not a vertex of ∂P_1. An improper vertex can be introduced in two different ways: 1) If the arc $\mathcal{C}$ specifying a cut intersects a cut arc and the intersection point v is on $\partial\mathcal{V}$, then v becomes a convex and hence improper vertex. 2) If the arc $\mathcal{C}$ begins or ends at the endpoint $v \notin \partial P_1$ of a cut arc which is not a convex vertex, v might be a convex and hence improper vertex after the cut has been performed.

Consider the first time an improper vertex appears. Let v be the vertex and let K be the connected component such that $v \in \partial K$. Then v will be removed by the subsequent cut performed in K since a cut arc with an endpoint at v is closest to the top of Σ among all objects in $\Sigma \cap \partial K$. It follows inductively that there can be one or two improper vertices in each component, and after the next iteration performed in the component, none of them will be left.

A vertex v of ∂P which is not a point on ∂P_1 (note the difference between being a *vertex* of ∂P_1 and being a *point* on ∂P_1) can only be created if the arc $\mathcal{C}$ specifying a cut to be performed in P intersects a cut arc at v. It follows that v is a convex and hence improper vertex. Equivalently, every proper vertex is on ∂P_1.

Consider an endpoint c of the arc $\mathcal{C}$ specifying a cut to be performed in P_i, where $P_{i+1} = P_i \setminus \mathcal{V}$. If c is an inner point of an arc A of ∂P_i, Lemma 4 together with Observation 5 gives that A is not a cut arc since the circle containing $\mathcal{C}$ osculates A at c. Hence, A is (a subset of) an arc of ∂P_1 and $c \in \partial P_1$. Furthermore, due to the way $\mathcal{C}$ is defined, an improper vertex of ∂P_i will never be chosen as c. Also in this case, we get that c is on ∂P_1. We summerize these observations in the following lemma.

▶ **Lemma 11.**

1. *There are at most two improper vertices of ∂K for each connected component K of P. If there are two, they are endpoints of the same cut arc.*
2. *All improper vertices of ∂K, where K is a connected component of P, are removed by the following cut performed in K.*
3. *The endpoints $s(\mathcal{C})$ and $t(\mathcal{C})$ of an arc $\mathcal{C}$ specifying a cut are both points on ∂P_1.*

▶ **Lemma 12.** *The algorithm performs $O(n)$ iterations.*

Sketch of proof. We divide the iterations into the types 1–5 defined below and prove that there is a linear number of each type. We bound the iterations of type 1 by observing that each such iteration corresponds to introducing an edge in a plane multigraph with $2n$ vertices, where each pair of vertices can be connected by at most $O(1)$ edges. Hence, Euler's formula implies that $O(n)$ edges are created in total. The iterations of types 2–5 can be bounded by the number of iterations of type 1 or by observing that each iteration completely removes one of the objects of ∂P_1 from ∂P. Hence there are $O(n)$ of each type.

Consider what happens during iterations i when the algorithm picks an object σ from Σ. Let $\mathcal{V}$ be the set removed from P_i during the iteration, i.e., $P_{i+1} = P_i \setminus \mathcal{V}$, where possibly $\mathcal{V} = \emptyset$. Lemma 11 implies that the iteration is of one of the following types.

1. $\mathcal{V}$ is specified by an arc $\mathcal{C}$ and each endpoint of $\mathcal{C}$ is either a vertex of ∂P_1 or an inner point of an arc of ∂P_1 which is not a vertex of ∂P_i.
2. $\mathcal{V}$ is specified by an arc $\mathcal{C}$ and one or both endpoints of $\mathcal{C}$ is a vertex v of ∂P_i which is not a vertex of ∂P_1. Hence, such a vertex v is the endpoint of a cut arc of ∂P_i.
3. A complete connected component K is removed from P, i.e., $\mathcal{V} = K$.
4. $\mathcal{V} = \emptyset$ and nothing happens since σ is not anymore on ∂P.
5. $\mathcal{V} = \emptyset$ and nothing happens since σ is a perfect cut arc.

Consider cuts of type 1. We see that each such cut corresponds to adding an edge to a plane graph G with $2n$ nodes. There is one node in G for each arc and vertex of ∂P_1. We choose a curve ϕ from $s(\mathcal{C})$ to $t(\mathcal{C})$ contained in $\mathcal{V}$ as the edge corresponding to the cut. This is possible since $\mathcal{V}$ is connected by Lemma 3. A vertex v of ∂P_1 is incident to ϕ if v is an endpoint of ϕ. An arc A of ∂P_1 is incident to ϕ if an endpoint of ϕ is an inner point of A. Let A_0 and A_1 be the objects of ∂P_1 incident to ϕ such that $s(\mathcal{C}) \in A_0$ and $t(\mathcal{C}) \in A_1$. Let C be the circle containing $\mathcal{C}$. Observe that if A_0 (resp. A_1) is an arc, then C osculates A_0 at $s(\mathcal{C})$ (resp. A_1 at $t(\mathcal{C})$).

We say that A_0 and A_1 are *parallel* if:

- A_0 and A_1 are concentric arcs, one is concave, the other is convex, and the radius of the convex is 2 larger than the radius of the concave.
- A_0 and A_1 are two parallel line segments (in the ordinary sense) with opposite directions and distance 2.
- A_1 (resp. A_0) is a vertex while A_0 (resp. A_1) is convex arc with radius 2 and center A_1 (resp. A_0).

If A_0 and A_1 are parallel, there are infinitely many unit-circles that osculates A_0 or A_1 like C does. Otherwise, there are only $O(1)$. Hence, by Euler's formula, we are adding $O(n)$ edges to G that connect non-parallel objects. One can verify that even if A_0 and A_1 are parallel, we make $O(1)$ edges between them in G. ◀

Since every cut arc is eventually picked from Σ by Algorithm 1, the linear bound on the number of iterations leads to the following lemma.

Algorithm 2: `PerformCut`$(\mathcal{C})$

```
1  U ← {s(C)}.
2  while U ≠ ∅
3      Remove a point e from U. If e is not a vertex of ∂P, create a vertex at e.
4      a ← e.
5      repeat
6          Let b ← TraverseP(C, a, e). If b is not a vertex of ∂P, create a vertex at b.
7          Mark all objects on ∂P[a, b] as removed, except for vertices a and b.
8          If b ∈ ∂P(s(C), t(C)), add b to U.
9          if C enters or leaves P at b
10             a ← TraverseC(C, b). If a is not a vertex of ∂P, create a vertex at a.
11             Change ∂P by setting the arc succeeding a and preceding b to C[a, b].
12         else
13             a ← e.
14     until a = e
```

▶ **Lemma 13.** *The total number of cut arcs created while running Algorithm 1 is $O(n)$. Likewise, the total number of different vertices appearing on ∂P while running Algorithm 1 is $O(n)$.*

We have now proved the following theorem.

▶ **Theorem 14.** *Given a simply-connected open region P_1 bounded by a curvilinear polygon consisting of n line segments and circular arcs, there is a unique maximum set $Q \subseteq \overline{P_1}$ of bounded convex curvature which contains every set $Q' \subseteq \overline{P_1}$ of bounded convex curvature. ∂Q consists of $O(n)$ line segments and circular arcs, and Algorithm 1 computes Q in $O(n)$ iterations.*

3 Implementation

In this section, we show how the algorithm can be implemented so that it uses $O(n \log n)$ time in total and $O(n)$ space.

Let $\mathcal{C}$ be the arc specifying a cut to be performed in P. Let $R_1, \ldots, R_r$ be the connected components of $P \setminus \mathcal{C}$, and assume without loss of generality that the sets $R_1, \ldots, R_s$, $s \leq r$, have to be removed since their boundaries contain a part of $\partial P(s(\mathcal{C}), t(\mathcal{C}))$. For ease of presentation we assume that $\mathcal{C}$ contains no vertices of ∂P except for possibly the endpoints $s(\mathcal{C})$ and $t(\mathcal{C})$. Algorithm 2 traverses the boundary of each of the regions $R_1, \ldots, R_s$ and introduces the new cut arcs in order to remove the regions from P.

The algorithm traverses $\partial P(s(\mathcal{C}), t(\mathcal{C}))$ from $s(\mathcal{C})$ and removes the regions $R_1, \ldots, R_s$ one by one. Each point in the set U is on the boundary of one of the regions $R_1, \ldots, R_s$ which has not so far been removed. The loop at line 5 traverses one such region.

The subroutine `TraverseP`$(\mathcal{C}, a, e)$ traverses ∂P from a until we meet e or a point where $\mathcal{C}$ enters or exits P. The point b where we stop traversing is returned.

The subroutine `TraverseC`$(\mathcal{C}, b)$ follows $\mathcal{C}$ from b through P until it exits P at some point a, which is returned. Since we assumed that no vertex of ∂P is an inner point of $\mathcal{C}$, it is uniquely defined which direction of $\mathcal{C}$ to follow.

Since P is simply-connected, the regions $R_1, \ldots, R_r$ induce a tree T such that there is an edge in T between two regions R_j and R_k if there is an arc $\mathcal{C}(x, y) \subset \partial R_j \cap \partial R_k$ for $x \neq y$. By Lemma 3, $\mathcal{V}$ is connected, so the regions $R_1, \ldots, R_s$ induce a subtree in T. Therefore, each of the regions in T is traversed at least once by Algorithm 2. On the other hand, a region is removed from P while its boundary is traversed, so each region is traversed at most once. Hence, we have the following lemma.

▶ **Lemma 15.** *Algorithm 2 traverses the boundary of each region $R_1, \ldots, R_s$ exactly once and thus correctly computes $P \leftarrow P \setminus \mathcal{V}$ as specified by the arc $\mathcal{C}$.*

The point $\texttt{TraverseC}(\mathcal{C}, b)$ is the first intersection point between $\mathcal{C}$ and ∂P when following $\mathcal{C}$ from b in the relevant direction. Cheng et al. [4] describes an efficient solution to the following problem. For a simple polygon V, preprocess V such that queries of the following kind can be answered efficiently: Given a circular unit-radius arc beginning at some point in the interior of V, find the first intersection point between the arc and V if it exists. The algorithm requires $O(n)$ space, uses $O(n \log n)$ time on preprocessing, and answers queries in $O(\log n)$ time, where n is the number of vertices of V.

It is straightforward to generalize the method of Cheng et al. to curvilinear polygons. We apply the preprocessing to the original input ∂P_1. Thus, by querying an arc following $\mathcal{C}$ from b in the direction through P, we know the point where $\mathcal{C}$ exits P_1. However, the arc $\mathcal{C}$ can exit P before it exits P_1, namely if and only if it crosses a cut arc of ∂P. In the following, we show how to detect if that is the case or not.

Since $\mathcal{C}$ enters or exits P at b, there exists a point $d \in \mathcal{C} \cap P$ such that $\mathcal{C}(d, b) \subset P$. In the following, d denotes such a point. It is not necessary to compute d, it is merely a tool in our analysis. We know from Lemma 11 that $s(\mathcal{C}), t(\mathcal{C}) \in \partial P_1$. Using the circular ray shooting data structure, we can find a closed subset $\mathcal{C}' \subseteq \mathcal{C}$ such that $s(\mathcal{C}'), t(\mathcal{C}') \in \partial P_1$, $\mathcal{C}'(s(\mathcal{C}'), t(\mathcal{C}')) \subset P_1$, and $b \in \mathcal{C}'$. Hence also $d \in \mathcal{C}'$. The following lemma says that if the arc $\mathcal{C}'$, when traversed in any direction from d, enters a removed area, i.e., a connected component of $P_1 \setminus P$, then it stays in that removed area.

▶ **Lemma 16.** *Let x be one of the endpoints of $\mathcal{C}'$. There is at most one cut arc A of ∂P which intersects $\mathcal{C}'(d, x)$. There is such an arc A if and only if $x \notin \partial P$. If A exists, $\mathcal{C}'(d, x)$ and A intersect at a unique point y, and $\mathcal{C}'[y, x) \subset P_1 \setminus P$.*

It is possible that $\mathcal{C}'(s(\mathcal{C}'), d)$ and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect the same cut arc B. That is easy to detect since we are given a point $b \in \partial P \cap \mathcal{C}'$ where $\mathcal{C}'$ enters of leaves P. Thus, we check if b is on a cut arc B and if $\mathcal{C}(b, x)$ intersects the same arc. Otherwise, every cut arc A satisfies that $A \cap \mathcal{C}'(s(\mathcal{C}), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C})) = \emptyset$, and we have the following lemma.

▶ **Lemma 17.** *Let A be a cut arc of ∂P such that $s(A), t(A) \in \partial P_1$. Assume that $A \cap \mathcal{C}'(s(\mathcal{C}'), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C}')) = \emptyset$. Then, the following two statements hold:*

- *A and $\mathcal{C}'(s(\mathcal{C}'), d)$ intersect if and only if the endpoints of $\mathcal{C}'$ and A appear in the order $s(\mathcal{C}'), t(A), t(\mathcal{C}'), s(A)$ on ∂P_1.*
- *A and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect if and only if the endpoints of $\mathcal{C}'$ and A appear in the order $s(\mathcal{C}'), s(A), t(\mathcal{C}'), t(A)$ on ∂P_1.*

See Figure 12, which illustrates Lemma 17. The black arc A is the only cut arc of ∂P. The thick part of the grey arc $\mathcal{C}$ is $\mathcal{C}'$.

Lemma 17 leads to our method for finding cut arcs intersecting an arc $\mathcal{C}'$ by searching after arcs with endpoints on specific portions of ∂P_1. We associate to each point p on ∂P_1 a unique number $\varphi(p) \in [0, n)$. Let the vertices of ∂P_1 be $v_0, v_1, \ldots, v_{n-1}, v_n$, where $v_0 = v_n$. We set $\varphi(v_i) = i$ for $i < n$. For the points p on the arc between two vertices v_i and v_{i+1}, we

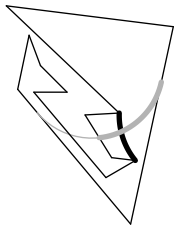


Figure 12

Algorithm 3: `TraverseC`$(\mathcal{C}, b)$

1 Use the circular ray shooting data structure to find the arc $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mathcal{C}'(s(\mathcal{C}'), t(\mathcal{C}')) \subset P_1$, $s(\mathcal{C}'), t(\mathcal{C}') \in \partial P_1$, and $b \in \mathcal{C}'$.
2 Let $z \in \{s(\mathcal{C}'), t(\mathcal{C}')\}$ be the endpoint of $\mathcal{C}'$ when following $\mathcal{C}'$ from b through P.
3 If b is on a cut arc B of ∂P and $\mathcal{C}(b, z)$ intersects B at a point a, return a.
4 Ask the data structure Θ if any point $\theta(A)$ is in the rectangle(s) as specified by Lemma 18. If so, return the intersection point between A and $\mathcal{C}'(b, z)$.
5 Return z.

interpolate between i and $i+1$ to uniquely define $\varphi(p)$. For an arc A with $s(A), t(A) \in \partial P_1$, we define an associated point $\theta(A) \in [0, n) \times (0, 2n)$ in the following way:

$$\theta(A) = \begin{cases} (\varphi(s(A)), \varphi(t(A))) & \text{if } \varphi(s(A)) < \varphi(t(A)) \\ (\varphi(s(A)), \varphi(t(A)) + n) & \text{otherwise.} \end{cases}$$

Lemma 17 then leads to the following.

▶ **Lemma 18.** *Let A be a cut arc of ∂P such that $s(A), t(A) \in \partial P_1$. Assume that $A \cap \mathcal{C}'(s(\mathcal{C}'), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C}')) = \emptyset$. Let $\theta(\mathcal{C}') = (x, y)$.*

- *A and $\mathcal{C}'(s(\mathcal{C}'), d)$ intersect if and only if*
  - *$y < n$ and $\theta(A)$ is in $[0, x) \times (x, y)$ or $(y, n) \times (x + n, y + n)$, or*
  - *$y \geq n$ and $\theta(A)$ is in $(y - n, x) \times (x, y)$.*
- *A and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect if and only if*
  - *$y < n$ and $\theta(A)$ is in $(x, y) \times (y, x + n)$, or*
  - *$y \geq n$ and $\theta(A)$ is in $[0, y - n) \times (y - n, x)$ or $(x, n) \times (y, x + n)$.*

For each cut arc A of ∂P where $s(A), t(A) \in \partial P_1$, we store the point $\theta(A)$ in a data structure Θ. It is necessary to add new points to and delete points from Θ as the algorithm proceeds, since new cut arcs are created and other no longer appear on ∂P. We need to be able to find a point $\theta(A)$ in a rectangle specified by $\mathcal{C}'$ as stated in Lemma 18. Therefore, we implement Θ as a fully dynamic orthogonal range searching structure as described by Blelloch [3]. Algorithm 3 sketches how to implement `TraverseC`.

It is now possible to bound the running time and memory requirement of Algorithm 1 when using our suggested implementation.

▶ **Theorem 19.** *Algorithm 1 can be implemented so that it runs in time $O(n \log n)$ and uses $O(n)$ space.*

References

1. M. Abrahamsen and M. Thorup. Finding the maximum subset with bounded convex curvature, 2016. URL: `http://arxiv.org/abs/1603.02080`.
2. H.-K. Ahn, O. Cheong, J. Matoušek, and A. Vigneron. Reachability by paths of bounded curvature in a convex polygon. *Computational Geometry*, 45(1):21–32, 2012.
3. G.E. Blelloch. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *Proceedings of SODA*, pages 894–903, 2008.
4. S.-W. Cheng, O. Cheong, H. Everett, and R. van Oostrum. Hierarchical decompositions and circular ray shooting in simple polygons. *Discrete and Computational Geometry*, 32:401–415, 2004.
5. H.S. Choy and K.W. Chan. A corner-looping based tool path for pocket milling. *Computer-Aided Design*, 35(2):155–166, 2003.
6. X. Han and L. Tang. Precise prediction of forces in milling circular corners. *International Journal of Machine Tools and Manufacture*, 88:184–193, 2015.
7. M. Held. Voronoi diagrams and offset curves of curvilinear polygons. *Computer-Aided Design*, 30(4):287–300, 1998.
8. R. Howard and A. Treibergs. A reverse isoperimetric inequality, stability and extremal theorems for plane-curves with bounded curvature. *Rocky Mountain Journal of Mathematics*, 25(2):635–684, 1995.
9. H. Iwabe, Y. Fujii, K. Saito, and T. Kishinami. Study on corner cut by end mill. *International Journal of the Japan Society for Precision Engineering*, 28(3):218–223, 1994.
10. S.C. Park and Y.C. Chung. Mitered offset for profile machining. *Computer-Aided Design*, 35(5):501–505, 2003.
11. V. Pateloup, E. Duc, and P. Ray. Corner optimization for pocket machining. *International Journal of Machine Tools and Manufacture*, 44(12):1343–1353, 2004.
12. G. Pestov and V. Ionin. The largest possible circle imbedded in a given closed curve. *Doklady Akademii Nauk SSSR*, 127(6):1170–1172, 1959.
13. C.K. Yap. An $O(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, 2(1):365–393, 1987.
14. Z.Y. Zhao, C.Y. Wang, H.M. Zhou, and Z. Qin. Pocketing toolpath optimization for sharp corners. *Journal of Materials Processing Technology*, 192:175–180, 2007.

Coloring Points with Respect to Squares

Eyal Ackerman[*1], **Balázs Keszegh**[†2], **and Máté Vizer**[‡3]

1 **Department of Mathematics, Physics, and Computer Science, University of Haifa at Oranim, Tivon, Israel**
ackerman@sci.haifa.ac.il

2 **Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
keszegh@renyi.hu

3 **Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary**
vizermate@gmail.com

Abstract

We consider the problem of 2-coloring geometric hypergraphs. Specifically, we show that there is a constant m such that any finite set $\mathcal{S}$ of points in the plane can be 2-colored such that every axis-parallel square that contains at least m points from $\mathcal{S}$ contains points of both colors. Our proof is constructive, that is, it provides a polynomial-time algorithm for obtaining such a 2-coloring. By affine transformations this result immediately applies also when considering homothets of a fixed parallelogram.

1998 ACM Subject Classification G.2.2 Graph Theory – Hypergraphs

Keywords and phrases Geometric hypergraph coloring, Polychromatic coloring, Homothets, Cover-decomposability

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.5

1 Introduction

In this paper we consider the problem of coloring a given set of points in the plane such that every region from a given set of regions contains a point from each color class. To state our results, known results and open problems, we need the following definitions and notations.

A *hypergraph* is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set and $\mathcal{E}$ is a set of subsets of $\mathcal{V}$. The elements of $\mathcal{V}$ and $\mathcal{E}$ are called *vertices* and *hyperedges*, respectively. For a hypergraph $H := (\mathcal{V}, \mathcal{E})$, let $H|_m := (\mathcal{V}, \{e \in \mathcal{E} : |e| \geq m\})$. A *proper coloring* of a hypergraph is a coloring of its vertex set such that in every hyperedge not all vertices are assigned the same color. Proper colorability of a hypergraph with two colors is sometimes called *Property B* in the literature. A *polychromatic k-coloring* of a hypergraph is a coloring of its vertex set with k colors such that every hyperedge contains at least one vertex from each of the k colors.

Given a family of regions $\mathcal{F}$ in $\mathbb{R}^d$ (e.g., all disks in the plane), there is a natural way to define two types of finite hypergraphs that are dual to each other. First, for a finite set of points $\mathcal{S}$, let $H^{\mathcal{F}}(\mathcal{S})$ denote the *primal hypergraph* on the vertex set $\mathcal{S}$ whose hyperedges are

* Most of this work was done during a visit of the first author to the Rényi Institute that was partially supported by Hungarian National Science Fund (OTKA), under grant PD 108406 and by ERC Advanced Research Grant no. 267165 (DISCONV).

† Second author supported by Hungarian National Science Fund (OTKA), under grant PD 108406 and by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

‡ Third author supported by Hungarian National Science Fund (OTKA), under grant SNN 116095.

© Eyal Ackerman, Balázs Keszegh, and Máté Vizer;
licensed under Creative Commons License CC-BY

all subsets of $\mathcal{S}$ that can be obtained by intersecting $\mathcal{S}$ with a member of $\mathcal{F}$. We say that a finite subfamily $\mathcal{F}_0 \subseteq \mathcal{F}$ *realizes* $H^{\mathcal{F}}(\mathcal{S})$ if for every hyperedge $\mathcal{S}' \subseteq \mathcal{S}$ of $H^{\mathcal{F}}(\mathcal{S})$ there is $F' \in \mathcal{F}_0$ such that $F' \cap \mathcal{S} = \mathcal{S}'$. The *dual hypergraph* $H^*(\mathcal{F}_0)$ is defined with respect to a finite subfamily $\mathcal{F}_0 \subseteq \mathcal{F}$. Its vertex set is $\mathcal{F}_0$ and for each point $p \in \mathbb{R}^d$ it has a hyperedge that consists of all the regions in $\mathcal{F}_0$ that contain p.

The general problems we are interested in are the following.

▶ **Problem 1.** *For a given family of regions $\mathcal{F}$,*

(i) *Is there a constant m such that for any finite set of points $\mathcal{S}$ the hypergraph $H^{\mathcal{F}}(\mathcal{S})|_m$ admits a proper 2-coloring?*

(ii) *Is there a constant m^* such that for any finite subset $\mathcal{F}_0 \subseteq \mathcal{F}$ the hypergraph $H^*(\mathcal{F}_0)|_{m^*}$ admits a proper 2-coloring?*

(iii) *Given a constant k, is there a constant m_k such that for any finite set of points $\mathcal{S}$ the hypergraph $H^{\mathcal{F}}(\mathcal{S})|_{m_k}$ admits a polychromatic k-coloring? If so, is $m_k = O(k)$?*

(iv) *Given a constant k, is there a constant m_k^* such that for any finite subset $\mathcal{F}_0 \subseteq \mathcal{F}$ the hypergraph $H^*(\mathcal{F}_0)|_{m_k^*}$ admits a polychromatic k-coloring? If so, is $m_k^* = O(k)$?*

Examples of families $\mathcal{F}$ for which such coloring problems are studied are translates of convex sets [3, 12, 23, 30, 33, 37], homothets of triangles [6, 7, 15, 16, 17, 18], axis-parallel rectangles [11, 28, 26, 10] and half-planes [14, 36]. If $\mathcal{F}$ is the family of disks in the plane then these hypergraphs generalize Delaunay graphs.

The main motivation for studying proper and polychromatic colorings of such geometric hypergraphs comes from cover-decomposability problems and conflict-free coloring problems [35]. We concentrate on the first connection, as the problems we regard are in direct connection with cover-decomposability problems.

Multiple coverings and packings were first studied by Davenport and L. Fejes Tóth almost 50 years ago. Since then a wide variety of questions related to coverings and packings has been investigated. In 1986 Pach [23] published the first paper about decomposability problems of multiple coverings. It turned out that this area is rich of deep and exciting questions, and it has important practical applications as well (e.g., in the area of surveillance systems [12, 25]). Following Pach's papers, most of the efforts were concentrated on studying coverings by translates of some given shape. Recently, many researchers started to study also cover-decomposability of homothets of a given shape.

A family of planar sets is called an *r-fold covering* of a region R, if every point of R is contained in at least r members of the family. A 1-fold covering is simply called a covering. A family $\mathcal{F}$ of planar sets is called *cover-decomposable*, if there is an integer l with the property that for any region R, any subfamily of $\mathcal{F}$ that forms an l-fold covering of R can be decomposed into two coverings. We can generalize the problem of decomposition into more than 2 coverings, in which case we are interested in the existence of a number l_k such that any subfamily of $\mathcal{F}$ that covers every point in R at least l_k times, can be split into k subfamilies, each forming a covering. If we consider only coverings with finite subfamilies, then we call it the *finite cover-decomposition* problem. One of the first observations of Pach was that if $\mathcal{F}$ is the family of translates of an open convex set, then cover-decomposition is equivalent to finite cover-decomposition.

It is easy to see that the finite cover-decomposition problem is equivalent to Problems 1(ii) and (iv) (i.e., $l = m^*$ and $l_k = m_k^*$ in the notation above). Pach also observed that if $\mathcal{F}$ is the family of translates of an open set, then Problems 1(i) and (ii) are equivalent and also Problems 1(iii) and (iv) are equivalent. That is, it is enough to consider the primal hypergraph coloring problem.

Pach conjectured that translates of every open convex planar set are cover-decomposable [22]. During the years researchers acquired a good understanding of convex planar shapes whose translates are cover-decomposable. On the positive side, Pach's conjecture was verified for every open convex polygon: Pach himself proved it for every open centrally symmetric convex polygon [23], then Tardos and Tóth [37] proved the conjecture for every open triangle, and finally Pálvölgyi and Tóth [33] proved it for every open convex polygon. For open convex polygons we also know that $l_k = O(k)$ [3, 12, 30]. In [33] Pálvölgyi and Tóth also gave a complete characterization of cover-decomposable open concave polygons. Thus, the cover-decomposability problem is settled for translates of an open polygon. However, Pach's conjecture was refuted by Pálvölgyi [32] who showed that it does not hold for a disk and for convex shapes with smooth boundary.

In the three dimensional space it follows from the planar results [31, 32] that every bounded polytope is not cover-decomposable. Thus, it is not easy to come up with a cover-decomposable set in the space. An important exception is the *octant*[1], whose translates were proved to be cover-decomposable [15]. The currently best bounds are $5 \leq l \leq 9$ [18] and $l_k = O(k^{5.09})$ [9, 17, 18]. It is an interesting open problem whether $l_k = O(k)$.

For a long time no positive results were known about cover-decomposability and geometric hypergraph coloring problems concerning homothets of a given shape. For disks, the answer is negative for all parts of Problem 1 [24, 27]. As a first positive result, the cover-decomposability of octants along with a simple reduction implied that both the primal and dual hypergraphs with respect to homothets of a triangle are properly 2-colorable:

▶ **Theorem 2** ([15, 18]). *For the family $\mathcal{F}$ of all homothets of a given triangle both Problems* 1(i) *and* 1(iii) *have a positive answer with $m = m^* \leq 9$.*

This result was later used to obtain polychromatic colorings of the primal and dual hypergraphs defined by the family of homothets of a fixed triangle. For the dual hypergraph, the best bound comes from the corresponding result about octants and so it is $m_k^* = O(k^{5.09})$. For the primal hypergraph there is a better bound $m_k = O(k^{4.09})$ [8, 16, 18]. An important tool for obtaining these results is the notion of *self-coverability* (see Section 2.2), which is also essential for proving our results. It is an interesting open problem whether $m_k = O(k)$ and $m_k^* = O(k)$ for the homothets of a given triangle.

For polygons other than triangles, somewhat surprisingly, Kovács [20] recently provided a negative answer for Problems 1(ii) and (iv). Namely, he showed that the homothets of any given convex polygon with at least four sides are not cover-decomposable. In other words, there is no constant m^* for which the dual hypergraph consisting of hyperedges of size at least m^* is 2-colorable. Our main contribution is showing that this is not the case when considering 2-coloring of the primal graph. Indeed, Problem 1(i) has a positive answer for homothets of any given *parallelogram*.

▶ **Theorem 3.** *There is an absolute constant $m_q \leq 1484$ such that the following holds. Given an (open or closed) parallelogram Q and a finite set $\mathcal{S}$ of points in the plane, the points of $\mathcal{S}$ can be 2-colored in polynomial time, such that any homothet of Q that contains at least m_q points contains points of both colors.*

This is the first example that exhibits such different behavior for coloring the primal and dual hypergraphs with respect to the family of some geometric regions. Furthermore, combined with results about self-coverability, the proof of Theorem 3 immediately implies

[1] An octant is the set of points $\{(x, y, z) | x < a, y < b, z < c\}$ in the space for some a, b and c.

the following generalization to polychromatic k-colorings, thus partially answering also Problem 1(iii) (it remains open whether linearly many points per hyperedge/parallelogram suffice).

▶ **Corollary 4.** *Let Q be a given (open or closed) parallelogram and let $\mathcal{S}$ be a set of points in the plane. Then $\mathcal{S}$ can be k-colored, such that any homothet of Q that contains at least $m_k = \Omega(k^{11.6})$ points from $\mathcal{S}$ contains points of all k colors.*

Our proof of Theorem 3 also works for homothets of a triangle, i.e., we give a new proof for the primal case of Theorem 2 (with a larger constant though):

▶ **Theorem 5** ([15])**.** *There is an absolute constant m_t such that the following holds. Given an (open or closed) triangle T and a finite set $\mathcal{S}$ of points in the plane, the points of $\mathcal{S}$ can be 2-colored in polynomial time, such that any homothet of T that contains at least m_t points contains points of both colors.*

This paper is organized as follows. In Section 2 we introduce definitions, notations, tools and some useful lemmas. In Section 3 we describe a generalized 2-coloring algorithm and then apply it for parallelograms and for triangles. Concluding remarks and open problems appear in Section 4. Due to space limitations, some proofs are omitted and can be found in the full version of this extended abstract.

2 Preliminaries

Unless stated otherwise, we restrict ourselves to the two-dimensional Euclidean space $\mathbb{R}^2$. For a point $p \in \mathbb{R}^2$ let $(p)_x$ and $(p)_y$ denote the x- and y-coordinate of p, respectively. We denote by ∂S the boundary of a subset $S \subseteq \mathbb{R}^2$ and by $Cl(S)$ the closure of S. A *homothet* of S is a translated and scaled copy of S. That is, a set $S' = \alpha S + p$ for some number $\alpha > 0$ and a point $p \in \mathbb{R}^2$.

▶ **Lemma 6** (e.g., [21])**.** *Let C be a convex and compact set and let C_1 and C_2 be homothets of C. Then if ∂C_1 and ∂C_2 intersect finitely many times, then they intersect in at most two points.*

2.1 Generalized Delaunay triangulations

For proving Theorems 3 and 5 we will use the notion of generalized Delaunay triangulations, which are the dual of generalized Voronoi diagrams. In the generalized Delaunay triangulation of a point set $\mathcal{S}$ with respect to some convex set C, two points of $\mathcal{S}$ are connected by a straight-line edge if there is a homothet of C that contains them and does not contain any other point of $\mathcal{S}$ in its interior. The generalized Delaunay triangulation of $\mathcal{S}$ with respect to C is denoted by $\mathcal{DT}(C, \mathcal{S})$. We say that $\mathcal{S}$ is *in general position* with respect to (homothets of) C, if there is no homothet of C whose boundary contains four points from $\mathcal{S}$. If $\mathcal{S}$ is in general position with respect to a convex polygon P and no two points of $\mathcal{S}$ define a line that is parallel to a line through two vertices of P, then we say that $\mathcal{S}$ is in *very general position* with respect to P. The following properties of generalized Delaunay triangulations will be useful.

▶ **Lemma 7** ([4, 19, 34])**.** *Let C be a convex set and let $\mathcal{S}$ be a set of points in general position with respect to C. Then $\mathcal{DT}(C, \mathcal{S})$ is a well-defined connected plane graph whose inner faces are triangles.*

It would be convenient to consider generalized Delaunay triangulations in which the boundary of the outer face is a convex polygon. In such a case we say that $\mathcal{DT}(C,\mathcal{S})$ is *nice*.

▶ **Lemma 8.** *Let P be a closed convex polygon and let $\mathcal{S}$ be a set of points in the plane that is in very general position with respect to P. Suppose that P' is a homothet of P and $Z \subseteq \mathcal{S} \cap \partial P'$. Then there is a homothet of P, denote it by P'', such that $P'' \cap \mathcal{S} = (P' \cap \mathcal{S}) \setminus Z$.*

For a homothet C' of a convex set C we denote by $\mathcal{DT}(C,\mathcal{S})[C']$ the subgraph of $\mathcal{DT}(C,\mathcal{S})$ that is induced by the points of $\mathcal{S} \cap C'$. Note that it is not the same as $\mathcal{DT}(C, \mathcal{S} \cap C')$, however the following is true.

▶ **Lemma 9.** *Let P be a closed convex polygon, let $\mathcal{S}$ be a set of points in very general position with respect to P, and let P' be a homothet of P. Then $\mathcal{DT}(P,\mathcal{S})[P']$ is a connected graph that is contained in P'.*

▶ **Corollary 10.** *Let P be a closed convex polygon and let $\mathcal{S}$ be a set of points in very general position with respect to P. Suppose that P' is a homothet of P and e is an edge of $\mathcal{DT}(P,\mathcal{S})$ that crosses $\partial P'$ twice and thus splits P' into two parts. Then one of these parts does not contain a point from $\mathcal{S}$.*

A *rotation* of a vertex v in a plane graph G is the clockwise order of its neighbors. For three straight-line edges vx, vy, vz we say that vy is *between* vx and vz if x, y, z appear in this order in the rotation of v and $\angle xvz < \pi$ ($\angle xvz$ is the angle by which one has to rotate the vector $\vec{vx}$ around v clockwise until its direction coincides with that of $\vec{vz}$). The following will be useful later on.

▶ **Proposition 11.** *Let C be a convex set and let $\mathcal{S}$ be a set of points such that $\mathcal{DT} := \mathcal{DT}(C,\mathcal{S})$ is nice. Let C' be a homothet of C and let v be a vertex in $\mathcal{DT}[C']$. Suppose that x and z are two vertices such that z immediately follows x in the rotation of v in $\mathcal{DT}[C']$, $\angle xvz < \pi$ and $xz \notin \mathcal{DT}$. Then there exists an edge $vy \in \mathcal{DT}$ between vx and vz (which implies that $y \notin C'$).*

Proof. Suppose that x and z are also consecutive in the rotation of v in $\mathcal{DT}$. Then the face that is incident to vx and vz and is to the right of $\vec{vx}$ and to the left of $\vec{vz}$ cannot be the outer face since $\angle xvz < \pi$ and $\mathcal{DT}$ is nice. However, since this face is an inner face, then by Lemma 7 it must be a triangle and so $xz \in \mathcal{DT}$. ◀

2.2 Self-coverability of convex polygons and polychromatic k-coloring

Keszegh and Pálvölgyi introduced in [16] the notion of *self-coverability* and its connection to polychromatic k-coloring. In this section we list the definition and results from their work that we use.

▶ **Definition 12** ([16])**.** A collection of closed sets $\mathcal{F}$ in a topological space is *self-coverable* if there exists a *self-coverability function* f such that for any set $F \in \mathcal{F}$ and for any finite point set $\mathcal{S} \subset F$, with $|\mathcal{S}| = l$ there exists a subcollection $\mathcal{F}' \subset \mathcal{F}$, $|\mathcal{F}'| \leq f(l)$ such that $\bigcup_{F' \in \mathcal{F}'} F' = F$ and no point of $\mathcal{S}$ is in the interior of some $F' \in \mathcal{F}'$.

▶ **Theorem 13** ([16])**.** *For every convex polygon P there is a constant $c_f := c_f(P)$ such that the family of all homothets of P is self-coverable with $f(l) \leq c_f l$.*

▶ **Theorem 14** ([16])**.** *The family of all homothets of a square is self-coverable with $f(l) := 2l + 2$ and this is sharp.*

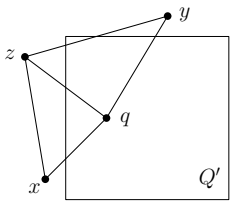


Figure 1 Considering homothets of an axis-parallel square, x-q-y is a good 2-path whereas x-q-z is not since the square Q' separates it and both qx and qz cross the left side of Q'.

▶ **Theorem 15** ([16]). *The family of all homothets of a given triangle is self-coverable with* $f(l) := 2l + 1$ *and this is sharp.*

▶ **Theorem 16** ([16, Theorem 2]). *If* $\mathcal{F}$ *is self-coverable with a monotone self-coverability function* $f(l) > l$ *and any finite set of points can be colored with two colors such that any member of* $\mathcal{F}$ *with at least* m *points contains both colors, then any finite set of points can be colored with* k *colors such that any member of* $\mathcal{F}$ *with at least* $m_k := m(f(m-1))^{\lceil \log k \rceil - 1} \leq k^d$ *points contains all* k *colors (where* d *is a constant that depends only on* $\mathcal{F}$*).*

Theorems 3 and 16 immediately imply Corollary 4.

3 A 2-coloring algorithm

In this section we prove Theorems 3 and 5. In fact, we prove a more general result, for which we need the following definitions.

▶ **Definition 17** (Good paths and good homothets). Let P be an (open or closed) convex polygon, let $\mathcal{S}$ be a finite set of points, let $\mathcal{DT} := \mathcal{DT}(P, \mathcal{S})$, and let P' be a homothet of P.

- Let x-y-z be a 2-path in $\mathcal{DT}$ (i.e., a simple path of length two). If P' contains y and does not contain x and z, then we say that it *separates* the 2-path x-y-z.
- A 2-path x-y-z is *good*, if there is no homothet of P that separates it such that the edges yx and yz cross the same side of this homothet of P (see Figure 1 for an example).
- A 3-path x-y-z-w in $\mathcal{DT}$ is *good* if both x-y-z and y-z-w are good 2-paths.
- P' is *good* if it contains a good 3-path or $\mathcal{DT}[P']$ contains a cycle.

▶ **Definition 18** (Universally good polygons). We say that an (open or closed) polygon P is *universally good* with a constant $c_g := c_g(P)$ if for any finite set of points $\mathcal{S}$ such that $\mathcal{S}$ is in very general position with respect to P and $\mathcal{DT}(P, \mathcal{S})$ is nice, every homothet of P that contains at least c_g points from $\mathcal{S}$ is good.

▶ **Theorem 19.** *Let* P *be an (open or closed) convex polygon with* n *vertices such that* P *is a universally good polygon with a constant* $c_g := c_g(P)$*, and let* $f(l) \leq c_f l$ *be a self-coverability function of the family of homothets of* $Cl(P)$ *(where* c_f *is a constant). Then there is a constant* $m := m(P) \leq c_g f(n) + n$ *such that it is possible to 2-color in polynomial time the points of any given finite set of points* $\mathcal{S}$ *such that every homothet of* P *that contains at least* m *points from* $\mathcal{S}$ *contains points of both colors.*

Theorems 3 and 5 immediately follow from Theorems 14, 15, 19, and the following.

▶ **Lemma 20.** *Every triangle is a universally good polygon with a constant* $c_g \leq 7382$.

▶ **Lemma 21.** *Every parallelogram is a universally good polygon with a constant* $c_g \leq 148$.

3.1 Proof of Theorem 19

Let P be an (open or closed) convex polygon with n vertices such that P is a universally good polygon with a constant $c_g := c_g(P)$, and let $f(l) \leq c_f l$ be a self-coverability function of the family of homothets of $Cl(P)$ (where c_f is a constant). Set $m := c_g f(n) + n$. We first argue that it is enough to prove Theorem 19 when P is a closed polygon. Indeed, suppose that P is open, let $\mathcal{P}$ be the family of homothets of P and let $\mathcal{P}_0 \subseteq \mathcal{P}$ be a finite subfamily that realizes $H^{\mathcal{P}}(\mathcal{S})$. By slightly shrinking every homothet of P in $\mathcal{P}_0$ with respect to an interior point, we get a subfamily $\mathcal{P}_0' \subseteq \mathcal{P}$ that realizes $H^{\mathcal{P}}(\mathcal{S})$ such that there is no $p \in \mathcal{S}$ and $P' \in \mathcal{P}_0'$ with $p \in \partial P'$. Let $\bar{P} := Cl(P)$ be the closed polygon that is the closure of P. Note that by definition $\bar{P}$ is universally good with the same constant c_g and is self-coverable with the same self-coverability function. Let $\bar{\mathcal{P}}$ be the family of homothets of $\bar{P}$, and let $\bar{\mathcal{P}}_0' \subseteq \bar{\mathcal{P}} := \{Cl(P') | P' \in \mathcal{P}_0'\}$. Since there is no homothet of P in $\mathcal{P}_0'$ that contains a point of $\mathcal{S}$ on its boundary, every hyperedge of $H^{\mathcal{P}}(\mathcal{S})$ appears also in $H^{\bar{\mathcal{P}}}(\mathcal{S})$. Thus, it is enough to show that $\bar{P}$ satisfies Theorem 19.

Suppose therefore that P is a closed convex polygon. Let $\mathcal{P}$ be the family of homothets of P and let $\mathcal{P}_0 \subseteq \mathcal{P}$ be the smallest subfamily that realizes $H^{\mathcal{P}}(\mathcal{S})$. For convenience we pick $\mathcal{P}_0$ such that every $P' \in \mathcal{P}_0$ is minimal in the sense that it does not contain any other homothet of P that contains the same set of points from $\mathcal{S}$. We may also assume that $\mathcal{S}$ is in very general position with respect to P. Indeed, otherwise note that a small perturbation of the points will achieve that and observe also that for every homothet $P' \in \mathcal{P}_0$ a slightly inflated P' will contain the (perturbed) points that correspond to the points in $\mathcal{S} \cap P'$ and no other points. After a perturbation every homothet in $\mathcal{P}_0$ is 'almost' minimal, which is fine for our purposes. It will also be convenient to assume that after the perturbation, the boundaries of every two polygons in $\mathcal{P}_0$ do not overlap, and no edge in $\mathcal{DT} := \mathcal{DT}(P, \mathcal{S})$ crosses the boundary of a polygon in $\mathcal{P}_0$ at one of its vertices. It follows from Lemma 6 that $\mathcal{P}_0$ is a family of *pseudo-disks*.[2] This implies that $|\mathcal{P}_0| = O(|\mathcal{S}|^3)$ by a result of Buzaglo et al. [5] who proved the following: Suppose that $(\mathcal{V}, \mathcal{E})$ is a hypergraph where $\mathcal{V}$ is a set of points in the plane and for every hyperedge $e \in \mathcal{E}$ there is a region bounded by a simple closed curve that contains the points of e and no other points from $\mathcal{V}$. If all the regions that correspond to $\mathcal{E}$ define a family of pseudo-disks, then $|\mathcal{E}| = O(|\mathcal{V}|^3)$.

We can also assume that $\mathcal{DT}$ is nice, that is, the boundary of its outer face is a convex polygon: Set $-P := \{(-x, -y) \mid (x, y) \in P\}$ and let $-P'$ be a homothet of $-P$ that contains in its interior all the polygons in $\mathcal{P}_0$. By adding the vertices of $-P'$ to $\mathcal{S}$ (and perturbing again if needed) we obtain a set of points such that $-P'$ is the boundary of the outer face in its generalized Delaunay triangulation with respect to P. Moreover, we have only extended the set of point subsets $\mathcal{S}'$ that contain at least m points and for which there is a homothet $P' \in \mathcal{P}_0$ such that $\mathcal{S}' \cap P' = \mathcal{S}'$. Therefore a valid 2-coloring of the new set of points induces a valid 2-coloring of the original set of points.

Recall that $\mathcal{DT}$ is a plane graph, and therefore, by the Four Color Theorem, we can color the points in $\mathcal{S}$ with four colors, say $1, 2, 3, 4$, such that there are no adjacent vertices in $\mathcal{DT}$ with the same color. In order to obtain two color classes, we recolor all the vertices of colors 1 or 2 with the color *light red* and all the vertices of colors 3 or 4 with the color *light blue*.

Call a homothet $P' \in \mathcal{P}_0$ *heavy monochromatic* if it contains exactly c_g points from $\mathcal{S}$ and all of them are of the same color. If all of these points are colored light blue (resp., red), then P' a heavy light blue (resp., red) homothet. Obviously, if there are no heavy monochromatic

[2] In a family of pseudo-disks the boundaries of every two regions cross at most twice.

homothets, then we are done since $m > c_g$ and by Lemma 8 a monochromatic homothet with $m > c_g$ points from $\mathcal{S}$ can be shrinked to a monochromatic homothet with exactly c_g points from $\mathcal{S}$. Suppose that P' is a heavy monochromatic homothet of P. Observe that $\mathcal{DT}[P']$ is a tree, for otherwise it would contain a cycle which in turn would contain a triangle by Lemma 7. That triangle must be 3-colored in the initial 4-coloring, so not all of its points can be light red or light blue, contradicting the monochromaticity of the points in P'.

Since P is universally good, P' contains c_g points and $\mathcal{DT}[P']$ is a tree, it follows that P' contains a good 3-path x-y-z-w. We associate this 3-path with P'. Suppose that P' is a heavy light red homothet of P. Then one of y and z was originally colored 1 and the other was originally colored 2. Recolor the one whose original color was 1 with the color *dark blue*. Similarly, if P' is a heavy light blue homothet of P, then one of y and z was originally colored 3 and the other was originally colored 4. In this case we recolor the one whose original color was 3 with the color *dark red*. Repeat this for every heavy monochromatic homothet, and, finally, in order to obtain a 2-coloring, merge the color classes light red and dark red into one color class – red, and merge the color classes light blue and dark blue into one color class – blue.

▶ **Lemma 22.** *There is no homothet $P' \in \mathcal{P}_0$ that contains m points from $\mathcal{S}$ all of which of the same color.*

Proof. Suppose for contradiction that P' is a homothet of P that contains m points from $\mathcal{S}$ all of which of the same color. We may assume without loss of generality that all the points in P' are colored red, therefore, before the final recoloring each point in P' was either light red or dark red. Recall that n is the number of vertices of P. We consider two cases based on the number of dark red points in P'.

Case 1: There are at most n dark red points in P'

By Theorem 13 there is a set $\mathcal{P}'$ of at most $f(n)$ homothets of P whose union is P' such that no dark red point in P' is in the interior of one of these homothets. Using Lemma 8 we can change these homothets slightly such that none of them contains a dark red point yet all light red points are still covered by these homothets. Thus the at least $m - n = c_g f(n)$ light red points are covered by these at most $f(n)$ homothets. By the pigeonhole principle one of these homothets, denote it by P'', contains at least $\frac{c_g f(n)}{f(n)} = c_g$ light red points and no other points. However, in this case it follows from Lemma 8 that there is a heavy light red homothet in $\mathcal{P}_0$ that contains exactly c_g points from $\mathcal{S} \cap P''$. Therefore, the coloring algorithm should have found within this heavy light red homothet a good 3-path and recolored one of its vertices with dark blue and then blue. This contradicts the assumption that all the points in P' are red.

Case 2: There are more than n dark red points in P'

Let y be one of these dark red points. Then there is a good 3-path x-y-z-w within a heavy light blue homothet $P_y \in \mathcal{P}_0$ with whom this 3-path is associated. Furthermore, the original color of y is 3 and therefore the original color of x and z is 4, and thus their final color is blue. It follows that P' separates x-y-z, moreover, since x-y-z is a good 2-path, the edges yx and yz cross different sides of P'. Let s_x (resp., s_z) be the side of P' that is crossed by yx (resp., yz), and let q_x (resp., q_z) be the crossing point of yx and s_x (resp., yz and s_z). See Figure 2.

Note that $\partial P'$ and ∂P_y cross each other exactly twice. Indeed, this follows from Lemma 6 and the fact that there are points from $\mathcal{S}$ in each of $P' \cap P_y$ (e.g., y), $P_y \setminus P'$ (e.g., x and z)

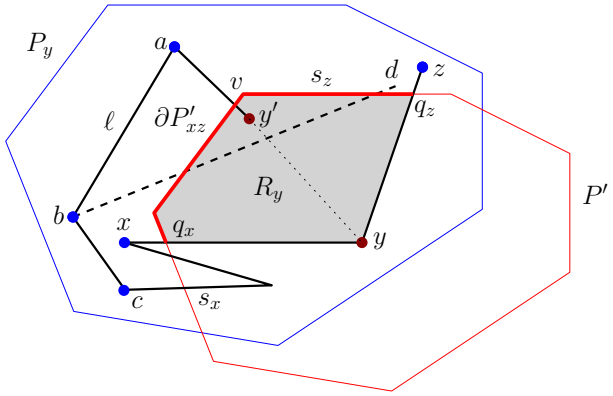


Figure 2 An illustration for the proof of Lemma 22.

and $P' \setminus P_y$ (since $|P' \cap \mathcal{S}| \geq m > c_g = |P_y \cap \mathcal{S}|$). The points q_x and q_z partition $\partial P'$ into two parts $\partial P'_1$ and $\partial P'_2$. Note that since $q_x, q_z \in P' \cap P_y$, the two crossing points between $\partial P'$ and ∂P_y must lie either in $\partial P'_1$ or in $\partial P'_2$. Assume without loss of generality that both of them lie in $\partial P'_1$. Thus $\partial P'_2 \subset P_y$. Let v be a vertex of P' in $\partial P'_2$ (note that since $s_x \neq s_z$ each of $\partial P'_1$ and $\partial P'_2$ contains a vertex of P'). We associate the vertex v with the dark red point y. We also define R_y to be the region whose boundary consists of the segment yq_x of yx, the segment yq_z of yz, and the part of $\partial P'_2$ whose endpoints are q_x and q_z (call this part $\partial P'_{xz}$). Observe that $R_y \subseteq P' \cap P_y$.

▶ **Proposition 23.** *There is no other point but y in $\mathcal{S} \cap R_y$.*

Proof. Suppose that the claim is false and let $y' \in \mathcal{S} \cap R_y$ be another point in R_y. As y' is in P', it must be red after the final coloring. Also, as it is also in P_y, it must be a dark red point (which was light blue before having been recolored to dark red and finally to red). Thus, y' is a dark red point in R_y.

Since x and y' both lie in the heavy light blue homothet P_y, they are connected by a path in $\mathcal{DT}[P_y]$ that alternates between points of colors 3 and 4 (considering the initial 4-coloring). We may assume without loss of generality that y' is the first point in R_y along this path from x to y': indeed, there are no points of color 4 in R_y, and if there is point of color 3 before y', then we can name it y'. Denote by ℓ the path (in $\mathcal{DT}$) from y to y' that consists of the edge yx and the above-mentioned path from x to y'. Consider the polygon $\hat{P}$ whose boundary consists of ℓ and a straight-line segment yy' ($\hat{P}$ is not a homothet of P). Since y' and y are the only vertices of $\hat{P}$ in R_y, there is no edge of ℓ that crosses yy'. Indeed, if there was such an edge then it would split P' into two parts such that one contains y and the other contains y'. This would contradict Corollary 10. Hence $\hat{P}$ is a simple polygon.

Since every simple polygon has at least three convex vertices, $\hat{P}$ has a convex vertex different from y and y' (thus this vertex is not in R_y). Denote this vertex by b and let a and c be its neighbors along ℓ such that $\angle abc < \pi$. Observe that since $\angle abc < \pi$ it is impossible that the outer face of $\mathcal{DT}$ is incident to a, b, c and lies inside $\hat{P}$. Therefore, since the initial color of a and c is the same (thus $ac \notin \mathcal{DT}$), it follows from Proposition 11 that there is a neighbor of b in $\mathcal{DT}$ in between a and c. Furthermore, it is not hard to see that there is such a neighbor d whose initial color is 1 or 2. Note that $\hat{P} \subseteq P_y$ since all of its edges are inside P_y. Thus $d \notin \hat{P}$ and also $d \notin P_y$ since P_y does not contain vertices of color 1 or 2. Now consider the directed edge bd: it starts inside $\hat{P}$ (since d is in between a and c) and so it must cross yy'. Before doing so bd must cross ∂R_y and so it crosses $\partial P'_{xz}$, since it cannot cross yq_z

or yq_x. After crossing yy', the directed edge bd must cross $\partial P'_{xz}$ again, since $d \notin R_y$. But then bd splits P' into two parts such that one contains y and the other contains y', which is impossible by Corollary 10. ◀

In a similar way to the one described above, we associate a vertex of P' with every dark red point in P'. Since there are more than n dark red points in P', there are two of them, denote them by y and y', that are associated with the same vertex of P', denote it by v. Let x-y-z (resp., x'-y'-z') be the good 2-path that corresponds to y (resp., y') as above. Let yq_x and yq_z (resp., $yq_{x'}$ and $yq_{z'}$) be the edge-segments of yx and yz (resp., $y'x'$ and $y'z'$) as above, and let R_y (resp., $R_{y'}$) be the region as defined above.

It follows from Proposition 23 that $y \notin R_{y'}$ and $y' \notin R_y$. However, ∂R_y and $\partial R'_y$ both contain v. This implies that one of the segments yq_x and yq_z crosses one of the segments $y'q_{x'}$ and $y'q_{z'}$, which is impossible since these are segments of edges of a plane graph. Lemma 22 is proved. ◀

To complete the proof of Theorem 19, we need to argue that the described algorithm runs in polynomial time. Indeed, constructing the generalized Delaunay triangulation and then 4-coloring it can be done in polynomial time. Recall that there are at most $O(|\mathcal{S}|^3)$ combinatorially different homothets of P. Among them, we need to consider those that contain exactly c_g points, and for each such heavy monochromatic homothet P' we need to find a good 3-path in $\mathcal{DT}[P']$, for the final recoloring step. This takes a constant time for every heavy monochromatic homothet, since c_g is a constant. Therefore, the overall running time is polynomial with respect to the size of $\mathcal{S}$. ◀

3.2 Parallelograms are universally good

In this section we prove Lemma 21. Let Q be a parallelogram, let $\mathcal{S}$ be a set of points in very general position with respect to Q, and let $\mathcal{DT} := \mathcal{DT}(Q, \mathcal{S})$ be the generalized Delaunay triangulation of $\mathcal{S}$ with respect to Q such that $\mathcal{DT}$ is nice (i.e., the boundary of its outer face is a convex polygon). By applying an affine transformation, we may assume without loss of generality that Q is an axis-parallel square. Since $\mathcal{S}$ is in very general position, no two points in $\mathcal{S}$ share the same x- or y-coordinate.

Suppose that Q' is a homothet of Q that contains at least 148 points from $\mathcal{S}$ and that $\mathcal{DT}[Q']$ is a tree. We will show that Q' contains a good 3-path.

Let $q \in \mathcal{S}$ be a point. We partition the points of the plane into four open quadrants according to their position with respect to q: NE(q) (North-East), NW(q) (North-West), SE(q) (South-East), and SW(q) (South-West).

▶ **Proposition 24.** *For every point $q \in \mathcal{S} \cap Q'$ there are no two neighbors of q in $\mathcal{DT}[Q']$ that lie in the same quadrant of q.*

▶ **Proposition 25.** *Let x and y be two neighbors of q in $\mathcal{DT}[Q']$. Let $z \notin Q'$ be a neighbor of q in $\mathcal{DT}$ that lies between x and y in the rotation of q and let Q_z be a square that contains q and z and no other point from $\mathcal{S}$. Then:*

- *if $x \in$ NW(q) and $y \in$ NE(q), then qz crosses the top side of Q', x is to the left of Q_z and y is to the right of Q_z;*
- *if $x \in$ NE(q) and $y \in$ SE(q), then qz crosses the right side of Q', x is above Q_z and y is below Q_z;*
- *if $x \in$ SE(q) and $y \in$ SW(q), then qz crosses the bottom side of Q', x is to the right of Q_z and y is to the left of Q_z; and*

- *if* $x \in \mathrm{SW}(q)$ *and* $y \in \mathrm{NW}(q)$*, then* qz *crosses the left side of* Q'*,* x *is below* Q_z *and* y *is above* Q_z*.*

Call a (simple) path in $\mathcal{DT}$ *x-monotone* (resp., *y*-monotone) if there is no vertical (resp., horizontal) line that intersects the path in more than one point.

▶ **Proposition 26.** *Every path in* $\mathcal{DT}[Q']$ *is x-monotone or y-monotone.*

Proof. Suppose for contradiction that there is a path $p := q_1\text{-}q_2\text{-}\ldots\text{-}q_k$ which is neither x-monotone nor y-monotone. Since p is a polygonal path, it follows that there are two points, q_i and q_j, that are "witnesses" to the non-x- and non-y-monotonicity of p, respectively. That is, both q_{i-i} and q_{i+1} are to the left of q_i or both of them are to its right, and both q_{j-1} and q_{j+1} are above q_j or both of them are below q_j. We choose i and j such that $|i-j|$ is minimized, and assume without loss of generality that $i < j$ (note that it follows from Proposition 24 that $i \neq j$). Thus, the sub-path $p' := q_i\text{-}q_{i+1}\text{-}\ldots, q_{j-1}\text{-}q_j$ is both x-monotone and y-monotone.

By reflecting about the x- and/or y-axis if needed, we may assume that p' is ascending, that is, for every $l = i, \ldots, j-1$ we have $q_{l+1} \in \mathrm{NE}(q_l)$. Then it follows from Proposition 24 that $q_{i-1} \in \mathrm{SE}(q_i)$ and $q_{j+1} \in \mathrm{SE}(q_j)$. By Proposition 11 there is a point $x \notin Q'$ which is a neighbor of q_i and is between q_{i-1} and q_{i+1} in the rotation of q_i, and it follows from Proposition 25 that q_ix crosses the right side of Q'. The same argument implies that there is a point $y \notin Q'$ which is a neighbor of q_j and is between q_{j+1} and q_{j-1} in the rotation of q_j, such that q_jy crosses the bottom side of Q'. However, since q_j is to the right of q_i and above it, the edges q_ix and q_jy must cross, which is impossible. ◀

Call a 2-path w-q-z *bad* if it is not good, that is, there is an axis-parallel square Q'' that contains q, does not contain w and z, and qw and qz are edges in $\mathcal{DT}$ that cross the same side of Q''. We say that w-q-z is a bad *left* 2-path if qw and qz cross the left side of Q'', and define *right*, *top*, and *bottom* bad 2-paths analogously.

▶ **Proposition 27.** *Let* w-q-z *be a 2-path. Then:*

- w-q-z *is a bad left 2-path iff* $w \in \mathrm{SW}(q)$ *and* $z \in \mathrm{NW}(q)$*, or vice versa;*
- w-q-z *is a bad right 2-path iff* $w \in \mathrm{SE}(q)$ *and* $z \in \mathrm{NE}(q)$*, or vice versa;*
- w-q-z *is a bad top 2-path iff* $w \in \mathrm{NW}(q)$ *and* $z \in \mathrm{NE}(q)$*, or vice versa; and*
- w-q-z *is a bad bottom 2-path iff* $w \in \mathrm{SW}(q)$ *and* $z \in \mathrm{SE}(q)$*, or vice versa.*

▶ **Proposition 28.** *Every path in* $\mathcal{DT}[Q']$ *contains at most four bad 2-paths.*

Proof. Let $p := q_1\text{-}q_2\text{-}\ldots\text{-}q_k$ be a simple path in $\mathcal{DT}[Q']$ and suppose for a contradiction that p contains at least five bad 2-paths. By Proposition 26 the path p is x-monotone or y-monotone. Assume without loss of generality that p is y-monotone and that it goes upwards, that is, q_{i+1} is above q_i for every $i = 1, 2, \ldots, k-1$. It follows that p does not contain bad top or bad bottom 2-paths, for otherwise it would not be y-monotone. It is not hard to see that bad left and bad right 2-paths must alternate along p, that is, between every two bad left 2-paths there is a bad right 2-path and vice versa.

Consider the first five such bad 2-paths along the path p, and denote them by $q_{i_1-1}\text{-}q_{i_1}\text{-}q_{i_1+1}$, $q_{i_2-1}\text{-}q_{i_2}\text{-}q_{i_2+1}$, $q_{i_3-1}\text{-}q_{i_3}\text{-}q_{i_3+1}$, $q_{i_4-1}\text{-}q_{i_4}\text{-}q_{i_4+1}$ and $q_{i_5-1}\text{-}q_{i_5}\text{-}q_{i_5+1}$. By symmetry we may assume without loss of generality that $q_{i_1-1}\text{-}q_{i_1}\text{-}q_{i_1+1}$ is a bad left 2-path, and therefore $q_{i_3-1}\text{-}q_{i_3}\text{-}q_{i_3+1}$ and $q_{i_5-1}\text{-}q_{i_5}\text{-}q_{i_5+1}$ are also bad left 2-paths, whereas the 2-paths $q_{i_2-1}\text{-}q_{i_2}\text{-}q_{i_2+1}$ and $q_{i_4-1}\text{-}q_{i_4}\text{-}q_{i_4+1}$ are bad right.

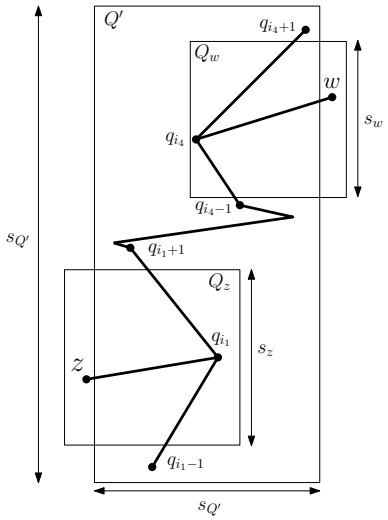


Figure 3 An illustration for the proof of Proposition 28.

Note that we may assume without loss of generality that q_{i_1} is to the right of q_{i_4}, for otherwise q_{i_5} must be to the right of q_{i_2} and by reflecting about the x-axis and renaming the points we get the desired assumption. By Proposition 11, q_{i_1} has a neighbor $z \notin Q'$ between q_{i_1-1} and q_{i_1+1} in the rotation of q_{i_1}. Let Q_z be a square that contains q_{i_1} and z and no other point from $\mathcal{S}$ and let s_z be its side length (refer to Figure 3).

It follows from Proposition 25 that q_{i_1-1} lies below Q_z, q_{i_1+1} lies above Q_z, and z lies to the left of Q'. Therefore, $(q_{i_1+1})_y - (q_{i_1-1})_y > s_z$. Similarly, q_{i_4} has a neighbor $w \notin Q'$ between q_{i_4+1} and q_{i_4-1} in the rotation of q_{i_4}. Let Q_w be a square that contains q_{i_4} and w and no other point from $\mathcal{S}$ and let s_w be its side length. Then q_{i_4-1} lies below Q_w, q_{i_4+1} lies above Q_w, and w lies to the right of Q'. Therefore, $(q_{i_4+1})_y - (q_{i_4-1})_y > s_w$.

Note that since q_{i_1} is to the right of q_{i_4} and z and w are to the left and to the right of Q', respectively, we have $s_z + s_w > ((q_{i_1})_x - (z)_x) + ((w)_x - (q_{i_4})_x) > s_{Q'}$, where $s_{Q'}$ is the side length of Q'. Observe also that since there are at least two other vertices between q_{i_1} and q_{i_4} along p, we have that $q_{i_1+1} \neq q_{i_4-1}$, and thus q_{i_1+1} lies below q_{i_4-1}. This implies that $((q_{i_1+1})_y - (q_{i_1-1})_y) + ((q_{i_4+1})_y - (q_{i_4-1})_y) < s_{Q'}$. Combining the inequalities we get, $s_{Q'} > ((q_{i_1+1})_y - (q_{i_1-1})_y) + ((q_{i_4+1})_y - (q_{i_4-1})_y) > s_z + s_w > ((q_{i_1})_x - (z)_x) + ((w)_x - (q_{i_4})_x) > s_{Q'}$, a contradiction. ◀

To complete the proof of Lemma 21 we will consider a path of length 11 in $\mathcal{DT}[Q']$. It follows from Proposition 24 that for every $q \in \mathcal{S} \cap Q'$ we have $\deg_{\mathcal{DT}[Q']}(q) \leq 4$. This implies that if Q' contains at least $1 + \sum_{i=1}^{5} 4^i = 1366$ points from $\mathcal{S}$ then $\mathcal{DT}[Q']$ contains a simple path of length at least 11. However, one can show that already 148 points suffice to guarantee the existence of a path of length 11. The proof of this can be found in the full version of this paper.

Let $p := q_1\text{-}q_2\text{-}\ldots\text{-}q_{12}$ be a simple path of length 11 in $\mathcal{DT}[Q']$. By Proposition 28 there are at most four bad 2-paths $q_{i-1}\text{-}q_i\text{-}q_{i+1}$ in p. Therefore, there is $2 \leq i \leq 10$ such that $q_{i-1}\text{-}q_i\text{-}q_{i+1}$ and $q_i\text{-}q_{i+1}\text{-}q_{i+2}$ are good 2-paths, and therefore Q contains a good 3-path $q_{i-1}\text{-}q_i\text{-}q_{i+1}\text{-}q_{i+2}$. Lemma 21 is proved. ◀

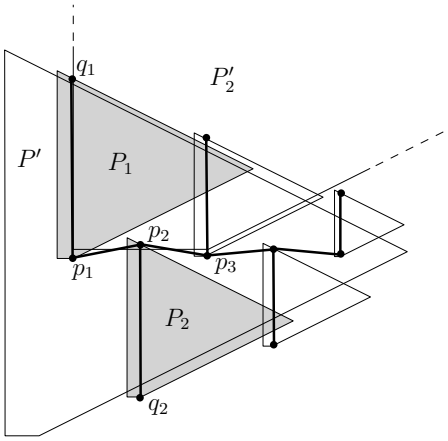


Figure 4 a convex quadrilateral that is not universally good.

4 Discussion

We have presented a general framework showing that if a convex polygon P satisfies a certain property (namely, is *universally good* as defined in Section 3.1), then there is an absolute constant m that depends only on P such that every set of points in the plane can be 2-colored such that every homothet of P that contains at least m points contains points of both colors.

We then used this framework for 2-coloring points with respect to squares, showing that one can 2-color any set of points in the plane such that every square that contains at least 1484 points from the point set contains points of both colors. It would be intersecting to improve this constant, as it is most likely not the best possible, and any such improvement would also improve the bound for polychromatic colorings (Corollary 4). The results for squares apply also for homothets of any given parallelogram by affine transformations. The main open problem related to our work is the following.

▶ **Problem 29.** *Is it true that for every convex polygon P there is a constant $m := m(P)$ such that it is possible to 2-color any set of points $\mathcal{S}$ such that every homothet of P that contains at least m points from $\mathcal{S}$ contains points of both colors?*

By Theorem 19 it would be enough to show that every convex polygon is universally good. In the full version of this paper we have demonstrated that any triangle is universally good, and thus provided a new proof for the known positive answer to Problem 29 for triangles. However, for other classes of convex polygons it seems that additional ideas are needed, in light of the following.

▶ **Lemma 30.**

1. *For every integer $n \geq 4$ there exists a convex polygon with n vertices that is not universally good.*
2. *For every even integer $n \geq 6$ there exists a centrally-symmetric convex polygon with n vertices that is not universally good.*

Figure 4 describes the construction that we use to prove the first part of Lemma 30.

We conclude with two interesting related open problems. Considering coloring of points with respect to disks, recall that Pálvölgyi [32] recently proved that there is no constant m such that any set of points in the plane can be 2-colored such that any (unit) disk that

contains at least m points from the given set is non-monochromatic (that is, contains points of both colors). Coloring the points with four colors such that any disk that contains at least two points is non-monochromatic is easy since the (generalized) Delaunay graph is planar. Therefore, it remains an interesting open problem whether there is a constant m such that any set of points in the plane can be 3-colored such that any disk that contains at least m points is non-monochromatic.

Perhaps the most interesting and challenging problem of coloring geometric hypergraphs is to color a planar set of points $\mathcal{S}$ with the minimum possible number of colors, such that every axis-parallel rectangle that contains at least two points from $\mathcal{S}$ is non-monochromatic. It is known that $\Omega(\log(|\mathcal{S}|)/\log^2\log(|\mathcal{S}|))$ colors are sometimes needed [11], and it is conjectured that $polylog(|\mathcal{S}|)$ colors always suffice. The latter holds when considering rectangles that contain at least three points [1], however, for the original question only polynomial upper bounds are known [2, 10, 13, 29].

References

1 Eyal Ackerman and Rom Pinchasi. On coloring points with respect to rectangles. *J. Comb. Theory, Ser. A*, 120(4):811–815, 2013. doi:10.1016/j.jcta.2013.01.005.

2 Deepak Ajwani, Khaled M. Elbassioni, Sathish Govindarajan, and Saurabh Ray. Conflict-free coloring for rectangle ranges using $O(n^{.382})$ colors. *Discrete & Computational Geometry*, 48(1):39–52, 2012. doi:10.1007/s00454-012-9425-5.

3 Greg Aloupis, Jean Cardinal, Sébastien Collette, Stefan Langerman, David Orden, and Pedro Ramos. Decomposition of multiple coverings into more parts. *Discrete & Computational Geometry*, 44(3):706–723, 2010. doi:10.1007/s00454-009-9238-3.

4 Prosenjit Bose, Paz Carmi, Sébastien Collette, and Michiel H. M. Smid. On the stretch factor of convex delaunay graphs. *JoCG*, 1(1):41–56, 2010. URL: http://jocg.org/index.php/jocg/article/view/5.

5 Sarit Buzaglo, Rom Pinchasi, and Günter Rote. Topological hypergraphs. In János Pach, editor, *Thirty Essays on Geometric Graph Theory*, pages 71–81. Springer New York, 2013. doi:10.1007/978-1-4614-0110-0_6.

6 Jean Cardinal, Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. Making triangles colorful. *arXiv preprint arXiv:1212.2346*, 2012.

7 Jean Cardinal, Kolja Knauer, Piotr Micek, and Torsten Ueckerdt. Making octants colorful and related covering decomposition problems. *SIAM journal on discrete mathematics*, 28(4):1948–1959, 2014.

8 Jean Cardinal, Kolja B. Knauer, Piotr Micek, and Torsten Ueckerdt. Making triangles colorful. *J. of Computational Geometry*, 4(1):240–246, 2013. URL: http://jocg.org/index.php/jocg/article/view/136.

9 Jean Cardinal, Kolja B. Knauer, Piotr Micek, and Torsten Ueckerdt. Making octants colorful and related covering decomposition problems. *SIAM J. Discrete Math.*, 28(4):1948–1959, 2014. doi:10.1137/140955975.

10 Timothy M. Chan. Conflict-free coloring of points with respect to rectangles and approximation algorithms for discrete independent set. In Tamal K. Dey and Sue Whitesides, editors, *Symposuim on Computational Geometry 2012, SoCG '12, Chapel Hill, NC, USA, June 17-20, 2012*, pages 293–302. ACM, 2012. doi:10.1145/2261250.2261293.

11 Xiaomin Chen, János Pach, Mario Szegedy, and Gábor Tardos. Delaunay graphs of point sets in the plane with respect to axis-parallel rectangles. *Random Struct. Algorithms*, 34(1):11–23, 2009. doi:10.1002/rsa.20246.

12 Matt Gibson and Kasturi Varadarajan. Optimally decomposing coverings with translates of a convex polygon. *Discrete & Computational Geometry*, 46(2):313–333, 2011. doi:10.1007/s00454-011-9353-9.

13 Sariel Har-Peled and Shakhar Smorodinsky. Conflict-free coloring of points and simple regions in the plane. *Discrete & Computational Geometry*, 34(1):47–70, 2005. doi:10.1007/s00454-005-1162-6.

14 Balázs Keszegh. Coloring half-planes and bottomless rectangles. *Computational Geometry*, 45(9):495–507, 2012.

15 Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover-decomposable. *Discrete & Computational Geometry*, 47(3):598–609, 2012.

16 Balázs Keszegh and Dömötör Pálvölgyi. Convex polygons are self-coverable. *Discrete & Computational Geometry*, 51(4):885–895, 2014.

17 Balázs Keszegh and Dömötör Pálvölgyi. Octants are cover-decomposable into many coverings. *Computational Geometry*, 47(5):585–588, 2014.

18 Balázs Keszegh and Dömötör Pálvölgyi. More on decomposing coverings by octants. *Journal of Computational Geometry*, 6(1):300–315, 2015.

19 Rolf Klein. *Concrete and abstract Voronoi diagrams*, volume 400. Springer Science & Business Media, 1989.

20 István Kovács. Indecomposable coverings with homothetic polygons. *Discrete & Computational Geometry*, 53(4):817–824, 2015. doi:10.1007/s00454-015-9687-9.

21 L. Ma. *Bisectors and Voronoi Diagrams for Convex Distance Functions*. PhD thesis, FernUniversität Hagen, Germany, 2000.

22 János Pach. Decomposition of multiple packing and covering. In *2. Kolloquium über Diskrete Geometrie*, number DCG-CONF-2008-001, pages 169–178. Institut für Mathematik der Universität Salzburg, 1980.

23 János Pach. Covering the plane with convex polygons. *Discrete & Computational Geometry*, 1:73–81, 1986. doi:10.1007/BF02187684.

24 János Pach and Dömötör Pálvölgyi. Indecomposable coverings. In *Proc. 41st Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, to appear.

25 János Pach, Dömötör Pálvölgyi, and Géza Toth. Survey on decomposition of multiple coverings. In Imre Bárány, Károly J. Böröczky, Gábor Fejes Tóth, and János Pach, editors, *Geometry — Intuitive, Discrete, and Convex*, volume 24 of *Bolyai Society Mathematical Studies*, pages 219–257. Springer Berlin Heidelberg, 2013.

26 János Pach and Gábor Tardos. Coloring axis-parallel rectangles. *Journal of Combinatorial Theory, Series A*, 117(6):776–782, 2010. doi:10.1016/j.jcta.2009.04.007.

27 János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. In Jin Akiyama, William Y. C. Chen, Mikio Kano, Xueliang Li, and Qinglin Yu, editors, *Discrete Geometry, Combinatorics and Graph Theory, 7th China-Japan Conference, CJCDGCGT 2005, Tianjin, China, November 18-20, 2005, Xi'an, China, November 22-24, 2005, Revised Selected Papers*, volume 4381 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 2005. doi:10.1007/978-3-540-70666-3_15.

28 János Pach, Gábor Tardos, and Géza Tóth. Indecomposable coverings. *Canadian mathematical bulletin*, 52(3):451–463, 2009.

29 János Pach and Géza Tóth. Conflict-free colorings. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry*, volume 25 of *Algorithms and Combinatorics*, pages 665–671. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-642-55566-4_30.

30 János Pach and Géza Tóth. Decomposition of multiple coverings into many parts. *Comput. Geom.*, 42(2):127–133, 2009. doi:10.1016/j.comgeo.2008.08.002.

31 Dömötör Pálvölgyi. Indecomposable coverings with concave polygons. *Discrete & Computational Geometry*, 44(3):577–588, 2010. doi:10.1007/s00454-009-9194-y.

32 Dömötör Pálvölgyi. Indecomposable coverings with unit discs. *CoRR*, abs/1310.6900, 2013. URL: http://arxiv.org/abs/1310.6900.

33 Dömötör Pálvölgyi and Géza Tóth. Convex polygons are cover-decomposable. *Discrete & Computational Geometry*, 43(3):483–496, 2010. doi:10.1007/s00454-009-9133-y.

34 Deniz Sariöz. Generalized delaunay graphs with respect to any convex set are plane graphs. *CoRR*, abs/1012.4881, 2010. URL: http://arxiv.org/abs/1012.4881.

35 Shakhar Smorodinsky. Conflict-free coloring and its applications. In Imre Bárány, Károly J. Böröczky, Gábor Fejes Tóth, and János Pach, editors, *Geometry — Intuitive, Discrete, and Convex*, volume 24 of *Bolyai Society Mathematical Studies*, pages 331–389. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-41498-5_12.

36 Shakhar Smorodinsky and Yelena Yuditsky. Polychromatic coloring for half-planes. *Journal of Combinatorial Theory, Series A*, 119(1):146 – 154, 2012.

37 Gábor Tardos and Géza Tóth. Multiple coverings of the plane with triangles. *Discrete & Computational Geometry*, 38(2):443–450, 2007. doi:10.1007/s00454-007-1345-4.

Approximating Dynamic Time Warping and Edit Distance for a Pair of Point Sequences*

Pankaj K. Agarwal[1], Kyle Fox[2], Jiangwei Pan[3], and Rex Ying[4]

1 **Department of Computer Science, Duke University, Durham, USA**
pankaj@cs.duke.edu
2 **Department of Computer Science, Duke University, Durham, USA**
kylefox@cs.duke.edu
3 **Department of Computer Science, Duke University, Durham, USA**
jwpan@cs.duke.edu
4 **Department of Computer Science, Duke University, Durham, USA**
zhitao.ying@duke.edu

Abstract

We present the first subquadratic algorithms for computing similarity between a pair of point sequences in $\mathbb{R}^d$, for any fixed $d > 1$, using dynamic time warping (DTW) and edit distance, assuming that the point sequences are drawn from certain natural families of curves. In particular, our algorithms compute $(1 + \varepsilon)$-approximations of DTW and ED in near-linear time for point sequences drawn from κ-packed or κ-bounded curves, and subquadratic time for backbone sequences. Roughly speaking, a curve is κ-packed if the length of its intersection with any ball of radius r is at most $\kappa \cdot r$, and it is κ-bounded if the sub-curve between two curve points does not go too far from the two points compared to the distance between the two points. In backbone sequences, consecutive points are spaced at approximately equal distances apart, and no two points lie very close together. Recent results suggest that a subquadratic algorithm for DTW or ED is unlikely for an arbitrary pair of point sequences even for $d = 1$.

The commonly used dynamic programming algorithms for these distance measures reduce the problem to computing a minimum-weight path in a grid graph. Our algorithms work by constructing a small set of rectangular regions that cover the grid vertices. The weights of vertices inside each rectangle are roughly the same, and we develop efficient procedures to compute the approximate minimum-weight paths through these rectangles.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Dynamic time warping, Edit distance, Near-linear-time algorithm, Dynamic programming, Well-separated pair decomposition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.6

1 Introduction

Motivation. Trajectories are functions from a time interval to $\mathbb{R}^d$, for $d \geq 1$, and they describe how physical systems change over time. Trajectories are recorded and inferred from numerous sources and are often represented as ordered sequences of points. These sources include GPS sensors in smart phones and vehicles, surveillance videos, shape-based touch

* Work on this paper is supported by NSF under grants CCF-11-61359, IIS-14-08846, CCF-15-13816, and ISS-14-47554 by an ARO grant W911NF-15-1-0408, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

© Pankaj K. Agarwal, Kyle Fox, Jiangwei Pan, and Rex Ying;
licensed under Creative Commons License CC-BY

screen authentication patterns, hurricane patterns, and time series data. A fundamental task for analyzing trajectory data is to measure the similarity between trajectories. For example, computing trajectory similarity is an important step in object segmentation from video trajectories [11], smart phone authentication using touch screen trajectories [14], and stock price prediction using historical patterns [29]. In many applications, it is not enough to merely *quantify* how similar pairs of trajectories are; we need to compute correspondences between their sample points as well. These correspondences represent shared structures between trajectories, which can be present not only in trajectories with physical constraints such as vehicle trajectories, but also in trajectories representing the movement of pedestrians [19] or hurricanes [24]. Having an effective way to identify similar portions between a pair of trajectories can greatly aid in identifying and understanding these shared structures.

Problem statement. Let $P = \langle p_1, \ldots, p_m \rangle$ and $Q = \langle q_1, \ldots, q_n \rangle$ be two sequences of points in $\mathbb{R}^d$ for some fixed $d \geq 1$. We define a *correspondence* as a pair (p_i, q_j). A set C of correspondences is *monotone* if for any pair of correspondences $(p_i, q_j), (p_{i'}, q_{j'})$ with $i' \geq i$ we have $j' \geq j$. We define the cost of C to be $\sum_{(p,q)\in C} ||pq||$, where $||\cdot||$ is the Euclidean norm. The similar portions of P and Q are represented by a set C of monotone correspondences, with the cost of C quantifying the quality of similarity. The goal is to compute a monotone set of correspondences with certain properties. While numerous criteria for computing correspondences have been proposed, we focus on two, which are widely used: dynamic time warping (DTW) and edit distance (ED). They are used for matching various types of sequences such as speech signals, DNA and protein sequences, protein backbones, time-series data, GPS/video trajectories, touch screen authentication trajectories, etc. [27, 18, 16, 23, 26, 25, 14].

DTW computes a monotone set of correspondences in which every point in P and Q appears at least once, and minimizes the sum of distances of corresponding pairs of points. Formally, the cost of DTW, denoted by $\mathsf{dtw}(P, Q)$, is $\mathsf{dtw}(P, Q) = \min_C \sum_{(p,q)\in C} ||pq||$, where the minimum is taken over all sets C of monotone correspondences that cover all points of P and Q. DTW allows a point to appear in multiple correspondences, so it matches two sequences effectively even if the sampling rates are different.

Edit distance (also called Levenshtein distance) seeks a monotone *matching* on the points in P and Q of minimum cost; each point in P corresponds to at most one point in Q and vice versa. It also adds a *gap* penalty, say g, for each point in $P \cup Q$ that does not appear in any correspondence. The cost of ED, denoted by $\mathsf{ed}(P, Q)$, is $\mathsf{ed}(P, Q) = \min_C \sum_{(p,q)\in C} ||pq|| + g(m + n - 2|C|)$, where the minimum is taken over all sets C of monotone matchings in the complete bipartite graph $P \times Q$. More sophisticated gap penalty functions have been proposed [16], but we focus on the simple linear gap penalty function. By tuning g correctly, meaningful correspondences can be computed even when faced with outlier points that arise from measurement errors or short deviations in otherwise similar trajectories.

Given a parameter $\varepsilon \in (0, 1)$, we wish to develop efficient $(1+\varepsilon)$-approximation algorithms for computing $\mathsf{dtw}(P, Q)$ and $\mathsf{ed}(P, Q)$, i.e., they return a value Δ such that $\mathsf{dtw}(P, Q) \leq \Delta \leq (1 + \varepsilon)\mathsf{dtw}(P, Q)$ or $\mathsf{ed}(P, Q) \leq \Delta \leq (1 + \varepsilon)\mathsf{ed}(P, Q)$. We are also interested in computing correspondences that realize these distances.

Prior results. It is well-known that both $\mathsf{dtw}(P, Q)$ and $\mathsf{ed}(P, Q)$, as well as the relevant correspondences, can be computed in $O(mn)$ time using dynamic programming [22]. A series of recent papers show that there exists no algorithm for computing $\mathsf{dtw}(P, Q)$ or $\mathsf{ed}(P, Q)$

in time $O(n^{2-\delta})$ for any $\delta > 0$ unless the Strong Exponential Time Hypothesis (SETH) of Impagliazzo and Paturi [21] is false. In particular, Backurs and Indyk [6] showed a conditional lower bound for edit distance, and Abboud *et al.* [1] and Bringmann and Künnemann [9] independently showed similar lower bounds for DTW. While most of these lower bounds were presented for the string versions of their respective problems, the DTW lower bound of Bringmann and Künnemann uses sequences of points in $\mathbb{R}$. Unless SETH is false, there exists no strictly subquadratic time algorithm for DTW, even in our setting of point sequences in $\mathbb{R}^d$. Similar conditional lower bounds have been shown for other distance and similarity problems [2, 6, 1, 7, 9]. Some of these results suggest that even strongly subquadratic approximation schemes seem unlikely [1, 7].

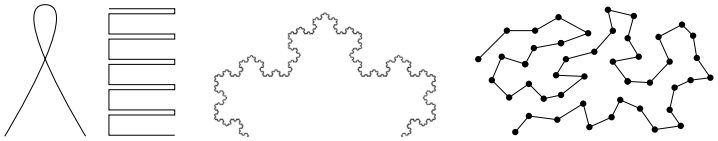
In view of the recent lower bounds, a natural question to ask is whether near-linear, or even subquadratic, algorithms exist for certain natural families of point sequences. Aronov *et al.* [5] gave subquadratic-time approximation schemes for the discrete Fréchet distance of κ-*bounded* and *backbone* point sequences. Discrete Fréchet distance is similar to DTW except that one uses max instead of sum in the definition. Restricting themselves to these families of sequences allowed them to subvert the hardness result of Bringmann [7] mentioned above. Driemel *et al.* [15] extended the approximation results to the continuous Fréchet distance and to the family of so-called κ-*packed* curves (defined below); see also [8, 10]. Roughly speaking, these algorithms guess the value of the (discrete) Fréchet distance, say, Δ and simplify the two sequences within an $\frac{\varepsilon}{2}\Delta$ error. Only a subquadratic number of entries in the dynamic-programming table need to be computed when matching each point p in one sequence with points in the other (simplified) sequence that lie within distance Δ from p.

We note that while (discrete) Fréchet distance is a reasonable measure to compute the similarity between two sequences, it is not effective in identifying similar portions of the sequences, and DTW or edit distance and their variants are more widely used for computing correspondences. Currently, no subquadratic-time approximation results are known for DTW, but there are a number of heuristics designed to speed up its exact computation in practice (see [30]). Subquadratic-time approximation algorithms are known for variants of edit distance, but these algorithm have at least a polylogarithmic approximation ratio [4].

The aforementioned algorithms for (discrete) Fréchet distance do not extend to DTW or ED, because these measures add the distances of corresponding pairs instead of taking their maximum value. As such, we cannot globally simplify the two curves, and we cannot restrain ourselves to computing a small number of entries in the dynamic-programming table for each point $p \in P$, because it may be matched with a far away point in Q.

Our results. We present algorithms for computing $\mathsf{dtw}(P, Q)$ and $\mathsf{ed}(P, Q)$ approximately which have subquadratic running time for several "well-behaved" families of input sequences. The correspondences realizing these distances can also be recovered. The two algorithms are almost identical except a few implementation details. In the worst case, their running time is quadratic for arbitrary point sequences, but it is near-linear if P and Q are κ-packed or κ-bounded sequences and subquadratic when P and Q satisfy the conditions for a backbone sequence. These are the first approximation algorithms that compute $\mathsf{dtw}(P, Q)$ and $\mathsf{ed}(P, Q)$ for such point sequences in subquadratic time.

For $x \in \mathbb{R}^d$ and $r \in \mathbb{R}^+$, let $\mathsf{B}(x, r)$ denote the ball of radius r centered at x. Given $\kappa \in \mathbb{R}^+$, a curve γ in $\mathbb{R}^d$ is κ-*packed* if the length of γ inside any ball of radius r is bounded by κr [15], and γ is κ-*bounded* if for any $0 \le t < t' \le 1$, $\gamma[t:t'] \subseteq \mathsf{B}\left(\gamma(t), \frac{\kappa}{2}||\gamma(t)\gamma(t')||\right) \cup \mathsf{B}\left(\gamma(t'), \frac{\kappa}{2}||\gamma(t)\gamma(t')||\right)$, where $\gamma : [0,1] \to \mathbb{R}^d$ and $\gamma[t:t']$ is the portion of γ between $\gamma(t)$ and $\gamma(t')$ [3]. We say a point sequence P is κ-packed (resp. κ-bounded) if the polygonal curve



(a) (b) (c)

Figure 1 (a) κ-packed curves that are not κ-bounded. (b) The top half of the Koch snowflake is a κ-bounded curve that is not κ-packed. (c) A backbone sequence that is neither κ-bounded nor κ-packed.

that connects points of P is κ-packed (resp. κ-bounded). A point sequence $P = \langle p_1, \ldots, p_m \rangle$ is said to be a backbone sequence if it satisfies the following two conditions: (i) for any pair of non-consecutive integers $i, j \in [1, m]$, $||p_i p_j|| > 1$; (ii) for any integer i in $(1, m]$, $c_1 \leq ||p_{i-1} p_i|| \leq c_2$, where c_1, c_2 are positive constants [5]. These sequences are commonly used to model protein backbones where each vertex represents a C_α atom, connected to its neighbors via covalent bonds. See Figure 1 for examples of κ-packed, κ-bounded, and backbone curves. We use γ_P to denote the polygonal curve connecting the points of sequence P. Our results are summarized in the following theorems.

▶ **Theorem 1.** *Let P and Q be two point sequences of length at most n in $\mathbb{R}^d$, and let $\varepsilon \in (0, 1)$ be a parameter. An $(1+\varepsilon)$-approximate value of $\mathsf{dtw}(P, Q)$ can be computed in $O(\frac{\kappa}{\varepsilon} n \log n)$, $O(\frac{\kappa^d}{\varepsilon^d} n \log n)$, and $O(\frac{1}{\varepsilon} n^{2-1/d} \log n)$ time if P, Q are κ-packed, κ-bounded, and backbone sequences, respectively.*

▶ **Theorem 2.** *Let P and Q be two point sequences of length at most n in $\mathbb{R}^d$, and let $\varepsilon \in (0, 1)$ be a parameter. An $(1+\varepsilon)$-approximate value of $\mathsf{ed}(P, Q)$ can be computed in $O(\frac{\kappa}{\varepsilon} n \log n)$, $O(\frac{\kappa^d}{\varepsilon^d} n \log n)$, and $O(\frac{1}{\varepsilon} n^{2-1/(d+1)} \log n)$ time if P, Q are κ-packed, κ-bounded, and backbone sequences, respectively.*

Recall that the standard dynamic programming algorithm for computing $\mathsf{dtw}(P, Q)$ or $\mathsf{ed}(P, Q)$ constructs a weighted grid $\mathsf{V} = \{(i, j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ and formulates the problem as computing a minimum-weight path from $(1, 1)$ to (m, n). Based on the observation that nearby grid points typically have similar weights when P, Q are "well-behaved", we construct a small number of potentially overlapping rectangular regions of V, whose union contains the minimum-weight path in V, such that all grid points within each rectangle have similar weights; see Figure 2. We construct the rectangles so that the number of "boundary points" of the rectangles is near linear when P, Q are κ-packed or κ-bounded and subquadratic when they are backbone sequences. We then use an efficient procedure to compute approximate minimum-weight paths from $(1, 1)$ to all boundary points.

The algorithm framework is quite general and can work for a variety of similar distance measures based on monotone correspondences. For example, our results immediately generalize to variants of dynamic time warping and edit distance that use the k-th powers of distance between points instead of their Euclidean distance for any constant $k > 0$. Moreover, the framework may prove useful in designing subquadratic-time algorithms for other problems that can be solved with standard dynamic programming.

2 Algorithm for DTW

Let $P = \langle p_1, \ldots, p_m \rangle$ and $Q = \langle q_1, \ldots, q_n \rangle$ be two point sequences in $\mathbb{R}^d$. Let $\varepsilon \in (0, 1)$ be a parameter. We present a $(1+\varepsilon)$-approximation algorithm for computing $\mathsf{dtw}(P, Q)$. Without

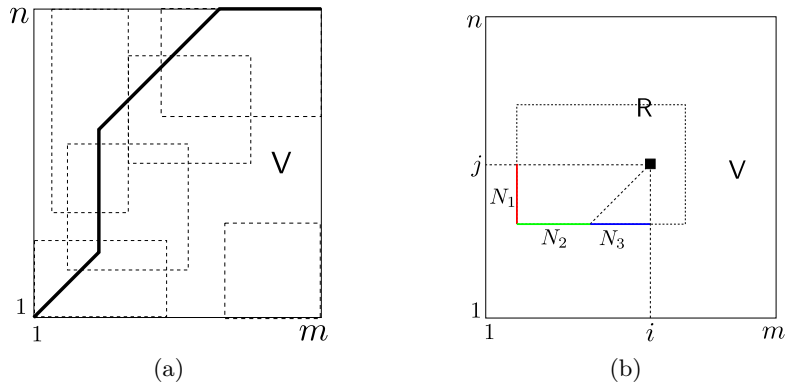


Figure 2 (a) Illustration of our algorithm: compute a small set of rectangles that covers the optimal admissible path from $(1,1)$ to (m,n) (drawn in bold); (b) partitioning the boundary points of rectangle R dominated by (i,j).

loss of generality, assume that $m \le n$ and $\varepsilon \ge 1/n$. If $\varepsilon < 1/n$, we can simply compute $\mathsf{dtw}(P,Q)$ in $O(mn) = O(n/\varepsilon)$ time via dynamic programming.

Given positive integers $i < i'$, let $[i:i'] := \{i, i+1, \ldots, i'\}$, and let $[i] := [1:i]$. Let $\mathsf{V} = [m] \times [n]$ denote a set of grid points[1] in $\mathbb{R}^2$, and define a weight function $\omega : \mathsf{V} \to \mathbb{R}_{\ge 0}$ where $\omega(i,j)$ is the Euclidean distance between p_i and q_j. Two different grid points in V are said to be *neighboring* if they differ by at most 1 in each coordinate. We say (i,j) dominates (i',j') if $i \ge i'$ and $j \ge j'$. A path π is a sequence of neighboring grid points; π is *admissible* if it is non-decreasing in both coordinates. Define the weight of the path π, $\omega(\pi)$, as the sum of the weights of the grid points along π. Define $\mu(i,j)$ as the minimum weight of an admissible path from $(1,1)$ to (i,j). So $\mathsf{dtw}(P,Q) = \mu(m,n)$.

For $1 \le i_1 \le i_2 \le m$ and for $1 \le j_1 \le j_2 \le n$, the set of grid points $[i_1:i_2] \times [j_1:j_2]$ is called a *rectangle*. A point $(i,j) \in \mathsf{V}$ is a *boundary point* of this rectangle if $i \in \{i_1, i_2\}$ or $j \in \{j_1, j_2\}$. We first outline the algorithm for computing an $(1+\varepsilon)$-approximate value of $\mu(m,n)$, and then describe it in detail in Sections 2.1-2.3. Section 2.4 analyzes its running time for well-behaved point sequences.

(i) Compute an estimate $\underline{\mathsf{d}}$ of $\mathsf{dtw}(P,Q)$ such that $\underline{\mathsf{d}} \le \mathsf{dtw}(P,Q) \le 4n\underline{\mathsf{d}}$. Let $\overline{\mathsf{d}} = 4n\underline{\mathsf{d}}$.

(ii) Compute a set $\mathcal{R}$ of (possibly overlapping) rectangles and a weight ω_{R} for each rectangle $\mathsf{R} \in \mathcal{R}$, such that:

- **(a)** for all $\mathsf{R} \in \mathcal{R}$ and $(i,j) \in \mathsf{R}$, $|\omega(i,j) - \omega_{\mathsf{R}}| \le \frac{\varepsilon}{2} \max\{\omega_{\mathsf{R}}, \underline{\mathsf{d}}/2n\}$,
- **(b)** if $(i,j) \in \mathsf{V}$ and $\omega(i,j) \le \overline{\mathsf{d}}$, then there exists a rectangle $\mathsf{R} \in \mathcal{R}$ such that $(i,j) \in \mathsf{R}$.

Conditions (a) and (b) above ensure that the weights of grid points in each rectangle are roughly the same, and the minimum-weight admissible path between $(1,1)$ and (m,n) is contained in the union of the rectangles. See Figure 2(a).

Let $\mathcal{B} = \bigcup_{\mathsf{R} \in \mathcal{R}} \partial\mathsf{R}$ be the set of boundary points of the rectangles in $\mathcal{R}$. The sizes of $\mathcal{R}$ and $\mathcal{B}$ depend on the input sequences P, Q. In the worst case $|\mathcal{R}|, |\mathcal{B}| = \Theta(mn)$, but they are subquadratic if P, Q are well-behaved.

(iii) For every $(i,j) \in \mathcal{B}$, compute a $(1+\varepsilon)$-approximate value $\tilde{\mu}(i,j)$ of $\mu(i,j)$, i.e., $\mu(i,j) \le \tilde{\mu}(i,j) \le (1+\varepsilon)\mu(i,j)$.

(iv) Return $\tilde{\mu}(m,n)$.

[1] Note that in this paper, a point can refer to either a grid point in V or a sequence point from $P \cup Q$.

2.1 An $O(n)$ approximation

Let $\mathsf{dfr}(P,Q)$ denote the discrete Fréchet distance between P and Q, i.e., replace sum with max in the definition of $\mathsf{dtw}(P,Q)$.

▶ **Lemma 3.** $\mathsf{dfr}(P,Q) \le \mathsf{dtw}(P,Q) \le 2n \cdot \mathsf{dfr}(P,Q)$.

Proof. Let π^* be the minimum-weight admissible path from $(1,1)$ to (m,n) corresponding to $\mathsf{dtw}(P,Q)$. Then $\mathsf{dfr}(P,Q) \le \max_{(i,j)\in\pi^*} ||p_iq_j|| \le \sum_{(i,j)\in\pi^*} ||p_iq_j|| = \mathsf{dtw}(P,Q)$. Similarly, let $\overline{\pi}$ be the admissible path corresponding to $\mathsf{dfr}(P,Q)$. Then, $\mathsf{dtw}(P,Q) \le \sum_{(i,j)\in\overline{\pi}} ||p_iq_j|| \le 2n \max_{(i,j)\in\overline{\pi}} ||p_iq_j|| = 2n\mathsf{dfr}(P,Q)$. The second inequality follows because $|\overline{\pi} \cap \mathsf{V}| \le m+n \le 2n$. ◀

Aronov *et al.* [5] gave a near-linear time algorithm for computing the approximate discrete Fréchet distance between κ-bounded point sequences. Their algorithm directly implies an $O(\kappa^d n \log n)$-time 2-approximation algorithm for computing $\mathsf{dfr}(P,Q)$ for κ-bounded sequences. They also prove that the same algorithm computes the 2-approximation of the discrete Fréchet distance between backbone sequences in $O(n^{2-2/d})$ time. With a slight modification of their analysis, it turns out that their algorithm also works for κ-packed sequences. We summarize these observations in the following lemma.

▶ **Lemma 4.** *A 2-approximation of* $\mathsf{dfr}(P,Q)$ *can be computed in* $O(\kappa n \log n)$, $O(\kappa^d n \log n)$, *and* $O(n^{2-2/d})$ *time if* P,Q *are* κ*-packed,* κ*-bounded, and backbone sequences, respectively.*

Let $\overline{\mathsf{dfr}}(P,Q)$ be the 2-approximation of $\mathsf{dfr}(P,Q)$ computed in Lemma 4; i.e., $\mathsf{dfr}(P,Q) \le \overline{\mathsf{dfr}}(P,Q) \le 2 \cdot \mathsf{dfr}(P,Q)$. Set $\underline{\mathsf{d}} = \overline{\mathsf{dfr}}(P,Q)/2$. By Lemma 3, $\underline{\mathsf{d}} \le \mathsf{dtw}(P,Q) \le 4n\underline{\mathsf{d}}$.

2.2 Computing rectangles $\mathcal{R}$

Let H be an axis-aligned hypercube in $\mathbb{R}^d$ that contains $P \cup Q$ and has side length of a power of 2. Let $\mathcal{T}$ be a *quadtree*, a 2^d-way tree, on $P \cup Q$. Each node v of $\mathcal{T}$ is associated with an axis-aligned box $\square_v$. The root of $\mathcal{T}$ is associated with H. A node v is a leaf if $|\square_v \cap (P \cup Q)| \le 1$. The boxes associated with the children of a node v are obtained by partitioning $\square_v$ into 2^d congruent hypercubes — the side length of each resulting box is half that of $\square_v$. For a node $v \in \mathcal{T}$, let $p(v)$ denote its parent, $\mathsf{ch}(v)$ the set of children of v, $\Delta(v)$ the side length of $\square_v$, $P_v = P \cap \square_v$, and $Q_v = Q \cap \square_v$. Let $m_v = |P_v|$ and $n_v = |Q_v|$. For two nodes $u, v \in \mathcal{T}$, let $\mathsf{d}_\square(u,v) = \min_{p,q\in\square_u\times\square_v} ||pq||$ denote the distance between $\square_u$ and $\square_v$. We say two nodes u, v are *neighboring* if u and v are of the same level of $\mathcal{T}$ and $\square_u$ and $\square_v$ share a facet. We do not construct the entire $\mathcal{T}$ but only a portion as described below.

Let $\underline{r}$ and $\bar{r}$ be powers of 2 such that $\underline{r} \le \frac{\varepsilon}{8\sqrt{d}}\underline{\mathsf{d}}/2n \le 2\underline{r}$ and $\bar{r} \le 4\overline{\mathsf{d}} \le 2\bar{r}$. We call a node v of $\mathcal{T}$ *active* if $\Delta(v) \in [\underline{r}, \bar{r}]$ and $m_v + n_v > 0$. Let A denote the set of active nodes of $\mathcal{T}$. We construct the set A of active nodes of $\mathcal{T}$ and the sets P_v, Q_v for each active node $v \in A$. By definition, the active nodes lie in a portion of the quadtree $\mathcal{T}$ of height $O(\log(\bar{r}/\underline{r})) = O(\log n)$. Thus, $|A| = O(n)$ and $\sum_{v\in A}(m_v + n_v) = O(n \log n)$. Computing A and P_v, Q_v for all $v \in A$ takes $O(n \log n)$ time.

To compute the rectangles in $\mathcal{R}$, we first construct a family $\mathcal{F} = \{(u_1,v_1), \ldots, (u_s,v_s)\}$ of "well-separated" pairs of active nodes with the following properties:

(P1) For every $t \le s$, $\max\{\Delta(u_t), \Delta(v_t)\} \le \frac{\varepsilon}{4\sqrt{d}} \max\{\mathsf{d}_\square(u_t,v_t), \underline{\mathsf{d}}/2n\}$.

(P2) For all pairs $(i,j) \in \mathsf{V}$ with $||p_iq_j|| \le \overline{\mathsf{d}}$, there exists a unique pair (u_t,v_t) such that $p_i \in P_{u_t}$ and $q_j \in Q_{v_t}$.

Intuitively, (u,v) is well-separated when for any $p \in \square_u$ and $q \in \square_v$, we have $||pq|| \approx \mathsf{d}_\square(u,v)$. Then, for each pair $(u_t,v_t) \in \mathcal{F}$, we construct a small number of rectangles.

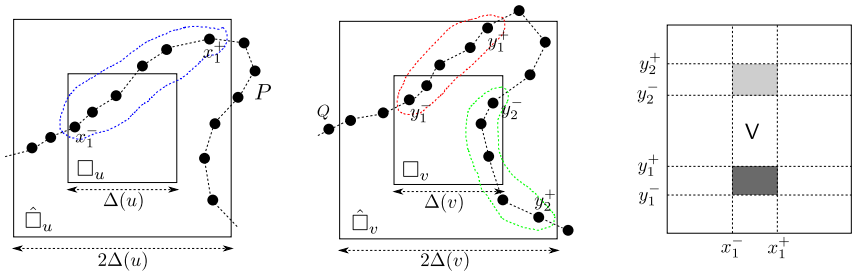


Figure 3 One MCS in $\hat{\square}_u$ (left) and two MCSs in $\hat{\square}_v$ (middle). Together, two rectangles $\mathsf{R}_{11}, \mathsf{R}_{12}$ (right) are created (shaded areas).

Constructing $\mathcal{F}$. Properties (P1) and (P2) are similar to those for the so-called *well-separated pair decomposition* (WSPD) of a point set [12] (see also [20]). We therefore adapt their algorithm. We first describe a recursive procedure PAIRING(u, v), where u, v are two active nodes, which generates a family of pairs for P_u, Q_v.

PAIRING(u, v)
if $\max\{\Delta(u), \Delta(v)\} \leq \frac{\varepsilon}{4\sqrt{d}} \max\{\mathsf{d}_\square(u, v), \underline{\mathsf{d}}/2n\}$
  add (u, v) to $\mathcal{F}$; return
if $\Delta(u) \geq \Delta(v)$, **then**
  $\forall w \in \mathsf{ch}(u)$ if $P_w \neq \varnothing$, do PAIRING(w, v)
  else $\forall z \in \mathsf{ch}(v)$ if $Q_z \neq \varnothing$, do PAIRING(u, z)

Let u_0 be a top-level active node with $\Delta(u_0) = \bar{r}$ and $P_{u_0} \neq \varnothing$. We call PAIRING(u_0, v_0) if $Q_{v_0} \neq \varnothing$ and either $v_0 = u_0$ or v_0 is a neighboring node of u_0.

(P1) is obviously true by the termination condition of the PAIRING procedure. (P2) is true because for each $(i, j) \in \mathsf{V}$ with $||p_i q_j|| \leq \overline{\mathsf{d}}$, it must be that p_i and q_j are contained in either the same active node or two neighboring active nodes of side length $\bar{r}$. The stopping criterion ensures that the PAIRING procedure never visits a node v with $\Delta(v) < \underline{r}$.

By adapting the analysis of the WSPD algorithm, the following lemma can be proven.

▶ **Lemma 5.** *If $(u, v) \in \mathcal{F}$, then*
(i) $\max\{\Delta(u), \Delta(v)\} \leq \min\{\Delta(p(u)), \Delta(p(v))\}$*;*
(ii) $\Delta(u)/2 \leq \Delta(v) \leq 2\Delta(u)$*; and*
(iii) *there is a constant $c \geq 0$ such that $\mathsf{d}_\square(u, v) \leq \frac{c}{\varepsilon}\Delta(u)$.*

Constructing $\mathcal{R}$. We describe how to construct rectangles from each well-separated pair $(u, v) \in \mathcal{F}$. Let $\hat{\square}$ denote the box concentric to $\square$ with twice the side length of $\square$. The algorithm visits a subset of points of P in sequential order. The algorithm starts from the first unvisited point of P_u and walks along P until P exits $\hat{\square}_u$; it then repeats this walk by jumping to the next point of P_u; this process stops when all points of P_u have been visited. Each walk corresponds to a maximal contiguous subsequence (MCS) of P in $\hat{\square}_u$ with the first point inside $\square_u$. Let $S_u(P) = \{[x_1^- : x_1^+], \ldots, [x_{\alpha_u}^- : x_{\alpha_u}^+]\}$ denote the MCSs as constructed above. Similarly, we compute $S_v(Q) = \{[y_1^- : y_1^+], \ldots, [y_{\beta_v}^- : y_{\beta_v}^+]\}$ denoting the MCSs of Q in $\hat{\square}_v$. For every pair $a \in [\alpha_u], b \in [\beta_v]$, we define the rectangle $\mathsf{R}_{ab} = [x_a^- : x_a^+] \times [y_b^- : y_b^+]$ and set its weight $\omega_{\mathsf{R}_{ab}} = \mathsf{d}_\square(u, v)$. Set $\mathcal{R}_{uv} = \{\mathsf{R}_{ab} \mid a \in [\alpha_u], b \in [\beta_v]\}$ and $\mathcal{R} = \bigcup_{(u,v)\in\mathcal{F}} \mathcal{R}_{uv}$. See Figure 3 for an illustration of the MCSs of $S_u(P), S_v(Q)$, and the rectangles in $\mathcal{R}_{uv}$.

▶ Remark. The rectangles in $\mathcal{R}_{uv}$ cover all the grid points corresponding to $P_u \times Q_v$, i.e., if $(p_i, q_j) \in P_u \times Q_v$ then $(i, j) \in \bigcup \mathcal{R}_{uv}$. Since $\bigcup \mathcal{R}_{uv}$ may also contain grid points that

correspond to pairs in $(P \cap (\hat{\square}_u \setminus \square_u)) \times (Q \cap (\hat{\square}_v \setminus \square_v))$, a grid point may lie in multiple rectangles of $\mathcal{R}$, implying that the rectangles in $\mathcal{R}$ may overlap. Had we defined $S_u(P), S_v(Q)$ to be MCSs of P_u and Q_v respectively, the rectangles would have been disjoint, but we might have ended up creating $\Omega(n^2)$ rectangles in the worst case. As we will prove in Section 2.4, by allowing $\mathcal{R}_{uv}$ to cover extra points, we keep the size of $\mathcal{R}$ and $\mathcal{B}$ small.

We show that the set of rectangles $\mathcal{R}$ satisfies the conditions in step (ii) of the algorithm.

▶ **Lemma 6.** *$\mathcal{R}$ satisfies the following properties:*

(i) *For all $\mathsf{R} \in \mathcal{R}$ and for all $(i,j) \in \mathsf{R}$, $|\omega(i,j) - \omega_{\mathsf{R}}| \leq \frac{\varepsilon}{2} \max\{\omega_{\mathsf{R}}, \underline{\mathsf{d}}/2n\}$.*

(ii) *If $(i,j) \in \mathsf{V}$ and $\omega(i,j) \leq \overline{\mathsf{d}}$, then there exists a rectangle $\mathsf{R} \in \mathcal{R}$ such that $(i,j) \in \mathsf{R}$.*

Proof.

(i) Suppose R is constructed from some well-separated pair $(u_t, v_t) \in \mathcal{F}$. By construction, if $(i,j) \in \mathsf{R}$, then $p_i \in \hat{\square}_{u_t}$ and $q_j \in \hat{\square}_{v_t}$. Therefore, $\omega(i,j) = ||p_i q_j|| \leq \mathsf{d}_{\square}(u_t, v_t) + \sqrt{d}(\Delta(u_t) + \Delta(v_t))$. By property (P1) and $\omega_{\mathsf{R}} = \mathsf{d}_{\square}(u_t, v_t)$, we have $\omega(i,j) \leq \omega_{\mathsf{R}} + 2\sqrt{d}\frac{\varepsilon}{4\sqrt{d}} \max\{\omega_{\mathsf{R}}, \underline{\mathsf{d}}/2n\} \leq \omega_{\mathsf{R}} + \frac{\varepsilon}{2} \max\{\omega_{\mathsf{R}}, \underline{\mathsf{d}}/2n\}$. Similarly, we can prove $\omega(i,j) \geq \omega_{\mathsf{R}} - \frac{\varepsilon}{2} \max\{\omega_{\mathsf{R}}, \underline{\mathsf{d}}/2n\}$.

(ii) By property (P2), there must exist a pair $(u,v) \in \mathcal{F}$ such that $p_i \in P_u$ and $q_j \in Q_v$. Since $\bigcup \mathcal{R}_{uv}$ "covers" $P_u \times Q_v$, there is a rectangle $\mathsf{R} \in \mathcal{R}_{uv}$ that contains the grid point (i,j). ◀

The time taken to construct the set $\mathcal{R}$ is $O(|\mathcal{R}|)$ plus the time taken to generate $\mathcal{F}$. We bound the latter quantity in Section 2.4.

2.3 Computing admissible paths

We now describe an algorithm that for each $(i,j) \in \mathcal{B}$ computes $(1+\varepsilon)$-approximate value $\tilde{\mu}(i,j)$ of $\mu(i,j)$ in amortized constant time.

We say a point $(i,j) \in \mathsf{V}$ *hits* a rectangle $\mathsf{R} = [i_1 : i_2] \times [j_1 : j_2]$ if $i_1 < i \leq i_2$ and $j_1 < j \leq j_2$, i.e., $(i,j) \in \mathsf{R}$ but not on its left or bottom boundary. The algorithm sets $\tilde{\mu}(1,1) = \omega(1,1)$, and processes the points of $\mathcal{B}$ from bottom to top and from left to right in a row. Suppose the current point is (i,j). There are two cases:

(i) If (i,j) does not hit any rectangle in $\mathcal{R}$, we set

$$\tilde{\mu}(i,j) = \min\{\tilde{\mu}(i-1,j), \tilde{\mu}(i,j-1), \tilde{\mu}(i-1,j-1)\} + \omega(i,j), \tag{1}$$

where $\tilde{\mu}(a,b) = \infty$ if $(a,b) \notin \mathcal{B}$.

(ii) Let $\mathsf{R} = [x^- : x^+] \times [y^- : y^+]$ be a rectangle hit by (i,j). Let $N(i,j)$ be the set of points on the left and bottom boundaries of R that are dominated by (i,j). Then the optimal path from $(1,1)$ to (i,j) has to pass through a point of $N(i,j)$. We temporarily set the weight of all points inside R to be ω_{R}. We therefore set

$$\tilde{\mu}(i,j) = \min_{(a,b) \in N(i,j)} \tilde{\mu}(a,b) + \max\{i-a, j-b\}\, \omega_{\mathsf{R}}. \tag{2}$$

The following lemma states that our algorithm returns a $(1+\varepsilon)$-approximation of $\mathsf{dtw}(P,Q)$.

▶ **Lemma 7.** *For each $(i,j) \in \mathcal{B}$, if $\mu(i,j) \leq \overline{\mathsf{d}}$, then*

$$|\tilde{\mu}(i,j) - \mu(i,j)| \leq \frac{\varepsilon}{2}(\mu(i,j) + (i+j)\underline{\mathsf{d}}/2n).$$

Proof. By induction on the order in which the $\tilde{\mu}$ values of points in $\mathcal{B}$ are computed, we prove $\tilde{\mu}(i,j) - \mu(i,j) \leq \frac{\varepsilon}{2}(\mu(i,j) + (i+j)\underline{\mathsf{d}}/2n)$. The lemma is obviously true for $(1,1)$. Assume it is true for all points of $\mathcal{B}$ processed before (i,j). We prove it is also true for (i,j).

If (i,j) does not hit any rectangle in $\mathcal{R}$, then $\tilde{\mu}(i,j)$ is computed using (1). Let $(a,b) \in \{(i-1,j),(i,j-1),(i-1,j-1)\}$ be the predecessor of (i,j) in the optimal admissible path from $(1,1)$ to (i,j). Then $\mu(i,j) = \mu(a,b) + \omega(i,j)$. Since $\mu(a,b) \leq \mu(i,j) \leq \overline{\mathsf{d}}$, there is a rectangle R containing (a,b). Since (i,j) does not hit any rectangle, (a,b) must actually lie on the boundary of R, and thus in $\mathcal{B}$. So by induction hypothesis,

$$\tilde{\mu}(i,j) - \mu(i,j) = \tilde{\mu}(a,b) - \mu(a,b) \leq \frac{\varepsilon}{2}(\mu(a,b) + (a+b)\underline{\mathsf{d}}/2n) \leq \frac{\varepsilon}{2}(\mu(i,j) + (i+j)\underline{\mathsf{d}}/2n).$$

In the second case, let $\mathsf{R} \in \mathcal{R}$ be the rectangle hit by (i,j) and used to compute $\tilde{\mu}(i,j)$. Let (a,b) be the intersection of the optimal admissible path from $(1,1)$ to (i,j) and the boundary of R. Then by (2),

$$\begin{aligned}\tilde{\mu}(i,j) &\leq \tilde{\mu}(a,b) + \max\{i-a, j-b\}\omega_{\mathsf{R}} \\ &\leq \mu(a,b) + \frac{\varepsilon}{2}(\mu(a,b) + (a+b)\underline{\mathsf{d}}/2n) + \max\{i-a, j-b\}\omega_{\mathsf{R}} \\ &\leq \mu(i,j) + \frac{\varepsilon}{2}(\mu(i,j) + (i+j)\underline{\mathsf{d}}/2n).\end{aligned}$$

The last inequality is satisfied, because $\omega_{\mathsf{R}} \leq \omega(h,k) + \frac{\varepsilon}{2}\max\{\omega(h,k), \underline{\mathsf{d}}/2n\}$ for any $(h,k) \in \mathsf{R}$ by Lemma 6. Similarly, we can prove that $\mu(i,j) - \tilde{\mu}(i,j) \leq \frac{\varepsilon}{2}(\mu(i,j) + (i+j)\underline{\mathsf{d}}/2n)$, and the lemma follows. ◀

▶ **Corollary 8.** $|\tilde{\mu}(m,n) - \mathsf{dtw}(P,Q)| \leq \varepsilon\mathsf{dtw}(P,Q)$.

We now describe how to implement the algorithm for computing each $\tilde{\mu}(i,j)$ efficiently.

Sorting points in $\mathcal{B}$. Using radix sort, we sort the points of $\mathcal{B}$ in (y,x) lexicographical order, where x and y denote the first and second coordinates of points, so that they are sorted in the order in which they are processed. We also perform the same radix sort for (x,y) and $(y-x,x)$ lexicographical orderings. For each point in $\mathcal{B}$, we add a pointer to the previous point in each of the three sorted orders, namely, a pointer to the first point below, to the left of, and on the lower-left diagonal of the current point. These pointers are used to identify the $\tilde{\mu}$ values required in (1).

Finding a rectangle hit by a point. The algorithm also needs to determine whether there exists a rectangle of $\mathcal{R}$ hit by (i,j). This can be achieved by maintaining the rectangle with the right-most right boundary when we traverse each row. More precisely, when processing the point $(i,j) \in \mathcal{B}$, we maintain a rectangle R_{curr} that is hit by (i,j) and whose right boundary spans the farthest; we denote the x-coordinate of the right boundary of R_{curr} by ξ_{curr}. If no rectangle hits (i,j), we set $\mathsf{R}_{curr} = \text{NULL}$. We update $\mathsf{R}_{curr}, \xi_{curr}$ while processing (i,j) as follows: If (i,j) is the left boundary point of a rectangle R with ξ being the x-coordinate of its right boundary and if $\xi > \xi_{curr}$, we set $\mathsf{R}_{curr} = \mathsf{R}$ and $\xi_{curr} = \xi$. Otherwise, if (i,j) is the right boundary point of R_{curr}, i.e., $i = \xi_{curr}$, we set $\mathsf{R}_{curr} = \xi_{curr} = \text{NULL}$. The total time spent at (i,j) is $O(1)$.

Range-min data structure. If (i,j) hits the rectangle R_{curr}, we compute $\tilde{\mu}(i,j)$ using (2). Without loss of generality, assume $i - x^- \geq j - y^-$. We divide the left and bottom boundary points that are dominated by (i,j) into three sets: $N_1 = \{(x^-, y) \mid y \in [y^- : j]\}$,

$N_2 = \{(x, y^-) \mid x \in [x^- : i - (j - y^-) - 1]\}$, and $N_3 = \{(x, y^-) \mid x \in [i - (j - y^-) : i]\}$. See Figure 2(b).

The optimal admissible path from $(1, 1)$ to (i, j) must pass through a point in $N_1 \cup N_2 \cup N_3$. So we compute $\tilde{\mu}(i, j)$ as follows:

$$\tilde{\mu}(i,j) = \min \left\{ \begin{array}{ll} (i - x^-)\omega_{\mathsf{R}} & + \min_{(a,b)\in N_1} \tilde{\mu}(a, b), \\ i\omega_{\mathsf{R}} & + \min_{(a,b)\in N_2} (\tilde{\mu}(a, b) - a\omega_{\mathsf{R}}), \\ (j - y^-)\omega_{\mathsf{R}} & + \min_{(a,b)\in N_3} \tilde{\mu}(a, b) \end{array} \right\}. \tag{3}$$

We compute the minimum value in the intervals N_1, N_2, and N_3 by performing range-min queries. We use the data structure proposed by Fischer and Heun [17] (see also [28]), which answers a range-min query in $O(1)$ time after linear time preprocessing. Thus, a range-min query on N_2 or N_3 can be answered in $O(1)$ time by constructing a static range-min data structure on the points on the bottom boundary of R_{curr} (all $\tilde{\mu}$ values for these points have been computed before visiting any point that hits R_{curr}). On the other hand, to support a range-min query on N_1, we need a range-min data structure on the left boundary points of R_{curr} that also supports inserting new points at the end when the $\tilde{\mu}$ values of the left boundary points are computed row by row.

We briefly describe the static data structure, and show how to extend it to support insertion in amortized $O(1)$ time. The input is an array of k real numbers. We say a data structure has time complexity $\langle p(k), q(k)\rangle$ if the preprocessing takes time $O(p(k))$ and each query takes time $O(q(k))$. The static data structure divides the array into blocks of size $b = \frac{1}{4}\log_2 k$. For each block, we construct a naive $\langle b^2, 1\rangle$-time data structure. Fischer and Heun show that many blocks can share the same data structure, so we create far fewer than $k/b = O(k/\log k)$ copies of the data structure. Next the algorithm computes the minimum for each block and uses an $\langle k \log k, 1\rangle$-time "exponential-range" data structure over the block minimums. We now describe each of the two structures in more detail.

A *Cartesian tree* of a length-b array stores the minimum element of the array at the root. The left (resp. right) child of the root is a Cartesian tree on the elements to the left (resp. right) of the minimum element. It can be built by a linear scan of the elements and pushing/popping elements into/from a stack at most $2b$ times; these push/pop operations serve as a fingerprint of the Cartesian tree. Thus, the number of different Cartisian trees for a length-b array is bounded by $2^{2b} = 4^b$. It turns out that all arrays of length b that have the same structured *Cartesian tree* [13] can share the same $\langle b^2, 1\rangle$-time data structure. We thus build 4^b copies of the $\langle b^2, 1\rangle$ data structure as follows: We go through each of the $O(k/\log k)$ blocks, and compute the fingerprint of the block in $O(b)$ time; if there is no data structure corresponding to the fingerprint, we build it in $O(b^2)$ time by computing the minimums for all possible $O(b^2)$ ranges.

The exponential-range data structure maintains the minimums of $O(\log k)$ ranges starting at each index $i \in [k]$ of exponentially increasing sizes $1, 2, 2^2, \ldots, 2^{\log k}$. Then the minimum of a range $[i, j]$ can be obtained by taking the minimum of two ranges $[i, i + 2^\alpha - 1]$ and $[j - 2^\alpha + 1, j]$, where α is the largest integer such that $2^\alpha \le j - i + 1$. The total preprocessing time is $O((k/b)\log(k/b) + 4^b b^2) = O(k)$.

To answer a range-min query, we compute the blocks containing the two end points of the query range; the minimum of the whole blocks in the range can be answered using the exponential-range data structure in $O(1)$ time; the minimums of the two partial blocks can also be answered in $O(1)$ time using the naive data structures associated with the two boundary blocks. So each query takes $O(1)$ time.

We now describe how to support inserting an element to the end of the array in amortized constant time. If the last block of the array contains less than b elements, the exponential-

range data structure remains the same, and we just need to update the fingerprint of the last block. We can encode the fingerprint information (a sequence of pushes and pops) as a path from the root to an internal node in a full binary tree of depth $2b$, where a push corresponds to branching to the left child and a pop corresponds to branching to the right child. At each node of the binary tree, we store a pointer to the corresponding naive range-min data structure. Recall that the Cartesian tree is built by a linear scan of the elements; so inserting a new element just means going one more level down the binary tree, which takes constant time. On the other hand, when the last block is already full, the newly inserted element starts a new block. In this case, we also need to update the exponential-range data structure, which takes $O(\log(k/b))$ time; but since this only happens every $O(b) = O(\log k)$ elements, the amortized time per insertion is still constant. Therefore, we can insert an element to the data structure in amortized $O(1)$ time.

▶ **Lemma 9.** *For all $(i, j) \in \mathcal{B}$, $\tilde{\mu}(i, j)$ can be computed in a total time of $O(|\mathcal{B}|)$.*

2.4 Running time analysis

We now bound the size of $|\mathcal{B}|$, which by Lemma 9 bounds the running time of step (iii) of the algorithm. A similar argument bounds the time spent in generating the set $\mathcal{F}$, which in turn bounds the running time of step (ii).

▶ **Lemma 10.** *The total number of points in $\mathcal{B}$ is $O(\frac{\kappa}{\varepsilon} n \log n)$, $O(\frac{\kappa^d}{\varepsilon^d} n \log n)$, and $O(\frac{1}{\varepsilon} n^{2-1/d} \log n)$ for κ-packed, κ-bounded and backbone sequences, respectively.*

Proof. Recall that for a node u, $\hat{\square}_u$ is the box of side length $2\Delta(u)$ and concentric with $\square_u$. For any well-separated pair of quadtree nodes $(u, v) \in \mathcal{F}$, let $\hat{m}_u = |P \cap \hat{\square}_u|, \hat{n}_v = |Q \cap \hat{\square}_v|$. Recall that A denotes the set of active nodes of quadtree $\mathcal{T}$, and α_u (resp. β_v) is the number of maximal contiguous subsequences of P in $\hat{\square}_u$ (resp. Q in $\hat{\square}_v$) computed by our algorithm. Then $\sum_{u \in A} \hat{m}_u \le 2^d \sum_{u \in A} m_u = O(m \log n)$. Let $N(u) = \{v \mid (u, v) \in \mathcal{F}\}$. The total number of rectangle boundary points is

$$|\mathcal{B}| \le 2 \sum_{(u,v) \in \mathcal{F}} (\hat{m}_u \beta_v + \alpha_u \hat{n}_v) = 2 \sum_{u \in A} \hat{m}_u \sum_{v \in N(u)} \beta_v + 2 \sum_{v \in A} \hat{n}_v \sum_{u \in N(v)} \alpha_u. \tag{4}$$

We show next that for any $u \in A$, $\sum_{v \in N(u)} \beta_v = O(\kappa/\varepsilon)$ for κ-packed sequences, $O(\kappa^d/\varepsilon^d)$ for κ-bounded sequences, and $O(n^{1-1/d}/\varepsilon)$ for backbone sequences. The first part of (4) is then bounded by $O(\frac{\kappa}{\varepsilon} n \log n)$, $O(\frac{\kappa^d}{\varepsilon^d} n \log n)$, and $O(\frac{1}{\varepsilon} n^{2-1/d} \log n)$ for κ-packed, κ-bounded, and backbone sequences. Symmetrically, the second part of (4) has the same bound, and the lemma follows.

We now bound $\sum_{v \in N(u)} \beta_v$ for any $u \in A$. By Lemma 5, there exists a constant c such that for any $v \in N(u)$, $\hat{\square}_v$ is contained in a ball B concentric with $\square_u$ of radius $\frac{c}{\varepsilon}\Delta(u)$.

There are two types of maximal contiguous subsequence $[y_b^- : y_b^+]$ of $Q \cap \hat{\square}_v$ computed by our algorithm: (i) $q_{y_b^+} = q_n$ is the last point of Q, and (ii) $q_{y_b^+ + 1}$, the point of Q after the last point in the MCS lies outside of $\hat{\square}_v$. The first type of MCS is bounded by the number of v's such that $\hat{\square}_v$ contains the last point of Q, q_n. Suppose node u is at level t of the quadtree. By Lemma 5(ii), such v's can be from levels $t-1, t, t+1$. Moreover, at each level, q_n can be in the $\hat{\square}_v$ of at most 2^d v's. Thus, the number of maximal contiguous subsequences of the first type is at most $2^d \times 3 = O(1)$. In the following, we bound the second type separately for each family of input sequence.

κ-packed sequences. Since the MCS starts inside $\square_v$ and leaves $\hat{\square}_v$ before the next MCS of $S_v(Q)$ starts, the length of γ_Q between $q_{y_b^-}$ and $q_{y_b^+}$ is at least $\Delta(v)/2$ (see Figure 3). Let $\hat{L}_v$ be the length of $\gamma_Q \cap \hat{\square}_v$. Then

$$\sum_{v \in N(u)} \beta_v \leq O(1) + \sum_{v \in N(u)} \frac{\hat{L}_v}{\Delta(v)/2} \leq O(1) + \frac{4}{\Delta(u)} \sum_{v \in N(u)} \hat{L}_v.$$

The last inequality follows from Lemma 5(ii). Because of the following four conditions– the side length of $\hat{\square}_u$ is twice that of $\square_u$, the nodes in $N(u)$ belong to three levels of $\mathcal{T}$ (Lemma 5(ii)), the cells of the nodes at the same level of $\mathcal{T}$ are disjoint, and $\hat{\square}_v \subseteq \mathsf{B}$ for all $v \in N(u)$–we can conclude that

$$\sum_{v \in N(u)} \hat{L}_v \leq 3 \cdot 2^d |\gamma_Q \cap \mathsf{B}| \leq \frac{3 \cdot 2^d c\kappa}{\varepsilon}.$$

The last inequality follows because P is κ-packed sequence. Hence $\sum_{v \in N(u)} \beta_v = O(\kappa/\varepsilon)$, as claimed.

κ-bounded sequences. We first show that for any two MCSs $[y_1^- : y_1^+]$ and $[y_2^- : y_2^+]$, $||q_{y_1^-} q_{y_2^-}|| \geq \Delta(v)/(\kappa + 2)$. This is because between points $q_{y_1^-}$ and $q_{y_2^-}$, the curve γ_Q goes from inside $\square_v$ to outside $\hat{\square}_v$ which spans distance at least $\Delta(v)/2$. Let q be the intersection of this portion of γ_Q with the boundary of $\hat{\square}_v$. By κ-boundedness, $\gamma_Q(y_1^- : y_2^-) \subseteq \mathsf{B}(q_{y_1^-}, \frac{\kappa}{2}||q_{y_1^-} q_{y_2^-}||) \cup \mathsf{B}(q_{y_2^-}, \frac{\kappa}{2}||q_{y_1^-} q_{y_2^-}||)$. Therefore, $(1 + \kappa/2)||q_{y_1^-} q_{y_2^-}|| \geq ||q_{y_1^-} q|| \geq \Delta(v)/2$, and the claim follows. By a packing argument, the number of MCSs in $\hat{\square}_v$ is bounded by $O(\kappa^d)$. Finally, $|N(u)| = O(1/\varepsilon^d)$ by another packing argument in the ball B. So the number of second-type MCSs in all $\hat{\square}_v$'s for $v \in N(u)$ is $O(\kappa^d/\varepsilon^d)$.

Backbone sequences. By the property that two consecutive points on a backbone sequence have distance between c_1 and c_2, there must exist one point on any MCS in the shell along the boundary of $\square_v$ with thickness c_2. The volume of the shell is $O(\Delta(v)^d - (\Delta(v) - c_2)^d) = O(\Delta(v)^{d-1})$. Furthermore, any two points on Q are at least distance 1 apart. So the number of MCSs is bounded by $O(\Delta(v)^{d-1})$. Since $|N(u)| = O(1/\varepsilon^d)$, the number of MCSs in all $\hat{\square}_v$'s for $v \in N(u)$ is $O(\frac{\Delta(v)^{d-1}}{\varepsilon^d})$. On the other hand, each second-type MCS consumes a portion of γ_Q of length at least $\Delta(v)/2$; this means that the subsequence contains $\Omega(\Delta(v)/c_2) = \Omega(\Delta(v))$ points of Q. Since there are a total of n points in Q, the total number of MCSs in all $\hat{\square}_v$'s with $(u, v) \in \mathcal{F}$ is $O(\frac{n}{\Delta(v)})$. The worst case happens when the two upper bounds are balanced; in other words $\frac{\Delta(v)^{d-1}}{\varepsilon^d} = (\frac{n}{\Delta(v)})$ or $\Delta(v) = \varepsilon n^{1/d}$. The total number of second-type MCSs is $O(\frac{1}{\varepsilon} n^{1-1/d})$. ◀

To bound the running time for constructing the family $\mathcal{F}$ for each active node $u \in A$, we bound the number of times PAIRING(u, v) is called for some $v \in A$. Following the same argument as in the proof of Lemma 10, we can show that the time for constructing $\mathcal{F}$ is $O(\frac{\kappa}{\varepsilon} n \log n)$, $O(\frac{\kappa^d}{\varepsilon^d} n \log n)$, and $O(\frac{1}{\varepsilon} n^{2-1/d})$ for κ-packed, κ-bounded, and backbone sequences, respectively. Combining this bound with Lemmas 4, 9, and 10, we obtain Theorem 1.

3 Edit Distance

We now show how our DTW algorithm can be extended to compute a $(1+\varepsilon)$-approximate value of $\mathsf{ed}(P,Q)$. Define $\mathsf{V} = [m+1] \times [n+1]$. For $i < m+1$ and $j < n+1$, we have $\omega(i,j) = ||p_i q_j||$. Otherwise $\omega(i,j)$ is undefined (e.g., $\omega(m+1,\cdot)$ and $\omega(\cdot,n+1)$ are undefined). We add an edge between every pair of neighboring points in V. The weight of a horizontal or vertical edge is set to g and the weight of a diagonal edge $\langle (i,j),(i+1,j+1)\rangle$ is set to $\omega(i,j)$. The weight of an admissible path π is defined as the sum of weights of the edges along π. As earlier, we define $\mu(i,j)$ to be the minimum weight of an admissible path from $(1,1)$ to (i,j). Then $\mathsf{ed}(P,Q) = \mu(m+1,n+1)$.

We compute an approximate value of $\mathsf{ed}(P,Q)$ using the same 4-step algorithm as for $\mathsf{dtw}(P,Q)$, with a different implementation of each step. Step (ii) of the algorithm remains the same, except that we add all the points on the $(m+1)$-st column and $(n+1)$-st row of V to $\mathcal{B}$. In the following, we give a simple $O(n)$-approximation for step (i), and point out modifications needed in step (iii) to compute a value $\tilde{\mu}(i,j)$ for every $(i,j) \in \mathcal{B}$, such that $\mu(i,j) \le \tilde{\mu}(i,j) \le (1+\varepsilon)\tilde{\mu}(i,j)$.

***O*(*n*)-approximation.** If $m = n$ and the monotone path $\pi = \langle (1,1),(2,2),\ldots,(n+1,n+1)\rangle$ has total weight at most g, then it is returned as the optimal path. Otherwise, $\mathsf{ed}(P,Q) \ge g$, and we set $\underline{\mathsf{d}} = g$. Since $\mathsf{ed}(P,Q)$ is no larger than the weight of an all-gap admissible path from $(1,1)$ to $(m+1,n+1)$, we have $\mathsf{ed}(P,Q) \le 2(m+n)g \le \overline{\mathsf{d}} = 4n\underline{\mathsf{d}}$.

Computing admissible paths. We describe how to compute $\tilde{\mu}(i,j)$, for all $(i,j) \in \mathcal{B}$, in the same row by row order. The main difference from DTW is that, since ED allows gaps, it is possible for the optimal admissible path between $(1,1)$ and (i,j) to have rectilinear subpaths through grid points that are not covered by any rectangle. As in Section 2.3, we consider whether there there exists a rectangle hit by (i,j).

First, assume there exists a rectangle $\mathsf{R} = [x^- : x^+] \times [y^- : y^+] \in \mathcal{R}$ hit by (i,j). Similar to DTW, we divide the relevant points on the left and bottom boundaries of R into three disjoint subsets N_1, N_2 and N_3. If $2g \le \omega_{\mathsf{R}}$, it is always preferable to take a rectilinear path inside R. Thus we can assume the admissible path to (i,j) goes through either (x^-,j) or (i,y^-), and we set $\tilde{\mu}(i,j) = \min\{\tilde{\mu}(x^-,j) + (i-x^-), \tilde{\mu}(i,y^-) + (j-y^-)\}$. If $2g > \omega_{\mathsf{R}}$, the minimum-weight path inside R should take as many diagonal steps as possible. So we set

$$\tilde{\mu}(i,j) = \min \left\{ \begin{array}{l} j\omega_{\mathsf{R}} + (i-j)g + \min_{(a,b)\in N_1}(\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})b), \\ j\omega_{\mathsf{R}} + (i-j)g + \min_{(a,b)\in N_2}(\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})b - ag), \\ i\omega_{\mathsf{R}} + (j-i)g + \min_{(a,b)\in N_3}(\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})a) \end{array} \right\}.$$

We use the same range-min data structure of Fischer and Heun to compute each $\tilde{\mu}(i,j)$ in amortized $O(1)$ time. The key used for the data structure is $\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})b$, $\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})b - ag$, and $\tilde{\mu}(a,b) + (g-\omega_{\mathsf{R}})a$ for N_1, N_2, and N_3, respectively.

Next, assume (i,j) does not hit any rectangle in $\mathcal{R}$. If $\{(i-1,j),(i,j-1),(i-1,j-1)\} \subset \mathcal{B}$, and thus their $\tilde{\mu}$ values have been computed, it is trivial to compute $\tilde{\mu}(i,j)$ in $O(1)$ time. We now focus on the case where one of the predecessors of (i,j) is not in $\mathcal{B}$. Let $U = \bigcup_{\mathsf{R}\in\mathcal{R}} \mathsf{R}$ denote the union of all rectangles in $\mathcal{R}$. A point $(h,k) \in U$ is on the boundary of U, denoted by ∂U, if (h,k) does not lie in the interior of any rectangle of $\mathcal{R}$; so at least one point of $\{(h-1,k),(h+1,k),(h,k-1),(h,k+1)\}$ is not in U. Consider any admissible path π from $(1,1)$ to (i,j) whose total weight is at most $\overline{\mathsf{d}}$. Let (a,b) denote the last point of $\mathcal{B}$ on π before reaching (i,j). The subpath of π between (a,b) and (i,j) must be outside U, and it can only

contain gaps since the weight of any point outside U is greater than $\overline{\mathsf{d}}$, except the first step out of (a, b), which costs $\omega(a, b)$ if it is diagonal. Let (i_0, j) (resp. (i, j_0)) be the first point of $\mathcal{B}$ to the left of (resp. below) (i, j) on row j (resp. column i). Let $\partial U_{ij} = [i-1] \times [j-1] \cap \partial U$ denote the points on ∂U that are lower-left of (i, j). We set

$$\tilde{\mu}(i,j) = \min \left\{ \begin{array}{l} \tilde{\mu}(i_0, j) + (i - i_0)g, \\ \tilde{\mu}(i, j_0) + (j - j_0)g, \\ \min_{(a,b)\in\partial U_{ij}} (\tilde{\mu}(a, b) + (i + j - a - b)g + \min(0, \omega(a, b) - 2g)) \end{array} \right\}. \quad (5)$$

To compute $\tilde{\mu}(i, j)$, we use a different and simpler range-min data structure for ∂U with key $\tilde{\mu}(a, b) - (a + b)g + \min(0, \omega(a, b) - 2g)$, that supports the decrease-key and query operations in $O(\log n)$ time each. More specifically, we maintain a minimum over points of ∂U in each column as we traverse $\mathcal{B}$ row by row. We maintain the column minimums in a complete binary tree where each leaf corresponds to a column and an internal node stores the minimum over the leaves of the subtree rooted at that node. Note that the column minimums are always non-increasing while we perform the row by row computations. When the minimum of a column corresponding to some leaf v gets decreased, we update the minimum information stored at each node along the path from v to the root of the binary tree. This takes $O(\log n)$ time. The last term in (5) can be computed by querying the the complete binary tree with range $[i-1]$ in $O(\log n)$ time. Following similar arguments to those in the proof of Lemma 10, we can show the following lemma.

▶ **Lemma 11.** *The number of points on ∂U is $O(\frac{\kappa}{\varepsilon} n)$, $O(\frac{\kappa^d}{\varepsilon^d} n)$, and $O(\frac{1}{\varepsilon^{1-1/(d+1)}} n^{2-1/(d+1)})$ for κ-packed, κ-bounded, and backbone sequences, respectively.*

Both the number of updates and the number of queries in the binary tree are bounded by $|\partial U|$, and each update or query takes $O(\log n)$ time. Moreover, the case when there exists a rectangle hit by the current point takes the same time as for DTW. Theorem 2 follows.

4 Conclusion

In this paper, we presented $(1 + \varepsilon)$-approximate algorithms for computing the dynamic time warping (DTW) and edit distance (ED) between a pair of point sequences. The running time of our algorithms is near-linear when the input sequences are κ-packed or κ-bounded, and subquadratic when the input sequences are protein backbone sequences. Our algorithms are the first near-linear or subquadratic-time algorithms known for computing DTW and ED for "well-behaved" sequences. One interesting open question is whether there exists a near-linear algorithm for computing DTW and ED for backbone sequences in $\mathbb{R}^2$. Another interesting open problem is to identify other dynamic-programming based geometric optimization problems that can be solved using our approach, i.e., visiting a small number of entries of the dynamic programming table using geometric properties of the input.

Acknowledgements. The authors thank Sariel Har-Peled for many helpful comments.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. 56th Annu. IEEE Sympos. on Found. Comp. Sci.*, pages 59–78, 2015.

2 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st Int. Colloq. on Auto., Lang., and Program.*, pages 39–51, 2014.

3 Helmut Alt, Christian Knauer, and Carola Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

4 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proc. 51st IEEE Sympos. on Found. Comp. Sci.*, pages 377–386, 2010.

5 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *Proc. 14th Annu. Euro. Sympos. Algo.*, pages 52–63, 2006.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th Annu. ACM Sympos. Theory of Comput.*, pages 51–58, 2015.

7 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th. Annu. IEEE Sympos. on Found. of Comp. Sci.*, pages 661–670, 2014.

8 Karl Bringmann and Marvin Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds. In *Proc. 26th Annu. Int. Sympos. Algo. Comp.*, pages 517–528, 2015.

9 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th Annu. IEEE Sympos. on Found. of Comp. Sci.*, pages 79–97, 2015.

10 Karl Bringmann and Wolfgang Mulzer. Approximability of the discrete Fréchet distance. *J. Comput. Geom.*, 7(2):46–76, 2016.

11 Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *Proc. 11th Annu. Euro. Conf. Comp. Vis.*, pages 282–295, 2010.

12 Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995.

13 Cartesian tree. `https://en.wikipedia.org/wiki/Cartesian_tree`.

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 *Proc. SIGCHI Conf. Human Factors in Comp. Sys.*, pages 987–996, 2012.

15 Anne Driemel, Sariel Har-Peled, and Carola Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete Comp. Geom.*, 48(1):94–127, 2012.

16 Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Camridge University Press, New York, 1998.

17 Johannes Fischer and Volker Heun. Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. In *Proc. 17th Annu. Sympos. Combin. Pattern Match.*, pages 36–48, 2006.

18 T. Gasser and K. Wang. Alignment of curves by dynamic time warping. *Annals of Statistics*, 25(3):1251–1276, 1997.

19 M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

20 Sariel Har-Peled. *Geometric Approximation Algorithms*. American Mathematical Society Providence, 2011.

21 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comp. Sys. Sci.*, 62(2):367–375, 201.

22 Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005.

23 Rachel Kolodny, Patrice Koehl, and Michael Levitt. Comprehensive evaluation of protein structure alignment methods: Scoring by geometric measures. *J. Mol. Bio.*, 346(4):1173–1188, 2005.

24 J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pages 593–604, 2007.

25 Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag Berlin Heidelberg, 2007.

26 Mario E. Munich and Pietro Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proc. 7th Annu. Int. Conf. Comp. Vis.*, pages 108–115, 1999.

27 L.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. PTR Prentice Hall, 1993.

28 Range minimum queries: Part two. `http://web.stanford.edu/class/archive/cs/cs166/cs166.1146/lectures/01/Slides01.pdf`.

29 Time series matching with dynamic time warping. `https://systematicinvestor.wordpress.com/2012/01/20/time-series-matching-with-dynamic-time-warping/`.

30 Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

An Improved Lower Bound on the Minimum Number of Triangulations

Oswin Aichholzer[*1], Victor Alvarez[2], Thomas Hackl[†3], Alexander Pilz[‡4], Bettina Speckmann[§5], and Birgit Vogtenhuber[6]

1 **Institute for Software Technology, TU Graz, Graz, Austria**
oaich@ist.tugraz.at
2 **Department of Computer Science, TU Braunschweig, Braunschweig, Germany**
alvarez@ibr.cs.tu-bs.de
3 **Institute for Software Technology, TU Graz, Graz, Austria**
thackl@ist.tugraz.at
4 **Institute for Software Technology, TU Graz, Graz, Austria**
apilz@ist.tugraz.at
5 **Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
b.speckmann@tue.nl
6 **Institute for Software Technology, TU Graz, Graz, Austria**
bvogt@ist.tugraz.at

Abstract

Upper and lower bounds for the number of geometric graphs of specific types on a given set of points in the plane have been intensively studied in recent years. For most classes of geometric graphs it is now known that point sets in convex position minimize their number. However, it is still unclear which point sets minimize the number of geometric triangulations; the so-called double circles are conjectured to be the minimizing sets. In this paper we prove that any set of n points in general position in the plane has at least $\Omega(2.631^n)$ geometric triangulations. Our result improves the previously best general lower bound of $\Omega(2.43^n)$ and also covers the previously best lower bound of $\Omega(2.63^n)$ for a fixed number of extreme points. We achieve our bound by showing and combining several new results, which are of independent interest:

1. Adding a point on the second convex layer of a given point set (of 7 or more points) at least doubles the number of triangulations.
2. Generalized configurations of points that minimize the number of triangulations have at most $\lfloor n/2 \rfloor$ points on their convex hull.
3. We provide tight lower bounds for the number of triangulations of point sets with up to 15 points. These bounds further support the double circle conjecture.

1998 ACM Subject Classification G.2.1 Combinatorics – Combinatorial algorithms

Keywords and phrases Combinatorial geometry, Order types, Triangulations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.7

[*] O. A. partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18.
[†] T. H. supported by the Austrian Science Fund (FWF): P23629-N18 'Combinatorial Problems on Geometric Graphs'.
[‡] A. P. partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18.
[§] B. S. supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.023.208.

© Oswin Aichholzer, Victor Alvarez, Thomas Hackl, Alexander Pilz, Bettina Speckmann, and Birgit Vogtenhuber;
licensed under Creative Commons License CC-BY

1 Introduction

Bounding the number of geometric graphs of a specific type on a given set of n points in the plane is a very active research topic in the field of combinatorial geometry. The first results [5] on bounding the number of plane Hamiltonian cycles were obtained already in the 1970s. Of particular interest are bounds on $T(n)$, the largest number of triangulations any set of n points can have, since bounds on $T(n)$ can be used to derive bounds on other graph classes (see, for example, [13, 18, 21]). The current bounds stem from Dumitrescu et al. [8], who constructed a set of n points with $\Omega(8.65^n)$ triangulations, and Sharir and Sheffer [20] who proved $T(n) < 30^n$.

Bounds on the minimum number of geometric graphs on a given point set have received comparably less attention, since for many classes of geometric graphs it is known that point sets in convex position minimize their number [2]. However, points in convex position do *not* minimize the number of triangulations. Specifically, points in convex position have C_{n-2} triangulations (where $C_m \in \Theta(4^m/m^{3/2})$ is the mth Catalan number) and the so-called *double circles* have only $\sqrt{12}^{n-\Theta(\log n)}$ triangulations [19]. A *double circle* of size $n = 2h$ has h extreme points, and each edge ab on the convex hull boundary is assigned a different inner point q_{ab} s.t. there is no segment between two points of the double circle crossing the segments $q_{ab}a$ or $q_{ab}b$.

Aichholzer, Hurtado, and Noy [3] proved the first non-trivial asymptotic lower bound on $t(n)$, the smallest number of triangulations any set of n points can have, namely $t(n) \in \Omega(2.33^n)$. They conjectured that $t(n)$ equals the number of triangulations of the double circle. This conjecture is further supported by the results in this paper. The best current bounds on $t(n)$ are as follows: For h extreme points and i inner points, McCabe and Seidel [15] proved $t(n) \in \Omega(2.\overline{72}^h \cdot 2.2^i)$ and $t(n) \in \Omega(2.63^i)$ when h is a constant. For general point sets, Sharir, Sheffer, and Welzl [22] proved $t(n) \in \Omega(2.4317^n)$.

In this paper we prove the following theorem:

▶ **Theorem 1.** *Let $t(n)$ be the smallest number of triangulations any set of n points in general position in the plane can have. Then $t(n) \in \Omega(2.631035^n)$ for $n \geq 35064$.*

Our proof uses two basic ingredients: (1) a structural theorem by Aichholzer et al. [3], which provides the mechanism to obtain asymptotic bounds from a large induction base, and (2) tight lower bounds for the number of triangulations of point sets with up to 15 points. These tight lower bounds, together with a combination of known general bounds (Section 5), feed into the structural theorem which then proves Theorem 1 (Section 6).

Computing tight lower bounds on the minimum number of triangulations for $12 \leq n \leq 15$ by brute-force is impossible with current technology. We hence prove two new structural results which we believe to be of independent interest:

1. Adding a point on the second convex layer of a given point set (of 7 or more points) at least doubles the number of triangulations (Section 2).
2. Generalized configurations of points that minimize the number of triangulations have at most $\lfloor n/2 \rfloor$ points on their convex hull (Section 3).

These results allow us to significantly reduce the number of point configurations to consider and to parallelize computations (Section 4). We could hence complete the automated part of our proof in roughly two months of computation time using up to 128 CPUs (which amounts to several hundred thousand CPU hours). After giving some necessary definitions and background, we describe our proof and computation strategy in more detail below.

Definitions, notation, and background. Let S be a finite point set in general position in the plane, that is, no three points of S lie on a line. A *triangulation* of S is a maximal plane graph on S. We denote the set of all triangulations of S with $\mathcal{T}(S)$ and the number of triangulations of S with $\mathrm{tr}(S) = |\mathcal{T}(S)|$.

The infinite family of sets of n points in general position in the plane can be partitioned into a finite number of equivalence classes by their order types. Two point sets have the same *order type* if there exists a bijection between them such that all point triples have the same orientation [11], that is, clockwise, counterclockwise, or collinear. Point sets with the same order type share many important properties. For example, they contain the same points on the convex hull boundary and the same pairs of edges cross. In particular, the number of triangulations is the same for all point sets of the same order type.

There is a finite number of order types with n points, so it is possible to enumerate them (see the database of all point set order types for up to 11 points [1, 4]). Points in the plane can be mapped to arrangements of lines, whose relative position also determines the orientation of each triple, and hence the order type. Line arrangements can be generalized to *pseudo-line arrangements*, that is, x-monotone curves that intersect pairwise exactly once in a proper crossing. We call the equivalence classes obtained from triple orientations in pseudo-line arrangements *abstract order types*. Abstract order types can be realized by *generalized configurations of points* [9], which are point sets where every pair of points is connected by exactly one pseudo-line. A point triple (a, b, c) is oriented clockwise iff point c is in the half-plane to the right of the directed pseudo-line through a and b. Many concepts involving point sets, like triangulations, can easily be abstracted to generalized configurations of points. An abstract order type is *realizable* if it is the order type of a point set. All abstract order types with up to eight points are realizable [10]. See [12] for further details.

Proof and computation strategy. The model of pseudo-line arrangements is the crucial tool for exhaustive enumeration of (abstract) order types in [1], and the generation of larger sets fulfilling certain properties [4]. A given representation of a pseudo-line arrangement is augmented by an additional pseudo-line in all possible ways. We use a similar strategy to compute tight lower bounds on the number of triangulations of point sets with up to 15 points: Suppose we know that, when adding a point to a point set (a pseudo-line to a pseudo-line arrangement), the number of triangulations increases by at least a factor α. To verify whether there exists an abstract order type S with n points with $\mathrm{tr}(S) \leq b$, we need to only extend those sets of size $n-1$ that have at most b/α triangulations (and for these, extend only sets of size $n-2$ with at most b/α^2 triangulations and so on). The basic idea is to select all order types of size n_0 ($n_0 = 8$ in our case) that have at most b/α^{n-n_0} triangulations, and extend them to order types of size n, provided that the number of triangulations of the intermediate order types are within the bounds.

We are facing various challenges with this strategy due to the vast number of abstract order types. First of all, we need the factor α to be as large as possible. The work by Sharir, Sheffer, and Welzl [22] on vertices with degree 3 in random triangulations implies that every point set contains at least one point which one can remove to reduce the number of triangulations by 1/2 (see Section 5 for details). However, their result does *not* show which point to remove (or add in our strategy). An abstract order type of size n can be obtained from n different (parent) order types of size $n-1$. If we do not know which point increases the number of triangulations by a factor $\alpha \geq 2$, we have to extend each order type in all possible ways. Such extensions are computationally infeasible, since we can expect an order type to be created close to $n!/n_0!$ times. However, if we know *which* point gives a factor of at

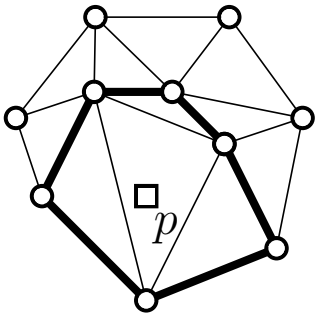


Figure 1 A convex 6-star for p.

least α, we can identify a *unique* parent order type and compute its number of triangulations only if this order type of size n is to be further extended. In Section 2 we prove that adding a point on the second convex layer of the order type increases the number of triangulations by a factor of at least 2. This structural result gives us the necessary control over the extensions and allows us to check the number of triangulations of each *relevant* order type of size n_0 to n only once. We can hence distribute the work load among different independent processes, each handling a fixed disjoint set of order types.

In principle it is not sufficient to extend order types by interior points only. However, in Section 3 we prove that we need to consider only abstract order types with at most $\lfloor n/2 \rfloor$ points on the convex hull. Since $\lfloor 15/2 \rfloor < 8$ we can start our extension from $n_0 = 8$ and extend by adding interior points only. For $n = 8$ all abstract order types are realizable. Hence our improved bound actually applies to all abstract order types (and not only to point sets). Finally, since all abstract order types with the minimum number of triangulations for $12 \leq n \leq 15$ are realizable as point sets, our results support the double circle conjecture.

2 A factor 2 for points on the second convex layer

Let S be a set of $n \geq 7$ points in general position in the plane. We denote the convex hull of S by $\mathrm{CH}(S)$ and the extremal points of S, that is, the points of S which lie on the boundary of $\mathrm{CH}(S)$, with $\mathrm{extr}(S)$. We say that a point $p \notin S$, with $(S \cup \{p\})$ in general position, is *interior-extremal* for S if $p \in \mathrm{extr}\left((S \cup \{p\}) \setminus \mathrm{extr}(S \cup \{p\})\right)$. That is, p is an extremal point of the second convex layer of the extended set $(S \cup \{p\})$.

Let $T \in \mathcal{T}(S)$ be an arbitrary triangulation of S. We call a simple polygon whose boundary is formed by edges from T and which does not contain vertices of S in its interior an *empty polygon of* T. For a point $p \notin S$ let P be an empty polygon of T formed by k edges such that p lies in the interior of P, and P is star shaped with respect to p. Then we call P a *k-star of T for p* (see Figure 1). An *edge-flip* (or flip) in a triangulation T is the operation of first removing an inner edge e from T such that the two triangles incident to e form a convex quadrilateral Q, and then inserting the opposite diagonal of Q into T.

Due to space constraints we omit the proof of the following lemma.

▶ **Lemma 2.** *Let $p \notin S$ be an interior-extremal point for S, T a triangulation of S, and P a convex k-star of T for p with $k < |S|$. Let $S' = S \cup \{p\}$ and let T' be the triangulation of S' obtained from T by removing all edges in the interior of P and adding all edges connecting p to vertices of P. Then at least one edge of the boundary of P can be flipped in T'.*

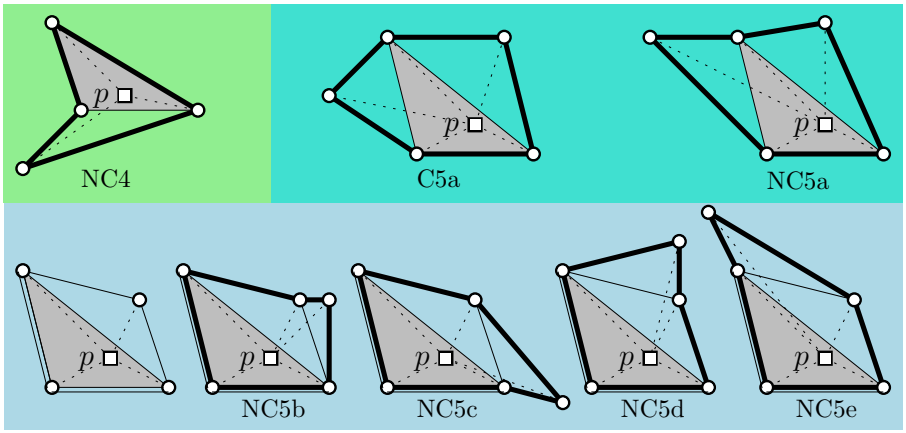


Figure 2 Certificates. Blocked edges are drawn with a double line.

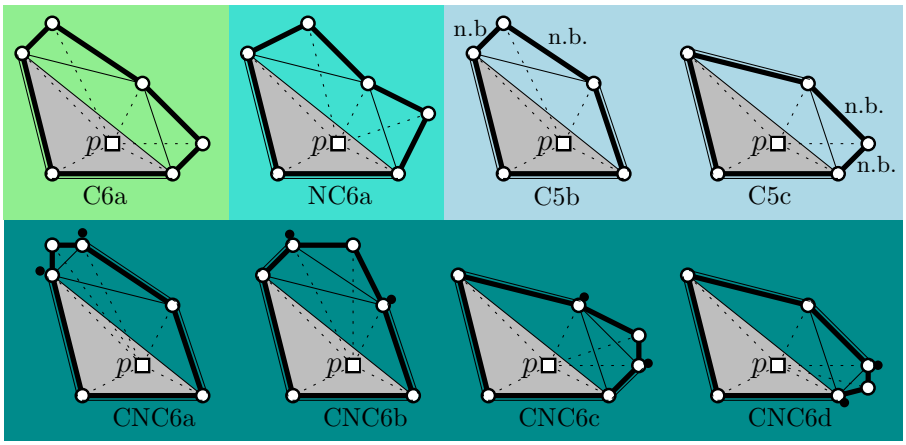


Figure 3 Certificates. Blocked edges are drawn with a double line, non-blocked edges are marked with n.b. The vertices of Certificates CNC which are marked with a black dot can be convex or non-convex.

2.1 Certificates for triangulations

We now argue that for any given interior-extremal point $p \notin S$, any triangulation $T \in \mathcal{T}(S)$ contains a star for p of a certain type (see below) which we call the certificate of T. We will argue in an algorithmic way, flipping edges of T to p in order to increase the degree of p. However, note that the certificate refers to the original triangulation T. See Figures 2 and 3 for an illustration of all certificates.

First add p to S and connect p with three edges to the corners of the triangle Δ of T in which p lies to obtain a triangulation T' of $S \cup \{p\}$. By Lemma 2 we can flip at least one edge of Δ, so that p has degree 4. If there exists one flip such that the resulting 4-star is non-convex, then this is the certificate of T (Figure 2: NC4). Otherwise, if at least two of the edges of Δ can be flipped (each flip is in a convex 4-star), we get a (convex or non-convex) 5-star which we take as the certificate of T (Figure 2: C5a and NC5a).

If both previous situations do not exist, then we flip the unique flippable edge to obtain a convex 4-star (c.f. Figure 2, lower left), where the two edges of the boundary near p are *blocked*. We call an edge on the boundary of a star for p blocked, if this edge cannot be

flipped to p in the underlying triangulation. Note that p can be placed anywhere in the shaded triangle.

Again by Lemma 2 we can flip at least one of the non-blocked edges of this 4-star. If there exists one flip such that the resulting 5-star is non-convex, then we take this as the certificate of T (Figure 2: NC5b to NC5e). Otherwise, if two edges of the 4-star can be flipped (each flip is in a convex 5-star), we get a (convex or non-convex - in the latter case it is important that we do not have collinear points) 6-star which we take as the certificate of T (Figure 3: C6a and NC6a).

In the remaining case, three edges of the unique convex 4-star are blocked. We make the unique possible flip (exists by Lemma 2), obtaining a convex 5-star, where at least three edges are blocked (those, which have already been blocked for the 4-star). If the remaining two edges are non-blocked, then this is our certificate for T (Figure 3: C5b and C5c). Otherwise, we again make the unique flip (exists by Lemma 2), and obtain a convex or non-convex 6-star, where at least four edges are blocked. This will be the certificate of T (Figure 3: CNC6a to CNC6d).

The above argumentation shows that any triangulation T contains a certificate.

▶ **Lemma 3.** *Let $p \notin S$ be an interior-extremal point for S and T a triangulation of S. Then T contains a certificate w.r.t. p out of the above list.*

Note that the triangulation T might have more than one certificate, as the above analysis just shows that every triangulation contains a certificate, but does not imply that this is unique. However, it will become clear in the next sections that we can choose any existing certificate for T without causing problems.

2.2 Distributing weights

Let $p \notin S$ be an interior-extremal point for S and let $S' = S \cup \{p\}$. We distribute weight at most one for each triangulation of S' to triangulations of S in a specific way (described below). Initially, all triangulations of S have weight zero.

Let T' be a triangulation of S' and let d be the degree of p in T'. Remove p and all its incident edges from T' and let P be the polygon in T' bounding the resulting face. Further, let $G(T')$ be the set of triangulations of S which are obtained from this construction by retriangulating the interior of P in all possible ways. For any triangulation of $G(T')$, P is by construction a d-star of p. We make a case analysis over the degree d of p in T'.

$d = 3$. T' adds weight 1 to the unique triangulation in $G(T')$.

$d = 4$. If P is non-convex, then assign weight 1 to the unique triangulation in $G(T')$. Otherwise assign weight $\frac{1}{2}$ to both triangulations in $G(T')$.

$d = 5$. If P is non-convex, then it can be triangulated in at most 3 different ways. If there are at most two triangulations then we distribute weight $\frac{1}{2}$ to each. If there are three triangulations of P then P has one reflex vertex and 4 diagonals lie in the interior of P. The weight is distributed as indicated in Figure 4, where only those triangulations get weight $\frac{1}{2}$ each, where p lies in the shaded area. Note that in this way a total weight of at most 1 is distributed.

If P is convex, then there are three different positions where p can lie within P; c.f. Figure 5. If p lies in the central area we do no distribute any weight, and if p lies in the area near a corner c of P we distribute weight $\frac{1}{2}$ to the two triangulations where c is not incident to a diagonal (Cases 5D and 5E). Otherwise, p lies near an edge e, i.e., in the intersection of the two neighboring ears in P. Depending on the blocked versus non-blocked pattern of the edges of P near e, we distinguish 5 disjoint cases as shown in

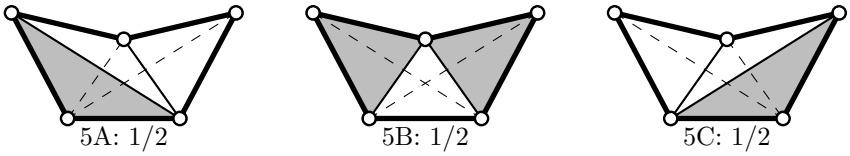


Figure 4 Non-convex 5-star weight distributing.

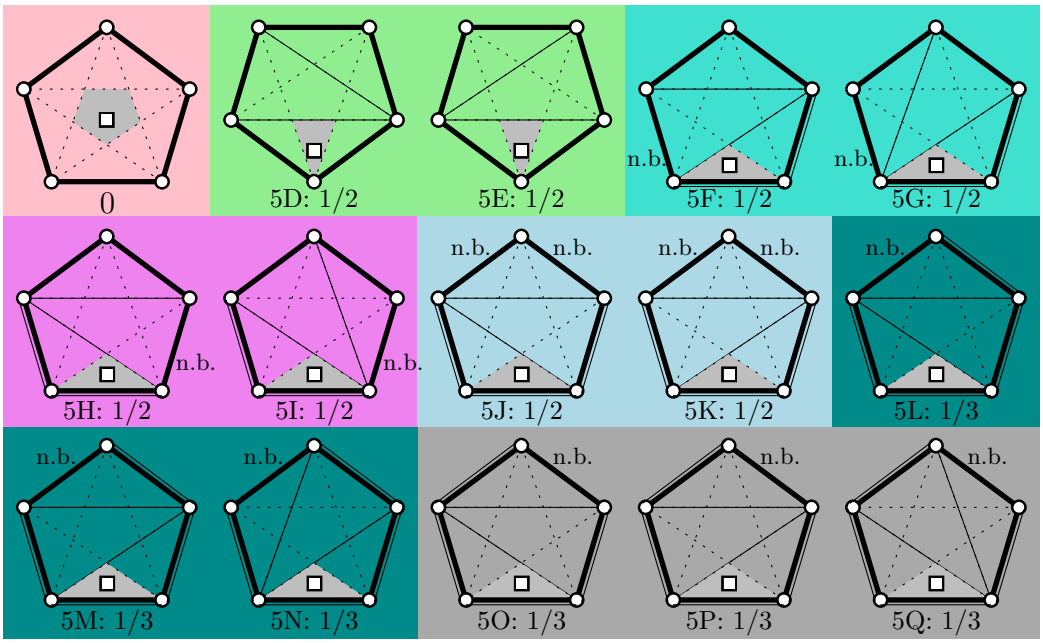


Figure 5 Convex 5-star weight distributing.

Figure 5. In the first three cases we distribute weight $\frac{1}{2}$ to two triangulations each, and in the remaining 2 cases weight $\frac{1}{3}$ to three triangulations each.

We remark that we distribure weights only for cases depicted in Figures 4 and 5.

$d = 6$. Similar as before we only distribute weight if p lies near an edge e, i.e., in the intersection of the two neighboring ears in P. Moreover, the vertices of e must be convex corners of P, while we do not make any assumption about the other vertices. We give weight $\frac{1}{6}$ to 6 different triangulations as indicated in Figure 6. Note that if P has reflex vertices then some of these triangulations do not exist and we simply waste the weight assigned to them.

$d \geq 7$. We do not assign any weight for T'.

Observe that in all cases we assign at most weight 1 for T', and that there are several cases where we assign weight less than one for T'. We denote with W the total sum of all distributed weights and have the following lemma.

▶ **Lemma 4.** *For $|S| > 6$ we have $W < |\mathcal{T}'(S')|$.*

The strict inequality comes from the fact that there is at least one triangulation $T' \in \mathcal{T}'(S')$ where p has degree 7 or more, for which we did not distribute any weight. Actually, for $|S| \geq 6$ it holds that $W \leq |\mathcal{T}'(S')| - (|S| - 6)$, as for any $7 \leq d \leq |S|$ we can construct a triangulation of $\mathcal{T}'(S')$ where p has degree d. One way to see this is that there exist triangulations of S'

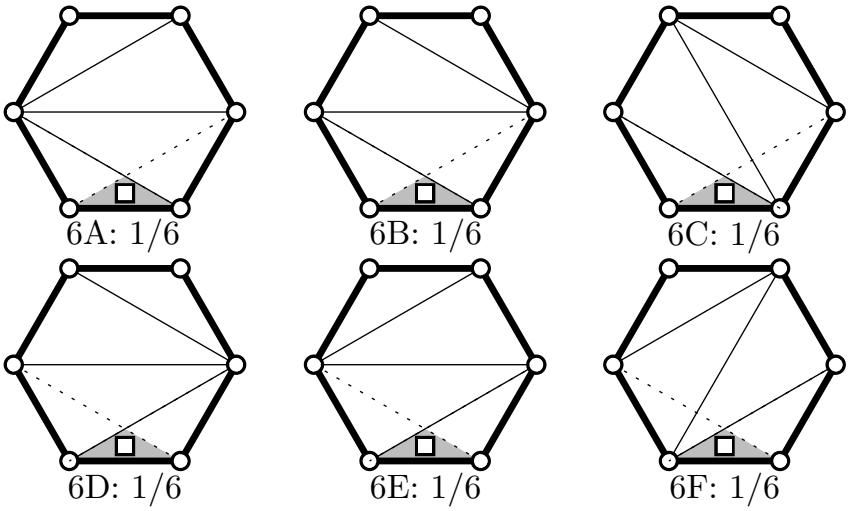


Figure 6 Convex or non-convex 6-star weight distributing.

where p has degree 3 and $|S|$, respectively, that the flip-graph of triangulations is connected, and that an edge flip does not change the degree of any vertex by more than one.

2.3 Combining the results

We now show that any certificate listed in Section 2.1 collects weight at least two. Note that we always get weight 1 from the triangle ($d = 3$). So we only have to argue that in addition we get at least weight 1:

NC4. We get weight 1 from the non-convex 4-gon ($d = 4$).
C5a, NC5a. We get two times weight $\frac{1}{2}$ for each convex 4-gon ($d = 4$).

In the following, we also always get weight $\frac{1}{2}$ for the convex 4-gon ($d = 4$) from the first unique flip. Hence it remains to reason that in addition we get at least weight $\frac{1}{2}$.

NC5b, NC5c, NC5d, NC5e. We get weight $\frac{1}{2}$ from cases 5A, 5B, 5C, and 5B, respectively ($d = 5$).

For the remaining certificates, p always lies in a convex 5-gon in the shaded ear-area. If p lies in the "central wedge" then we get weight $\frac{1}{2}$ from at least one of the two cases 5D or 5E ($d = 5$). Otherwise we know that p lies near an edge, where for all remaining certificates there are two possibilities: p either lies in the wedge close to the bottom edge (as indicated in the figures of the certificates), or in the wedge near the lower left edge. In the following cases we will always consider both possibilities in the just mentioned order (separated by 'or' and, if necessary, grouped by squared brackets), combining cases for $d = 5$ and $d = 6$.

C5b. Weight $\frac{1}{2}$ from case 5J or 5G.
C5c. Weight $\frac{1}{2}$ from case 5I or 5K.
C6a, NC6a. Weight $\frac{1}{2}$ from case 5H (use only the top flip) or 5F (use only the right flip).
CNC6a. [Weight $\frac{1}{3}$ from case 5L plus weight $\frac{1}{6}$ from case 6A] or weight $\frac{1}{2}$ from case 5G.
CNC6b. [Weight $\frac{1}{3}$ from case 5O plus weight $\frac{1}{6}$ from case 6B] or [weight $\frac{1}{3}$ from case 5N plus weight $\frac{1}{6}$ from case 6F].

CNC6c. [Weight $\frac{1}{3}$ from case 5Q plus weight $\frac{1}{6}$ from case 6C] or [weight $\frac{1}{3}$ from case 5M plus weight $\frac{1}{6}$ from case 6E].
CNC6d. Weight $\frac{1}{2}$ from case 5I or [weight $\frac{1}{3}$ from case 5P plus weight $\frac{1}{6}$ from case 6D].

As every triangulation of S gets assigned weight at least two, we get Lemma 5, which together with Lemma 4 implies Lemma 6.

▶ **Lemma 5.** *For $|S| > 6$ it holds that $2|\mathcal{T}(S)| \leq W$.*

▶ **Lemma 6.** *For $|S| > 6$ it holds that $2|\mathcal{T}(S)| < |\mathcal{T}'(S')|$.*

Note that for all triangulations in the set $\mathcal{T}'(S')$ we constructed the point p with degree at most 6. From the discussion at the end of Section 2.2 it follows that for $|S| \geq 6$ we can also write $|\mathcal{T}'(S')| \geq 2|\mathcal{T}(S)| + (|S| - 6)$. Hence we obtain Theorem 7, which in combination with the number of triangulations of convex point sets implies Corollary 8.

▶ **Theorem 7.** *Let $t(i,h) := \min_{S,|S|=i+h} \mathrm{tr}(S)$ denote the* minimum *number of triangulations that every set of $n = i + h \geq 7$ points, with i inner points and h extreme points, exhibits. Let $t(n) := \min_{n=i+h} t(i,h)$ denote the* minimum *number of triangulations that* every *set of $n \geq 7$ points exhibits. The following bounds hold for $t(n)$ and $t(i,h)$.*

$$t(n) \geq 2t(n-1) + (n-7) \tag{1}$$
$$t(i,h) \geq 2t(i-1,h) + (n-7) \tag{2}$$

▶ **Corollary 8.** $t(i,h) = \Omega^*(2^i 4^h)$.

3 Small convex hulls

Here we show that abstract order types which minimize the number of triangulations cannot have more than half of their vertices on the boundary of their convex hulls. We use the following result which is part of the characterization of so-called crossing-minimal point sets [16].

▶ **Proposition 9** ([16])**.** *Let P be an abstract order type with at least four extreme points and three consecutive vertices a, b, and c on the convex hull boundary s.t. no two points inside this triangle are in convex position with a and c. Then the following holds:*

1. *There exists an abstract order type Q and a bijection between P and Q s.t. the images of a and c are consecutive vertices on the convex hull of Q and any point triple in P not containing both a and c is oriented as its image in Q (either clockwise or counterclockwise).*
2. *For every crossing-free geometric graph on P, its image on Q is also crossing-free.*

See Figure 7. Intuitively, we need a set Q and a mapping from the triangulations on Q to the triangulations on P that keeps the image crossing-free.

▶ **Lemma 10.** *Let P and Q be defined as in Proposition 9. Then P has strictly more triangulations than Q.*

Proof. We give an injective mapping of the triangulations of Q to the triangulations of P. Recall that a geometric graph is a triangulation of a point set with h extreme points if it is crossing-free and has $3n - 3 - h$ edges. The analogous holds for graphs on abstract order types. Let h be the number of extreme points of P. Let T be an arbitrary triangulation of Q, which has $3n - 3 - (h-1)$ edges. As ac is on the boundary of the convex hull, it is an edge of T. We remove ac from T and draw the corresponding graph T' on P, as indicated

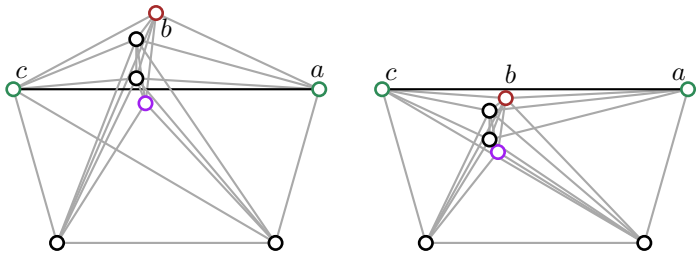


Figure 7 Illustration for Proposition 9. All point triples in the two point sets are the same except that all points are to the left of ac in the right set.

by the bijection between P and Q. As T' has $3n-3-h$ edges, it remains to show that T' is crossing-free.

Suppose that T' on P contains a crossing. Then, the only crossings may occur between two edges containing both a and c. However, T' does not contain the edge ac. Hence, the only crossings could be between an edge as and an edge ct for some points s and t, and at least one of s and t is inside the triangle abc in P (as otherwise the orientation of all point triples in $\{a, c, s, t\}$ would be the same as in Q, implying that there is no such crossing). However, they cannot be both inside the triangle abc due to the characterization of P (as $\{a, c, s, t\}$ would be in convex position). But this means that as and ct cannot cross. Hence, G' is crossing-free.

Since there exists a triangulation of P that contains the edge ac, there are strictly more triangulations of P than of Q. ◀

Suppose there is an abstract order type with four or more extreme points that minimizes the number of triangulations. Consider the triangles that are formed by all triples of points that are consecutive on the convex hull boundary. By Lemma 10, each such triangle must contain (at least) two points. Any point can only be in at most two such triangles. So for h points on the convex hull boundary, we need at least h interior points to have two points inside each such triangle.[1]

▶ **Theorem 11.** *Let P be an abstract order type that minimizes the number of triangulations with $|P| \geq 6$. Then P has at most $\lfloor |P|/2 \rfloor$ extreme points.*

The double circle is an example with $n/2$ extreme points where the local improvement due to Lemma 10 is indeed no longer possible.

4 Tight lower bounds for up to 15 points

Exact values for the minimum numbers of triangulations have been known for sets of at most 11 points [3]. To compute those values also for sets with $12 \leq n \leq 15$ points, we start with the exhaustive database of all order types of cardinality 8 and extend all sets in the abstract setting (that is, without realizing them as point sets). All arguments used in Sections 2 and 3 hold also in the abstract setting, since we used only arguments which use sidedness properties of the underlying (abstract) order type (and no metric properties). By Theorem 11, and

[1] Lemma 10 actually tells us that an abstract order type minimizing the number of triangulations is "crossing-minimal" (in terms of [16]). Due to space constraints, we omit a definition of that term, but state that it has already been noted in [16] that there cannot be large crossing-minimal abstract order types with more than half of the points being extreme.

Table 1 Abstract extension from $n = 8$ to $n = 15$.

n	upper bound nr. of triangulations	number of generated abstract order types	all realizable order types	thereof respecting the triangul. bound
8	534	–	3 315	3 315
9	1 069	158 829	158 817	158 817
10	2 139	13 645 342	14 309 547	13 635 816
11	4 278	437 639 146	2 334 512 907	436 539 440
12	8 556	570 219 167	unknown	unknown
13	17 112	32 884 369	unknown	unknown
14	34 224	101 218	unknown	unknown
15	68 448	1	unknown	1

because $\lfloor 15/2 \rfloor < 8$, it is sufficient to extend sets by adding interior points only. Moreover, choosing the right insertion order, by Lemma 6 we can further reduce the number of order types that have to be extended: For $n = 12, \ldots, 15$, the (in general conjectured triangulation minimizing) double circles have $2236, 7147, 20979, 68448$ triangulations, respectively. To show that, for example, for $n = 13$ the minimal number of triangulations is in fact 7147, it is sufficient to extend all (abstract) order types for $n = 12$ with at most $\lfloor 7147/2 \rfloor = 3573$ triangulations. To do so, we need all order types for $n = 11$ with at most $\lfloor 3573/2 \rfloor = 1786$ triangulations, and so on. Table 1 shows the maximum number of triangulations to be considered for each cardinality from 8 to 15. Note that starting at 8 is necessary as for $n \geq 9$ there exist non-realizable abstract order types and the exhaustive list of order types is limited to the realizable ones. For $n = 9$ there exist 158 830 abstract order types, thus all of them except the one with 9 points in convex position were generated.

We want to extend only those order types that have *at most* a certain number of triangulations. To do so we need a counting algorithm that is run for *each* candidate order type to accept or reject order types for further extensions. Previous sets of experiments comparing different counting algorithms [6, 7] yield three potential candidates described in [6, 7, 17]. However [7] is harder to translate to an abstract setting. After implementing and testing the other two, we decided to use the algorithm by Ray and Seidel [17].

To "translate a counting algorithm to the abstract setting" we use only one primitive, namely `orientation`. That is, given any three points, we answer in $O(1)$ time whether the triple turns left (1) or right (-1). Recall that we do not have collinear triples. Our input is the Λ-matrix [14] of the input order type, that is, a 3-dimensional table where the entry $\Lambda[i][j][k]$ contains either 1 or -1 for $i \neq j \neq k$. Using Λ we can pre-compute for *every* directed edge $\overrightarrow{e} = \overrightarrow{ab} \neq \overleftarrow{ba} = \overleftarrow{e}$ of the input order type S, the set of *empty* triangles $\triangle_{abc}$ such that $\Lambda[a][b][c] = 1$. That is, *every* such vertex $c \in \triangle_{abc} \subset S$ lies to the left of the oriented line supporting $\overrightarrow{e} = \overrightarrow{ab}$. The set of points c is saved in a data structure Δ so that we can retrieve it in $O(1)$ time.

The algorithm by Ray and Seidel is a divide-and-conquer algorithm that uses memoization for efficiency. The algorithm considers a *closed* polygon whose vertices are points of the input order type S, and which might contain other points of S. We call such a polygon, with its contained points, a *pointgon* P. The task is to count all different triangulations of P. The algorithm chooses an edge ab of the boundary of P. Afterwards, all empty triangles $\triangle_{abc}$ fully contained in P are enumerated. Observe that (1) the set of triangles $\triangle_{abc}$ is pairwise-crossing, (2) no two such triangles can appear in the same triangulation of P, and (3) every triangulation of P contains exactly one of those triangles. Hence the number of triangulations $\mathrm{tr}(P)$ of P can then be expressed as $\mathrm{tr}(P) = \sum_{\triangle_{abc}} \mathrm{tr}(P \mid \triangle_{abc})$, where $\mathrm{tr}(P \mid \triangle_{abc})$ is the number of triangulations of P containing the triangle $\triangle_{abc}$.

Table 2 Minimum number of triangulations for n points with h extreme points. Overall *tight* lower bounds for $t(n)$ for fixed $3 \leq n \leq 15$ are selected in gray.

$n\backslash h$	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	**1**	–	-	–	-	–	-	–	-	–	-	–	-	–
4	**1**	2	–	-	–	-	–	-	–	-	–	-	–	–
5	**2**	3	5	–	-	–	-	–	-	–	-	–	-	–
6	**4**	6	9	14	–	-	–	-	–	-	–	-	–	–
7	**11**	13	19	28	42	–	-	–	-	–	-	–	-	–
8	32	**30**	43	62	90	132	–	-	–	-	–	-	–	–
9	96	**89**	102	145	207	297	429	–	-	–	-	–	-	–
10	305	272	**250**	352	497	704	1001	1430	–	-	–	-	–	–
11	991	849	**776**	878	1230	1727	2431	3432	4862	–	-	–	-	–
12	3297	2694	2453	**2236**	3114	4344	6071	8502	11934	16796	–	-	–	–
13	11204	9022	7865	**7147**	8025	11139	15483	21554	30056	41990	58786	–	–	–
14	?	?	?	23128	**20979**	29004	40143	55626	77180	107236	149226	208012	–	–
15	?	?	?	?	**68448**	76473	105477	145620	201246	278426	385662	534888	742900	–
16	?	?	?	?	?	203748	280221	385698	531318	732564	1010990	1396652	1931540	2674440

If for the empty triangle $\triangle_{abc}$, the point c is a vertex of P, then the algorithm divides P into two sub-pointgons P_{ac}, P_{cb} and proceeds recursively on each one. For such a splitting triangle $\triangle_{abc}$, the number of triangulations is $\mathrm{tr}(P \mid \triangle_{abc}) = \mathrm{tr}(P_{ac}) \cdot \mathrm{tr}(P_{cb})$. If c is fully contained inside P, the algorithm continues counting with the pointgon $P_c = P \setminus \triangle_{abc}$ (P_c is a pointgon where the edge ab is replaced by the two edges ac, cb). Thus $\mathrm{tr}(P \mid \triangle_{abc}) = \mathrm{tr}(P_c)$. Observe that in both cases the resulting sub-polygon(s) are smaller than P (less triangles in any of their triangulations than in P) and hence the algorithm eventually terminates. To compute the number of triangulations of S, the convex hull of S (with its interior points) is used as initial pointgon P.

5 A huge induction base

From the abstract extension described in the last section we obtain *exact values* of $t(i, h)$ for $3 \leq i + h \leq 16$ as shown in Table 2.

The entries of Table 2 marked with a question mark are entries that do currently not seem to be computable within a reasonable amount of time (approx. six to twelve months of computation using a cluster of several CPUs). To extend Table 2 with lower bounds (not with *exact* values) for $t(i, h)$, for larger values of $n = i + h$, we use the inequalities provided by Theorem 7 along with two other inequalities from previous work of Sharir, Sheffer, and Welzl [22], and Aichholzer, Hurtado, and Noy [3], respectively.

Sharir, Sheffer, and Welzl [22] derived lower bounds for the number of triangulations by considering the expected number of vertices with degree three in a triangulation. Let $\hat{v}_3$ be the *expected* number of inner vertices with degree three for a triangulation of a set S of $n = i + h$ points, with i the number of inner points and h the number of extreme points.

Let $f(i, h)$ be an upper bound for $\hat{v}_3$. Then there exists an inner vertex $v \in S$ for which it holds that, in a (random) triangulation, $P(\deg(v) = 3) \leq f(i, h)/i$ (the probability that v has degree 3). From that we get $\mathrm{tr}(S \setminus \{v\}) = \mathrm{tr}(S) \cdot P(\deg(v) = 3)$, as any triangulation of $S \setminus \{v\}$ corresponds to a triangulation of S where v has degree 3 (just remove v and its three incident edges). This implies $\mathrm{tr}(S) \geq \frac{i}{f(i,h)} \mathrm{tr}(S \setminus \{v\})$, which in turn implies:

$$t(i, h) \geq \frac{i}{f(i, h)} t(i - 1, h) \tag{3}$$

The following two upper bounds for $\hat{v}_3$ where shown by Sharir et al. [22]: (1) $\hat{v}_3 \leq \frac{i}{2}$ for $n \geq 7$ (Lemma 2.2 in [22]) and (2) $\hat{v}_3 \leq \frac{2i+h/2}{5}$ for $n \geq 6$ (Lemma 2.3 in [22]). Combining

them with (3) we obtain:

$$t(i,h) \geq 2t(i-1,h) \qquad \text{for } n \geq 7 \tag{4}$$

$$t(i,h) \geq \frac{5i}{2i+h/2}t(i-1,h) \qquad \text{for } n \geq 6 \tag{5}$$

Observe that our lower bound (2) from Theorem 7 is better than bound (4) of Sharir et al. by the rather small additive factor of $n-7$. Our bound (2) is, however, considerably stronger in the sense that we know it holds for an *interior-extremal* point while bound (4) holds for *some* inner point of S, not necessarily interior-extremal. This additional property of our bound made it feasible to extend our computations to 15 points.

Aichholzer, Hurtado, and Noy [3] proved the following lower bound for $t(i,h)$:

$$t(i,h) \geq \left\lceil \left(h \cdot \sum\nolimits_{\text{EXT}} + i \cdot \sum\nolimits_{\text{INT}}\right)/(6n-4h-6) \right\rceil \qquad \text{for } n = i+h \geq 4, \tag{6}$$

where $\sum_{\text{INT}}$ is bounded by

$$\sum\nolimits_{\text{INT}} \geq t\left(\frac{n}{2}+1\right)^2 + 2 \cdot \sum_{k=1}^{\frac{n}{2}-1} \min_j \left\{ t\left(\frac{n}{2}+1+j\right) \cdot t\left(\frac{n}{2}+1-j\right) \mid 0 \leq j \leq \min\left\{k, \frac{n}{2}-2\right\}\right\}$$

for $n \geq 4$ even[2], and by

$$\sum\nolimits_{\text{INT}} \geq 2 \cdot \sum_{k=0}^{\frac{n-3}{2}} \min_j \left\{ t\left(\frac{n+3}{2}+j\right) \cdot t\left(\frac{n+1}{2}-j\right) \mid 0 \leq j \leq \min\left\{k, \frac{n-5}{2}\right\}\right\}$$

for $n \geq 5$ odd. For all $n \geq 4$, $\sum_{\text{EXT}}$ is bounded by $\sum_{\text{EXT}} \geq \sum_{k=3}^{n-1} t(k) \cdot t(n+2-k)$.

Observe that (6) makes use of the general minimum number of triangulations $t(n)$. To see that this gives indeed a recursive inequality observe that $\sum_{\text{EXT}}$ and $\sum_{\text{INT}}$ denote expressions that involve values of $t(k)$ only for $k < n$.

Combining the bounds. We want to extend Table 2 with lower bounds for $t(i,h)$ for higher values of $n = i+h$. For this we consider lower bounds (2), (5), and (6), and for each new entry $t(i,h)$ of Table 2 we keep the largest value found among the three inequalities. For the missing entries of Table 2 with $n \leq 15$ we additionally test the largest bound found against the general tight lower bound found at $t(n)$ and we keep the largest of the two. As a reference, the missing entries (marked with a question mark) of Table 2 are filled as in Table 3. Observe, for example, that for $n = 15$ entries for $h \leq 6$ match that of entry for $h = 7$. This is because the general tight lower bound for $n = 15$ is larger than the bounds we obtain using the three aforementioned inequalities. For $n \geq 16$, a lower bound on $t(n)$ can be set after the corresponding entries $t(i,h)$ of Table 2 have been computed. More precisely (and by definition of $t(n)$): $t(n) = \min_{n=i+h} t(i,h)$.

[2] Note that in [3] there was a typo in that formula, as the sum was wrongly taken up to $n/2+1$. However, in the text right after the formula this was stated correctly and it was also used correctly for the computations given in [3].

Table 3 Completing missing entries of Table 2. Overall lower bounds for fixed $3 \leq n \leq 16$ are selected in gray. Entries for $h \geq 9$ are omitted for readability.

$n\backslash h$	3	4	5	6	7	8
3	**1**	–	–	–	–	–
4	**1**	2	–	–	–	–
5	**2**	3	5	–	–	–
6	**4**	6	9	14	–	–
7	**11**	13	19	28	42	–
8	32	**30**	43	62	90	132
9	96	**89**	102	145	207	297
10	305	272	**250**	352	497	704
11	991	849	**776**	878	1230	1727
12	3297	2694	2453	**2236**	3114	4344
13	11204	9022	7865	**7147**	8025	11139
14	≥26223	≥20979	≥20979	23128	**20979**	29004
15	≥68448	≥68448	≥68448	≥68448	**68448**	76473
16	≥161787	≥157957	≥153659	≥148800	**≥143264**	203748

6 Improved Asymptotics

Aichholzer et al. [3] proved a structural theorem which provides the mechanism to obtain asymptotic bounds from a large induction base such as the one we provide in Table 2. For simplicity and completeness we summarize their results (mainly Theorem 2 and Corollary 1 of [3]) in the following theorem.

▶ **Theorem 12** ([3]). *Let $a \geq 3$ be an integer. The following four* independent *relations hold:*

1. *If $t(n) \geq \tau^{n-2}$ for $a \leq n \leq 2a-2$, then $t(n) \geq \tau^{n-2}$ holds for all $n \geq a$.*
2. *If $t(n) \geq \frac{1}{2}\tau^{n-2}$ for $a \leq n \leq 2a+2$, then $t(n) \geq \frac{1}{2}\tau^{n-2}$ holds for all $n \geq a$.*
3. *If $t(n) \geq \frac{1}{3}\tau^{n-2}$ for $a \leq n \leq 2a^2-11a+21$, then $t(n) \geq \frac{1}{3}\tau^{n-2}$ holds for all $n \geq a$.*
4. *If $t(n) \geq \frac{1}{4}\tau^{n-2}$ for $a \leq n \leq 2a^2-13a+29$, then $t(n) \geq \frac{1}{4}\tau^{n-2}$ holds for all $n \geq a$.*

To use Theorem 12 we need to ensure a lower bound $t^-(n,k) = \frac{1}{k} \cdot \tau^{n-2}$, for $1 \leq k \leq 4$, within a given interval $a \leq n \leq b(a,k)$. For $k = 3, 4$ the upper end $b(a,k)$ of the interval is quadratic in a, resulting in very large intervals for which $t(n) \geq t^-(n,k)$ has to be checked. Hence, for proving a bound of $t(n) \geq \frac{1}{k} \cdot \tau^{n-2}$ for some chosen τ and all $n \geq a \geq 3$, we have to verify $t(n) \geq t^-(n,k)$ for all $a \leq n \leq b(a,k)$ for large enough values of n, see Section 5. In other words, consider four closed intervals $[a_k, b_k = b(a_k,k)]$ and define $\tau_{\max} = \max_{1\leq k\leq 4}\{\min_{a_k\leq n\leq b_k}(k \cdot t(n))^{\frac{1}{n-2}}\}$ and let $k_{\max}$ be the $1 \leq k \leq 4$ at which $\tau_{\max}$ is achieved. Theorem 12 implies that $t(n) \geq \frac{1}{k_{\max}} \cdot \tau_{\max}^{n-2} = \Omega(\tau_{max}^n)$ for all $n \geq a_{k_{\max}}$. Table 4 shows lower bounds on $t(n)$ for $16 \leq n \leq 40$ along with values of $\tau = \tau(n,k)$, for $1 \leq k \leq 4$.

For example, using Table 4 and following Theorem 12 we obtain that for $k = 2$ and a base range of $19 \leq n \leq 40$, $\tau(n,2) \geq 2.3778$, and thus $t(n) \geq \Omega(2.3778^n)$ for $n \geq 19$. To obtain a better lower bound for $t(n)$ we extend Table 4 for larger values of n. More precisely, we extend it up to $n \leq 70131$. Due to the lack of space we only mention that the largest bound was achieved for $k = 2$. That is: $\tau(n,2) \geq 2.631035$ for $35064 \leq n \leq 70130$. In the full version of the paper we will provide information about the best bounds found for each of the four values of k. Theorem 12 now implies that $t(n) \geq \Omega(2.631035^n)$ for $n \geq 35064$, which proves Theorem 1.

Computational limits. With our current methods a stronger lower bound on $t(n)$ seems not feasible. The current bound required (exact) computations on very large numbers —

Table 4 Lower bounds on $t(n)$ and $\tau(n,k)$ for $16 \leq n \leq 40$ and $k = 1, \ldots, 4$.

n	$t(n) \geq$	Eq.	$k=1$	$k=2$	$k=3$	$k=4$
16	143264	(5)	2.3350	2.4535	2.5256	2.5781
17	304818	(5)	2.3206	2.4303	2.4969	2.5453
18	657451	(5)	2.3100	2.4122	2.4742	2.5190
19	1434439	(5)	2.3022	2.3981	2.4559	2.4978
20	3160629	(5)	2.2966	2.3868	2.4412	2.4805
21	7023620	(5)	2.2926	2.3778	2.4291	2.4662
22	17826487	(6)	2.3043	2.3856	2.4344	2.4697
23	461173087	(6)	2.3171	2.3948	2.4415	2.4752
24	121862959	(6)	2.3309	2.4055	2.4503	2.4825
25	315689696	(6)	2.3417	2.4133	2.4562	2.4871
26	837664309	(6)	2.3539	2.4229	2.4641	2.4939
27	2205647348	(6)	2.3645	2.4309	2.4707	2.4993
28	5849635236	(6)	2.3749	2.4391	2.4774	2.5050
29	14684224691	(5)	2.3798	2.4417	2.4786	2.5052
30	37399997550	(6)	2.3856	2.4454	2.4810	2.5067
31	95073504069	(6)	2.3908	2.4486	2.4831	2.5079
32	241432822521	(6)	2.3956	2.4516	2.4850	2.5089
33	612064398552	(6)	2.4000	2.4543	2.4866	2.5098
34	1561238258557	(6)	2.4046	2.4572	2.4886	2.5110
35	3973209273535	(6)	2.4087	2.4598	2.4902	2.5120
36	9878502893156	(6)	2.4109	2.4606	2.4901	2.5113
37	24308011357985	(6)	2.4124	2.4606	2.4893	2.5098
38	59495132694369	(5)	2.4133	2.4602	2.4881	2.5081
39	145702365782129	(5)	2.4143	2.4599	2.4870	2.5065
40	357019041982701	(5)	2.4152	2.4597	2.4861	2.5050

for both, the number of triangulations (integer arithmetic) and for $\tau(n,k)$ (floating-point arithmetic). For the former we *eventually* deal with integers that require between 48,000 and 98,000 bits of precision. For the latter we performed *exact* floating-point arithmetic with truncation up to 128 bits of precision. Extending Table 4 up to $n \leq 70131$ and computing the respective $\tau(n,k)$, for $1 \leq k \leq 4$, took about 65 hours on a desktop machine with a somewhat recent processor: Intel i7-4770 CPU at 3.40GHz. For arithmetic on big integers we used the well-known GMP library (for `C`), and for exact floating-point arithmetic we used the also well-known MPFR library (also for `C`). The data produced for this paper can be downloaded from: `http://www.victoralvarez.net/papers/aahpsv.tgz` (around 500 MB).

Acknowledgements. We would like to thank Ruy Fabila Monroy, Raimund Seidel, and Emo Welzl for fruitful discussions on the topic of this paper.

References

1 Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002. `doi:10.1023/A:1021231927255`.

2 Oswin Aichholzer, Thomas Hackl, Clemens Huemer, Ferran Hurtado, Hannes Krasser, and Birgit Vogtenhuber. On the number of plane geometric graphs. *Graphs Combin.*, 23:67–84, 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.*, 29(2):135–145, 2004.

4 Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. *Comput. Geom.*, 36(1):2–15, 2007.

5 Selim G. Akl. A lower bound on the maximum number of crossing-free Hamiltonian cycles in a rectilinear drawing of K_n. *Ars Combin.*, 7:7–18, 1979.

6 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.

7 Victor Alvarez and Raimund Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In *Proc. 29th Annual Symposium on Computational Geometry (SoCG 2013)*, pages 1–8. ACM, 2013.

8 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.

9 Jacob E. Goodman. Proof of a conjecture of Burr, Grünbaum, and Sloane. *Discrete Math.*, 32(1):27–35, 1980. doi:10.1016/0012-365X(80)90096-5.

10 Jacob E. Goodman and Richard Pollack. Proof of Grünbaum's conjecture on the stretchability of certain arrangements of pseudolines. *J. Comb. Theory, Ser. A*, 29(3):385–390, 1980. doi:10.1016/0097-3165(80)90038-2.

11 Jacob E. Goodman and Richard Pollack. Multidimensional sorting. *SIAM J. Comput.*, 12(3):484–507, 1983.

12 Jacob E. Goodman and Richard Pollack. Semispaces of configurations, cell complexes of arrangements. *J. Comb. Theory, Ser. A*, 37(3):257–293, 1984.

13 Michael Hoffmann, André Schulz, Micha Sharir, Adam Sheffer, Csaba D. Tóth, and Emo Welzl. *Thirty Essays on Geometric Graph Theory*, chapter Counting Plane Graphs: Flippability and Its Applications, pages 303–325. Springer New York, New York, NY, 2013.

14 Hannes Krasser. *Order Types of Point Sets in the Plane*. PhD thesis, Institute for Theoretical Computer Science, Graz University of Technology, October 2003.

15 Paul McCabe and Raimund Seidel. New lower bounds for the number of straight-edge triangulations of a planar point set. In *Proc. 20th European Workshop on Computational Geometry (EWCG 2004)*, pages 175–176, 2004.

16 Alexander Pilz and Emo Welzl. Order on order types. In *Proc. 31st Int. Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *LIPIcs*, pages 285–299, 2015.

17 Saurabh Ray and Raimund Seidel. A simple and less slow method for counting triangulations and for related problems. In *Proc. 20th European Workshop on Computational Geometry (EWCG 2004)*, pages 177–180, 2004.

18 Andreas Razen, Jack Snoeyink, and Emo Welzl. Number of crossing-free geometric graphs vs. triangulations. *Electr. Notes Discrete Math.*, 31:195–200, 2008.

19 Francisco Santos and Raimund Seidel. A better upper bound on the number of triangulations of a planar point set. *J. Comb. Theory, Ser. A*, 102(1):186–193, 2003.

20 Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electr. J. Comb.*, 18(1), 2011.

21 Micha Sharir and Adam Sheffer. Counting plane graphs: Cross-graph charging schemes. *Combinatorics, Probability & Computing*, 22(6):935–954, 2013.

22 Micha Sharir, Adam Sheffer, and Emo Welzl. On degrees in random triangulations of point sets. *J. Comb. Theory, Ser. A*, 118(7):1979–1999, 2011.

Recognizing Weakly Simple Polygons*

Hugo A. Akitaya[1], Greg Aloupis[1], Jeff Erickson[2], and Csaba D. Tóth[3]

1 **Department of Computer Science, Tufts University, Medford, MA, USA**
2 **Department of Computer Science, University of Illinois, Urbana-Champaign, IL, USA**
3 **Department of Mathematics, California State University Northridge, Los Angeles, CA, USA**

Abstract

We present an $O(n \log n)$-time algorithm that determines whether a given planar n-gon is weakly simple. This improves upon an $O(n^2 \log n)$-time algorithm by Chang, Erickson, and Xu [4]. Weakly simple polygons are required as input for several geometric algorithms. As such, how to recognize simple or weakly simple polygons is a fundamental question.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases weakly simple polygon, crossing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.8

1 Introduction

A polygon is *simple* if it has distinct vertices and interior-disjoint edges that do not pass through vertices. Geometric algorithms are often designed for simple polygons, but many also work for degenerate polygons that do not "self-cross." A polygon with at least three vertices is *weakly simple* if for every $\varepsilon > 0$, the vertices can be perturbed by at most ε to obtain a simple polygon. Such polygons arise naturally in numerous applications, e.g., for modeling planar networks or as the geodesic hull of points within a simple polygon (Fig. 1).

Several definitions have been proposed for weakly simple polygons, each formalizing the intuition that a weakly simple polygon does not cross itself. Some of these definitions were unnecessarily restrictive or incorrect; see [4] for a detailed discussion. Ribó Mor [7] proved that a weakly simple polygon with at least three vertices can be perturbed into a simple polygon continuously while preserving the lengths of its edges, and maintaining that no two edges properly cross. Chang et al. [4] gave an equivalent definition for simple polygons in terms of the Fréchet distance (see Section 2), in which a polygon is perturbed into a simple closed curve. The latter definition is particularly useful for recognizing weakly simple polygons. Apart from perturbing vertices, it allows transforming edges into polylines (by subdividing the edges with Steiner points which may be perturbed). The perturbation of a vertex incurs only local changes, and need not affect the neighborhood of adjacent vertices.

It is easy to decide whether an n-gon is simple in $O(n \log n)$ time by a sweepline algorithm [8]. Chazelle's triangulation algorithm recognizes simple polygons in $O(n)$ time, because it only produces a triangulation if the input is simple [5]. Recognizing weakly simple polygons is more subtle. Cortese et al. [6] achieved this in $O(n^3)$-time. Chang et al. [4] improved this to $O(n^2 \log n)$ in general; and to $O(n \log n)$ for several special cases. They

* This work was partially supported by the NSF grants CCF-1408763, CCF-1422311, and CCF-1423615.

© Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth;
licensed under Creative Commons License CC-BY

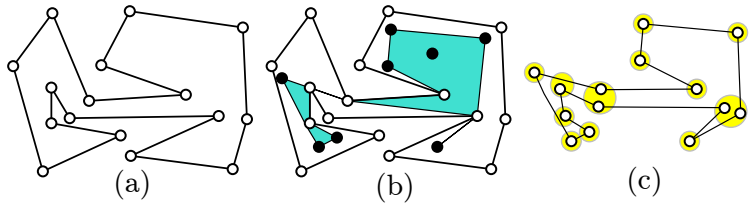


Figure 1 (a) A simple polygon P. (b) Eight points in the interior of P (solid dots); their geodesic hull is a weakly simple polygon P' with 14 vertices. (c) A perturbation of P' into a simple polygon.

identified two features that are difficult to handle: A *spur* is a vertex whose incident edges overlap, and a *fork* is a vertex that lies in the interior of an edge (a vertex may be both a fork and a spur). For polygons with no forks or no spurs, Chang et al. [4] gave an $O(n \log n)$-time algorithm. In the presence of both forks and spurs, their solution is to eliminate forks by subdividing all edges that contain vertices in their interiors, potentially creating a quadratic number of vertices. We show how to manage this situation efficiently, while building on ideas from [4, 6] and from Arkin et al. [2], and obtain the following main result.

▶ **Theorem 1.** *Deciding whether a given n-gon is weakly simple takes $O(n \log n)$ time.*

Our algorithm is detailed in Sections 3–5. It consists of three phases, simplifying the input polygon by a sequence of reduction steps. First, the *preprocessing* phase applies known methods such as *crimp reductions* and *node expansions* (Section 3). Second, the *bar simplification* phase successively eliminates all forks (Section 4). Third, the *spur elimination* phase eliminates all spurs (Section 5). We can also perturb any weakly simple polygon into a simple polygon, in $O(n \log n)$ time, by reversing the sequence of operations.

2 Preliminaries

Here, we review definitions from [4] and [6]. We adopt terminology from [4].

Polygons and weak simplicity. An *arc* in $\mathbb{R}^2$ is a continuous function $\gamma : [0,1] \to \mathbb{R}^2$. A *closed curve* is a continuous function $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$. A closed curve γ is *simple* (also known as a *Jordan curve*) if it is injective. A (*simple*) *polygon* is the image of a piecewise linear (*simple*) closed curve. Thus a polygon P can be represented by a cyclic sequence of points $(p_0, \ldots, p_{n-1})$, called *vertices*, where the image of γ consists of line segments $p_0p_1, \ldots, p_{n-2}p_{n-1}$, and $p_{n-1}p_0$ in this cyclic order. Similarly, a *polygonal chain* (alternatively, *path*) is the image of a piecewise linear arc, and can be represented by a sequence of points $[p_0, \ldots, p_{n-1}]$. A polygon $P = (p_0, \ldots, p_{n-1})$ is *weakly simple* if $n = 2$, or if $n > 2$ and for every $\varepsilon > 0$ there is a simple polygon $(p'_0, \ldots, p'_{n-1})$ such that $|p_ip'_i| < \varepsilon$ for all $i = 0, \ldots, n-1$. This definition is difficult to work with because a small perturbation of a vertex modifies the neighborhoods of the two adjacent vertices. Chang et al. [4] gave an equivalent definition in terms of the Fréchet distance: A polygon given by $\gamma : \mathbb{S}^1 \to \mathbb{R}^2$ is weakly simple if for every $\varepsilon > 0$ there is a simple closed curve $\gamma' : \mathbb{S}^1 \to \mathbb{R}^2$ such that $\mathrm{dist}_F(\gamma, \gamma') < \varepsilon$, where dist_F denotes the Fréchet distance between two closed curves. The curve γ' can approximate an edge of the polygon by a polyline, and any perturbation of a vertex can be restricted to a small neighborhood. With this definition, recognizing weakly simple polygons becomes a combinatorial problem independent of ε, as explained below.

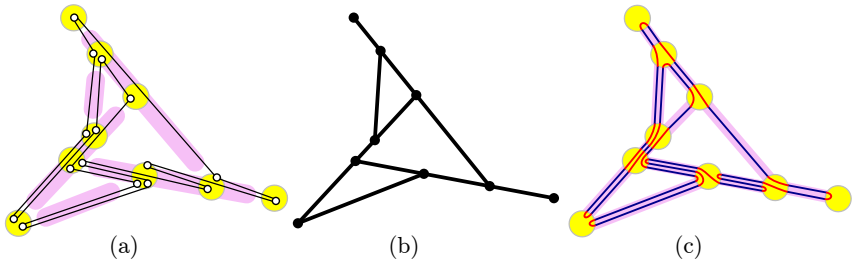


Figure 2 (a) The bar decomposition for a weakly simple polygon P with 16 vertices (P is perturbed into a simple polygon for clarity). (b) Image graph of P. (c) A combinatorial representation of P.

Bar decomposition and image graph. Two edges of a polygon P *cross* if their interiors intersect at precisely one point. The edges of a weakly simple polygon cannot cross. In the following, we assume that such crossings have been ruled out. Two edges of P *overlap* if their intersection is a nondegenerate line segment. The transitive closure of the overlap relation is an equivalence relation on the edges of P; see Fig. 2(a) where equivalence classes are shaded. The union of all edges in an equivalence class is called a *bar*. All bars of a polygon can be computed in $O(n \log n)$ time [4]. The bars are line segments that are pairwise noncrossing and nonoverlapping, and the number of bars is $O(n)$.

The vertices and bars of P define a planar straight-line graph G, called the *image graph* of P. We call the vertices and edges of G *nodes* and *segments* to distinguish them from the vertices and edges of P. Every node that is not in the interior of a bar is called *sober*. The set of nodes in G is $\{p_0, \ldots, p_{n-1}\}$ (note that P may have repeated vertices that correspond to the same node); two nodes are connected by an edge in G if they are consecutive nodes along a bar; see Fig. 2(b). Hence G has $O(n)$ vertices and edges, and it can be computed in $O(n \log n)$ time [4]. Note, however, that up to $O(n)$ edges of P may pass through a node of G, and there may be $O(n^2)$ edge-node pairs such that an edge of P passes through a node of G. An $O(n \log n)$-time algorithm cannot afford to compute these pairs explicitly.

Operations. We use certain elementary operations that modify a polygon and ultimately eliminate forks and spurs. An operation that produces a weakly simple polygon iff it is performed on a weakly simple polygon is called *ws-equivalent*. We shall use some known ws-equivalent operations, and introduce several new ws-equivalent operations in Sections 3.3–5.

Combinatorial characterization of weak simplicity. To show that an operation is ws-equivalent, it suffices to show the existence of ε-perturbations. We will use perfect matchings to combinatorially represent ε-perturbations (independent of ε or any specific embedding). This representation is a variation of the "strip system" introduced in [4].

Let P be a polygon and G its image graph. We construct a family of simple polygons as follows. Let $\varepsilon = \varepsilon(P) \in (0, 1)$, to be specified shortly. For every node u of G, draw a disk D_u of radius ε centered at u. Choose ε sufficiently small so that the disks are pairwise disjoint, and no disk intersects a nonincident segment of G. Let the *corridor* N_{uv} of segment uv be the set of points at distance at most ε^2 from uv, outside of the disks D_u and D_v, that is, $N_{uv} = \{p \in \mathbb{R}^2 : \mathrm{dist}(p, uv) \le \varepsilon^2, p \notin D_u \cup D_v\}$. Reduce ε further, so that all corridors are pairwise disjoint, and also disjoint from any disk D_w, $w \notin \{u, v\}$. For every segment uv of G, let the *volume* $\mathrm{vol}(uv)$ be the number of edges of P between u and v. For every segment uv, draw $\mathrm{vol}(uv)$ parallel line segments between ∂D_u and ∂D_v within N_{uv}. Finally, for every disk D_u, construct a plane straight-line perfect matching between the segment endpoints

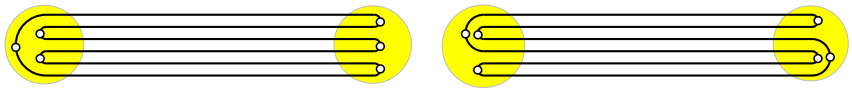


Figure 3 Two perturbations of a weakly simple polygon on 6 vertices (all of them spurs) that alternate between two distinct points in the plane.

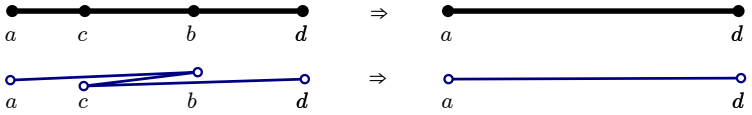


Figure 4 A crimp reduction replaces $[a, b, c, d]$ with $[a, d]$. Top: image graph. Bottom: polygon.

on the boundary ∂D_u (see Fig. 2(c) where the matchings are drawn with circular arcs for clarity). The line segments in the corridors and the perfect matchings in the disks produce a plane 2-regular graph Q. Denote by $\Phi(P)$ the family of all plane graphs constructed in this way such that Q is connected and visits the disks in the same cyclic order as P. By Theorem B.2 in [4], P is weakly simple iff $\Phi(P) \neq \emptyset$. Every $Q \in \Phi(P)$ defines (and is defined by) a linear order on overlapping edges of P.

Note that the above combinatorial representation, which will be used in our proofs, may have $\Theta(n^2)$ size, since each edge passing through a node u contributes one edge to a matching in D_u. We use this simple combinatorial representation in our proofs of correctness, but our algorithm will not maintain it explicitly.

In the absence of spurs, a weakly simple polygon P defines a unique crossing-free perfect matching in each disk D_u [4] which defines a 2-regular graph Q. Consequently, to decide whether P is weakly simple it is enough to check whether $Q \in \Phi(P)$. This is no longer the case in the presence of spurs. In fact, it is not difficult to construct weakly simple n-gons that admit $2^{\Theta(n)}$ combinatorially different perturbations into simple polygons; see Fig. 3.

3 Preprocessing

By a standard line sweep [8], we detect any edge crossing. We then simplify the polygon, using some known steps from [2, 4], and some new. All of this takes $O(n \log n)$ time.

3.1 Crimp reduction

Arkin et al. [2] gave an $O(n)$-time algorithm for recognizing weakly simple n-gons where all edges are collinear. They define the ws-equivalent crimp-reduction operation (see the full paper [1] for details). A *crimp* is a chain of three consecutive edges $[a, b, c, d]$ such that both the first edge $[a, b]$ and the last edge $[c, d]$ contain the middle edge $[b, c]$ (the containment need not be strict). The crimp-reduction replaces the crimp with edge $[a, d]$; see Fig. 4.

Given a chain of two edges $[a, b, c]$ such that $[a, b]$ and $[b, c]$ are collinear but do not overlap, the merge operation replaces $[a, b, c]$ with a single edge $[a, c]$. The merge operation (as well as its inverse, subdivision) is ws-equivalent by the definition of weak simplicity in terms of Fréchet distance [4]. If we greedily apply crimp-reductions and merge operations (cf. Section 2), in linear time we obtain a polygon with the following two properties:

(A1) Every two consecutive collinear edges overlap (i.e., form a spur).

(A2) No three consecutive collinear edges form a crimp.

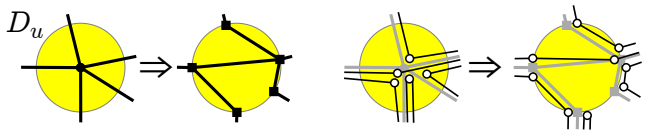


Figure 5 Node expansion. (Left) Changes in the image graph. (Right) Changes in P (the vertices are perturbed for clarity); new nodes are shown as squares.

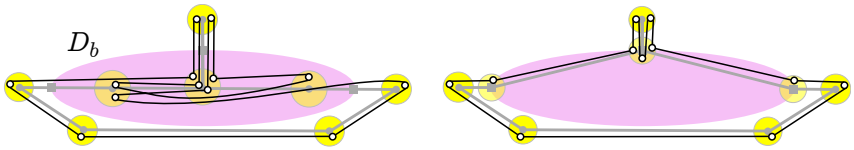


Figure 6 The old-bar-expansion converts a non-weakly simple polygon to a weakly simple one.

▶ **Lemma 2.** *Let $C = [e_i, \ldots, e_k]$ be a chain of collinear edges in a polygon with properties* (A1) *and* (A2)*. Then the sequence of edge lengths $(|e_i|, \ldots, |e_k|)$ is unimodal (all local maxima are consecutive); and no two consecutive edges have the same length, except possibly the maximal edge length that can occur at most twice.*

Proof. For any j such that $i < j < k$, consider $|e_j|$. If $|e_{j-1}|$ and $|e_{j+1}|$ are at least as large, then the three edges form a crimp, by (A1). However, this contradicts (A2). This proves unimodality, and that no three consecutive edges can have the same length. In fact if $|e_j|$ is not maximal, one neighbor must be strictly smaller, to avoid the same contradiction. ◀

3.2 Node expansion

Compute the bar decomposition of P and its image graph G (defined in Section 2, see Fig. 2). For every sober node of the image graph, we perform the ws-equivalent node-expansion operation, described by Chang et al. [4][Section 3] (Cortese et al. [6] call this a *cluster expansion*). Let u be a sober node of the image graph and D_u be the disk centered at u with radius sufficiently small so that D_u intersects only the segments incident to u. For each segment ux incident to u, create a new node u^x at the intersection point $ux \cap \partial D_u$. Then modify P by replacing each subpath $[x, u, y]$ passing through u by $[x, u^x, u^y, y]$; see Fig. 5. If a node expansion produces an edge crossing, report that P is not weakly simple.

3.3 Bar expansion

Chang et al. [4][Section 4] define a bar expansion operation, referred in this paper as old-bar-expansion. For a bar b of the image graph, draw a long and narrow ellipse D_b around the interior nodes of b, and replace each maximal path that intersects with D_b by a straight-line edge. If b contains no spurs, old-bar-expansion is known to be ws-equivalent [4]. Otherwise, it can produce false positives, hence it is not ws-equivalent; see Fig. 6 for an example.

New bar expansion operation. Let b be a bar in the image graph with at least one interior node; see Fig. 7. Let D_b be an ellipse whose major axis is in b such that D_b contains all interior nodes of b (all nodes in b except its endpoints), but does not contain any other node of the image graph and does not intersect any segment that is not incident to some node inside D_b. Similar to old-bar-expansion, the operation new-bar-expansion introduces subdivision vertices on ∂D_b, but all interior vertices of b remain at their original positions.

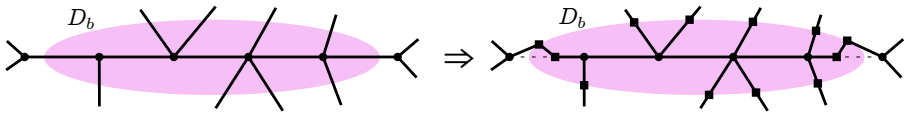


Figure 7 The changes in the image graph caused by new-bar-expansion.

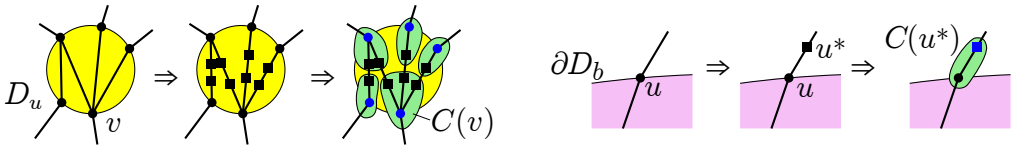


Figure 8 Formation of new clusters around (left) a sober node and (right) a node on the boundary of an elliptical disk. The roots of the induced trees are colored blue.

For each segment ux between a node $u \in b \cap D_b$ and a node $x \notin b$, create a new node u^x at the intersection point $ux \cap \partial D_b$ and subdivide every edge $[u, x]$ to a path $[u, u^x, x]$. For each endpoint v of b, create two new nodes, v' and v'', as follows. Node v is adjacent to a unique segment $vw \subset b$, where $w \in b \cap D_b$. Create a new node $v' \in \partial D_b$ sufficiently close to the intersection point $vw \cap \partial D_b$, but strictly above b; and create a new node v'' in the interior of segment $vw \cap D_b$. Subdivide every edge $[v, y]$, where $y \in b$, into a path $[v, v', v'', y]$. Since the new-bar-expansion operation consists of only subdivisions (and slight perturbations of the edges passing through the end-segments of the bars), it is ws-equivalent.

Terminology. Here, we classify each path in D_b. All nodes $u \in \partial D_b$ lie either above or below b. We call them *top* and *bottom* nodes, respectively. Let $\mathcal{P}$ denote the set of maximal paths $p = [u_1^x, u_1, \ldots, u_k, u_k^y]$ in D_b. The paths in $\mathcal{P}$ can be classified based on the position of their endpoints. A path p is called a

- *cross chain* if u_1^x and u_k^y are top and bottom nodes respectively;
- *top chain* (resp., *bottom chain*) if both u_1^x and u_k^y are top nodes (resp., bottom nodes);
- *pin* if $p = [u_1^x, u_1, u_1^x]$ (note that every pin is a top or a bottom chain);
- *V-chain* if $p = [u_1^x, u_1, u_1^y]$, where $x \neq y$ and p is a top or a bottom chain.

Let $\mathcal{P}in \subset \mathcal{P}$ be the set of pins, and $\mathcal{V} \subset \mathcal{P}$ the set of V-chains. Let M_{cr} be the set of longest edges of *cross chains* in $\mathcal{P}$ (by Lemma 2, each cross chain contributes one or two edges). Every weakly simple polygon has the following property.

(A3) No edge in M_{cr} lies in the interior of any other edge of P.

We can test property (A3) in $O(n \log n)$ time at preprocessing (for each bar, sort all edges by their endpoints, and compute M_{cr}). If property (A3) fails, we report that P is not weakly simple. The operations introduced in Section 2 maintain properties (A1)–(A3) in bars.

3.4 Clusters

As a preprocessing for spur elimination (Section 5), we group all nodes that do not lie inside a bar into *clusters*. After node-expansion and new-bar-expansion, all such nodes lie on a boundary of a disk (circular or elliptical). For every sober node u, we create $\deg(u)$ clusters as follows. Refer to Fig. 8. The node expansion has replaced u with new nodes on ∂D_u. Subdivide each segment in D_u with two new nodes. For each node $v \in \partial D_u$, form a cluster $C(v)$ that consists of v and all adjacent (subdivision) nodes inside D_u. For each node u on the boundary of an elliptical disk D_b, subdivide the unique edge outside D_b incident to u with a node u^*. Form a cluster $C(u^*)$ containing u and u^*.

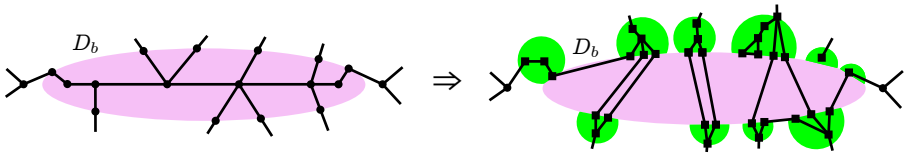


Figure 9 The changes in the image graph caused by a bar simplification.

Cluster Invariants. For every cluster $C(u)$:

(I1) $C(u)$ induces a tree $T[u]$ in the image graph rooted at u.

(I2) Every maximal path of P in $C(u)$ is of one of the following two types:

- **(a)** both endpoints are at the root of $T[u]$ and the path contains a single spur;
- **(b)** one endpoint is at the root, the other is at a leaf, and the path contains no spurs.

Additionally, each leaf node ℓ satisfies the following:

(I3) ℓ has degree one or two in the image graph of P;

(I4) there is no spur at ℓ;

(I5) no edge passes through ℓ (i.e., there is no edge $[a, b]$ such that $\ell \in ab$ but $\ell \notin \{a, b\}$).

Initially, every cluster trivially satisfies (I1) and (I2.b) and every leaf node satisfies (I3)–(I5) since it was created by a subdivision. The operations in Section 4 maintain these invariants.

Dummy vertices. Although the operations described in Sections 4 and 5 introduce nodes in clusters, the image graph will always have $O(n)$ nodes and segments. A vertex at a cluster node is called a *benchmark* if it is a spur or if it is at a leaf node; otherwise it is called a *dummy vertex*. Paths traversing clusters may contain $\Theta(n^2)$ dummy vertices in the worst case, however we do not store these explicitly. By (I1), (I2) and (I4) a maximal path in a cluster can be uniquely encoded by one benchmark vertex: if it goes from a root to a spur at an interior node s and back, we record only $[s]$; and if it traverses $T[u]$ from the root to a leaf ℓ, we record only $[\ell]$.

4 Bar simplification

In this section we introduce three new ws-equivalent operations and show that they can eliminate all vertices from each bar independently (thus eliminating all forks). The bar decomposition is pre-computed, and the bars remain fixed during this phase (even though all edges along each bar are eliminated). We give an overview of the effect of the operations (Section 4.1), define them and show that they are ws-equivalent (Sections 4.2 and 4.3), and then show how to use these operations to eliminate all vertices from a bar (Section 4.4).

4.1 Overview

After preprocessing in Section 3, we may assume that P has no edge crossings and satisfies (A1)–(A3). We summarize the overall effect of the bar simplification subroutine for a given expanded bar.

Changes in the image graph *G*. Refer to Fig. 9. All nodes in the interior of the ellipse D_b are eliminated. Some spurs on b are moved to new nodes in the clusters along ∂D_b. Segments inside D_b connect two leaves of trees induced by clusters.

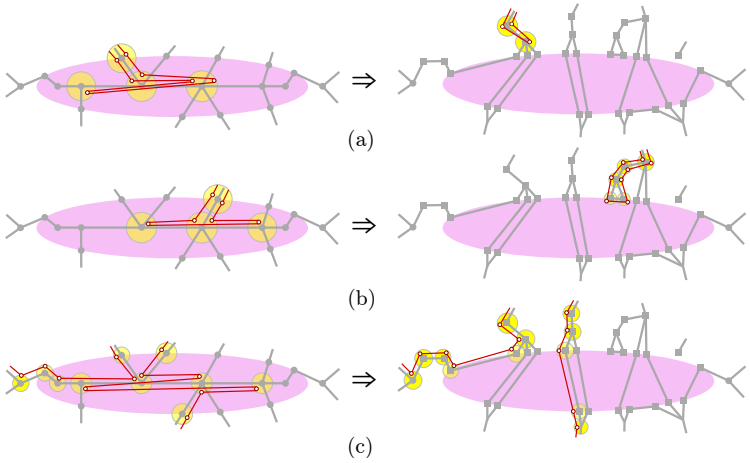


Figure 10 The changes in the polygon caused by a bar simplification.

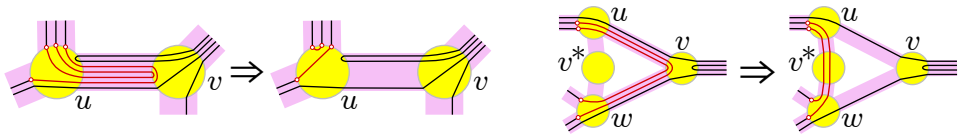


Figure 11 Left: Spur-reduction(u, v). Right: Node-split(u, v, w).

Changes in the polygon *P*. Refer to Fig. 10. Consider a maximal path p in P that lies in D_b. The bar simplification will replace $p = [u, \ldots, v]$ with a new path p'. By (I3)–(I4), only nodes u and v in p lie on ∂D_b. If p is the concatenation of p_1 and $p_2 (= p_1^{-1})$, then p' will be a spur in the cluster containing u (Fig. 10(a)). If p has no such decomposition, but its two endpoints are at the same node, $u = v$, then p' will be a single edge connecting two leaves in the cluster containing u (Fig. 10(b)). If the endpoints of p are at two different nodes, p' is an edge between two leaves of the clusters containing u and v respectively (Fig. 10(c)).

4.2 Primitives

The operations in Section 4.3 rely on two basic steps, *spur-reductions* and *node splits* (see Fig. 11). The proof of their ws-equivalence is available in the full paper [1]. Together with merge and subdivision, these operations are called *primitives*.

- spur-reduction(u, v). Assume that every vertex at node u has at least one incident edge $[u, v]$. Replace any path $[u, v, u]$, with a single-vertex path $[u]$.
- node-split(u, v, w). Assume segments uv and vw are consecutive in radial order around v, and not collinear with an adjacent segment; and P contains no spurs of the form $[u, v, u]$ or $[w, v, w]$. Create node v^* in the interior of the wedge $\angle(u, v, w)$ sufficiently close to v; and replace every path $[u, v, w]$ with $[u, v^*, w]$.

4.3 Operations

We describe three operations: pin-extraction, V-shortcut, and L-shortcut. The first two eliminate pins and V-chains, respectively, and the third simplifies chains in $\mathcal{P}$ with two or more vertices in the interior of D_b, removing one vertex at a time.

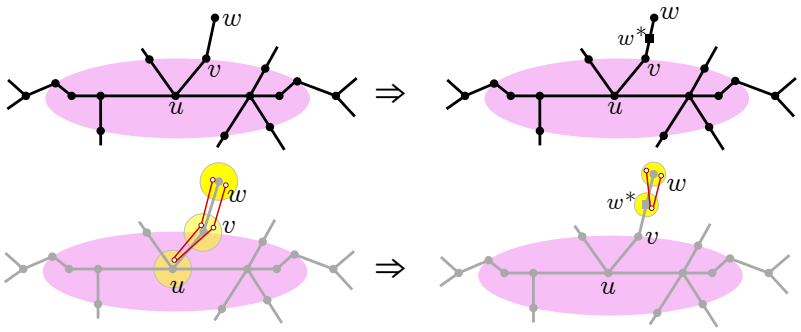


Figure 12 pin-extraction. Changes in the image graph (top), changes in the polygon (bottom).

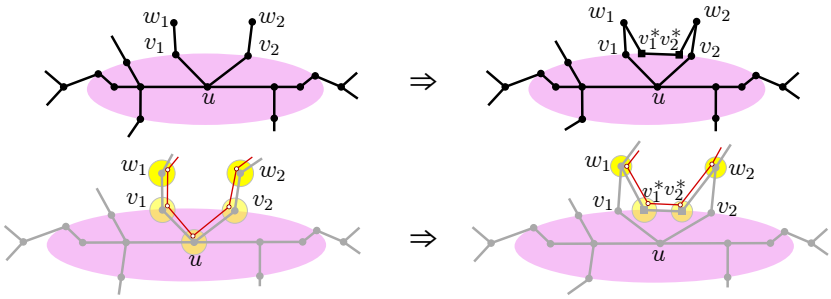


Figure 13 V-shortcut. Changes in the image graph (top), changes in the polygon (bottom).

Pin-extraction and V-shortcut operations. These operations are combinations of primitives and, therefore, they are ws-equivalent. (I1)–(I5) are maintained by construction, and (A1)–(A3) are also maintained within each bar. Proofs are available in the full paper [1].

- pin-extraction(u, v). Assume that P satisfies (I1)–(I5) and contains a pin $[v, u, v] \in \mathcal{P}in$. By (I3), node v is adjacent to a unique node w outside of D_b. Perform the following three primitives: (1) subdivision of every path $[v, w]$ into $[v, w^*, w]$; (2) spur-reduction(v, u). (3) spur-reduction(w^*, v). See Fig. 12 for an example.
- V-shortcut(v_1, u, v_2). Assume that P satisfies (I1)–(I5) and $[v_1, u, v_2] \in \mathcal{V}$. Furthermore, P contains no pin of the form $[v_1, u, v_1]$ or $[v_2, u, v_2]$, and no edge $[u, q]$ such that segment uq is in the interior of the wedge $\angle(v_1, u, v_2)$.
  By (I3), nodes v_1 and v_2 are each adjacent to unique nodes w_1 and w_2 outside of D_b, respectively. The operation executes the following primitives sequentially: (1) node-split(v_1, u, v_2), which creates u^*; (2) node-split(u^*, v_1, w_1) and node-split(u^*, v_2, w_2); which create $v_1^*, v_2^* \in \partial D_b$; (3) merge every path $[v_1^*, u^*, v_2^*]$ to $[v_1^*, v_2^*]$. See Fig. 13 for an example.

L-shortcut operation. The purpose of this operation is to eliminate a vertex of a path that has an edge along a given bar. Before describing the operation, we introduce some notation. For a node $v \in \partial D_b$, let L_v be the set of paths $[v, u_1, u_2]$ in P such that $u_1, u_2 \in \text{int}(D_b)$. Each path in $\mathcal{P}$ is either in $\mathcal{P}in$, in $\mathcal{V}$ or has two subpaths in some L_v. Recall that M_{cr} is the set of longest edges of cross chains in $\mathcal{P}$. Denote by $\widehat{L_v} \subset L_v$ the set of paths $[v, u_1, u_2]$, where $[u_1, u_2]$ is *not* in M_{cr}. We partition L_v into four subsets: a path $[v, u_1, u_2] \in L_v$ is in

1. L_v^{TR} (*top-right*) if v is a *top* vertex and $x(u_1) < x(u_2)$;
2. L_v^{TL} (*top-left*) if v is a *top* vertex and $x(u_1) > x(u_2)$;

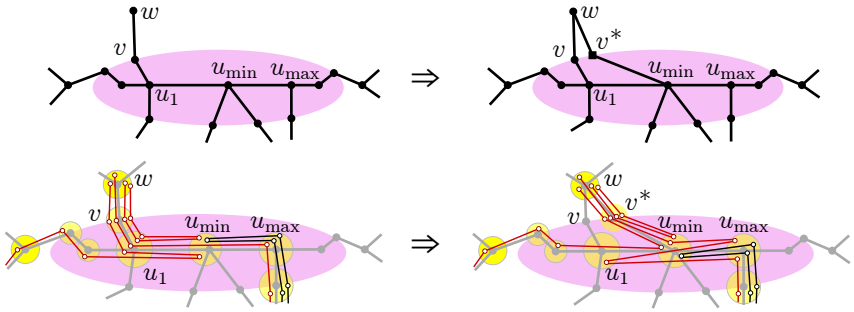


Figure 14 L-shortcut. Changes in the image graph (top), changes in the polygon (bottom).

3. L_v^{BR} (*bottom-right*) if v is a *bottom* vertex and $x(u_1) < x(u_2)$;
4. L_v^{BL} (*bottom-left*) if v is a *bottom* vertex and $x(u_1) < x(u_2)$.

We partition $\widehat{L_v}$ into four subsets analogously. We define the operation L-shortcut for paths in L_v^{TR}; the definition for the other subsets can be obtained by suitable reflections.

- L-shortcut(v, TR). Assume that P satisfies (I1)–(I5), $v \in \partial D_b$ and $L_v^{TR} \neq \emptyset$. By (I3), v is adjacent to a unique node $u_1 \in b$ and to a unique node $w \notin D_b$. Let U denote the set of all nodes u_2 for which $[v, u_1, u_2] \in L_v^{TR}$. Let $u_{\min} \in U$ and $u_{\max} \in U$ be the leftmost and rightmost node in U, respectively. Further assume that P satisfies:
  - **(B1)** no pins of the form $[v, u_1, v]$;
  - **(B2)** no edge $[p, u_1]$ such that segment pu_1 is in the interior of the wedge $\angle(v, u_1, u_2)$;
  - **(B3)** no edge $[p, q]$ such that $p \in \partial D_b$ is a top vertex and $q \in b$, $x(u_1) < x(q) < x(u_{\max})$.

  (See the full paper [1] for an justification of these assumptions.) Do the following.
  - **(0)** Create a new node $v^* \in \partial D_b$ to the right of v sufficiently close to v.
  - **(1)** For every path $[v, u_1, u_2] \in L_v^{TR}$ where u_1u_2 is the *only* longest edge of a cross chain, create a crimp by replacing $[u_1, u_2]$ with $[u_1, u_2, u_1, u_2]$.
  - **(2)** Replace every path $[w, v, u_1, u_{\min}]$ by $[w, v^*, u_{\min}]$.
  - **(3)** Replace every path $[w, v, u_1, u_2]$, where $u_2 \in U$, by $[w, v^*, u_{\min}, u_2]$.

  See Fig. 14.

▶ **Lemma 3.** *L-shortcut is ws-equivalent and maintains* (I1)–(I5).

Proof Sketch (see [1] for a full proof). W.l.o.g., assume L-shortcut(v, TR) is executed. Phase (1) is ws-equivalent by [2]. The rest of the operation is equivalent to subdividing every path in L_v^{TR} where $u_2 \neq u_{\min}$ into $[v, u_1, u_{\min}, u_2]$, node-split$(v, u_1, u_{\min})$ (which creates u_1^*), node-split(w, v, u_1^*) (which creates v^*) and merging every path $[v^*, u_1^*, u_{\min}]$ to $[v^*, u_{\min}]$. Except for node-split$(v, u_1, u_{\min})$, all primitives satisfy their constraints and therefore are ws-equivalent. It remains to show that (B1)–(B3) ensure that this primitive is ws-equivalent. Let P' be obtained from P after node-split$(v, u_1, u_{\min})$. If P' is weakly simple, by changing its embedding we can move u_1^* arbitrarily close to u_1 without affecting weak simplicity, hence P is weakly simple. It remains to show that if P is weakly simple, there exists $Q \in \Phi(P)$ such that the paths in L_v^{TR} are the topmost paths in the linear order induced by Q. Indeed, if there is one edge $[p, q]$ above one path in L_v^{TR}, by (B1)–(B3), it must be part of a path $[p, q, r]$ such that q is a spur and $x(u_{\max}) < x(p)$ and $x(u_{\max}) < x(r)$. Then, it can always be moved below the lowest edge $[u_2, u_3]$ adjacent to a path in L_v^{TR} without introducing any crossing (similar to crimp reduction; see Fig. 15). Phase (1) ensures that this is always possible, since after that phase every path in L_v^{TR} has an adjacent edge

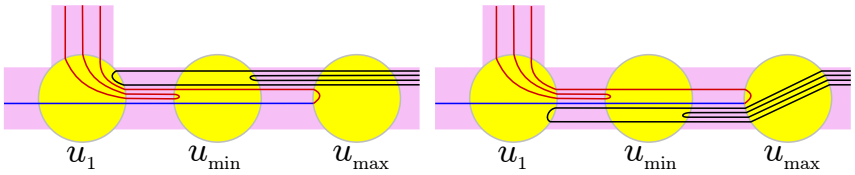


Figure 15 If P_1 is weakly simple, we can change the linear order of the edges as shown.

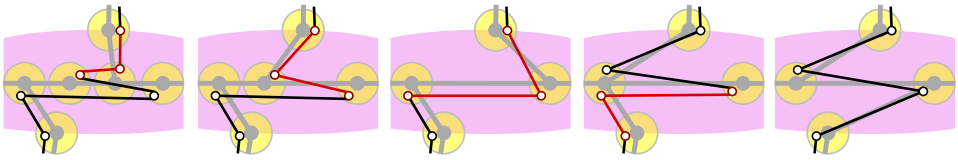


Figure 16 Life cycle of a cross chain in the while loop of bar-simplification. The steps applied, from left to right, are: (4), (3), (4), (6).

on b. Notice that phases (2) and (3) restore (A2) in the bar. Therefore we can "shorten" the lengths of paths in L_v^{TR} to create a simple polygon $Q' \in \Phi(P')$, hence P' is weakly simple. ◀

4.4 Bar simplification algorithm

In this section, we show that the three operations (pin-extraction, V-shortcut, and L-shortcut) can successively remove all spurs of the polygon P from a bar b.

Algorithm bar-simplification(P, b).
While P has an edge along b, perform one operation as follows.

- **(i)** If $\mathcal{P}in \neq \emptyset$, pick an arbitrary pin $[v, u, v]$ and perform pin-extraction(u, v).
- **(ii)** Else if $\mathcal{V} \neq \emptyset$, then let $[v_1, u, v_2] \in \mathcal{V}$ be a path where $|x(v_1) - x(v_2)|$ is minimal, and perform V-shortcut(v_1, u, v_2).
- **(iii)** Else if there exists $v \in \partial D_b$ such that $\widehat{L_v}^{TR} \neq \emptyset$, do:
  - **(a)** Let v be the rightmost node where $L_v^{TR} \neq \emptyset$.
  - **(b)** If $L_{v'}^{TL} = \emptyset$ for all $v' \in \partial D_b$, $x(v) < x(v')$ and $x(u_1') < x(u_{\max})$, where u_1' is the unique neighbor of v' on b, do L-shortcut(v, TR).
  - **(c)** Else let v' be the leftmost node such that $x(v) < x(v')$ and $L_{v'}^{TL} \neq \emptyset$. If $L_{v'}^{TL}$ satisfies (B3) do L-shortcut(v', TL), otherwise halt and report that P is not weakly simple.
- **(iv)** Else if there exists $v \in \partial D_b$ such that $L_v^{TL} \neq \emptyset$, repeat steps (iiia-c) with left–right and TR–TL interchanged. (Note the use of L_v instead of $\widehat{L_v}$. The same applies to (vi)).
- **(v)** Else if there exists $v \in \partial D_b$ such that $\widehat{L_v}^{BL} \neq \emptyset$, repeat steps (iiia-c) using BL and BR in place of TR and TL, respectively, and left-right interchanged.
- **(vi)** Else if there exist $v \in \partial D_b$ such that $L_v^{BR} \neq \emptyset$, repeat steps (iiia-c) using BR and BL in place of TR and TL respectively.

After the loop ends, perform old-bar-expansion (cf. Section 3.3) in the ellipse D_b;
Return P (end of algorithm).

Informally, bar-simplification "unwinds" each polygonal chain in the bar, while extracting pins and V-chains as they appear, by alternating between steps (3) to (6) (see Fig. 16). Step (3) uses $\widehat{L_v}^{TR}$ (instead of ${L_v}^{TR}$) to avoid an infinite loop.

▶ **Lemma 4.** *The operations performed by* bar-simplification(P, b) *are ws-equivalent, and maintain properties* (A1)–(A3) *and* (I1)–(I5) *inside* D_b*. The algorithm either removes all nodes from the ellipse* D_b*, or reports that* P *is not weakly simple. The* L-shortcut *operations performed by the algorithm create at most two crimps in each cross-chain in* $\mathcal{P}$*.*

Proof. We show that the algorithm only uses operations that satisfy their preconditions, and reports that P is not weakly simple only when P contains a forbidden configuration.

Steps (1)–(2). Since every pin can be extracted from a polygon satisfying (I1)–(I5), we may assume that $\mathcal{P}in = \emptyset$. Suppose that $\mathcal{V} \neq \emptyset$. Let $[v_1, u, v_2] \in \mathcal{V}$ be a V-chain such that $|x(v_1) - x(v_2)|$ is minimal. Since $\mathcal{P}in = \emptyset$, the only obstacle for condition (B1) is an edge $[u, q]$ such that segment uq is in the interior of the wedge $\angle(v_1, u, v_2)$ (or else the image graph would have a crossing). This edge is part of a path $[p, u, q]$. Node q must be on ∂D_b between v_1 and v_2, otherwise paths $[p, u, q]$ and $[v_1, u, v_2]$ cross. However, $p \neq q$, otherwise $[p, u, q]$ would be a pin. Consequently, $[p, u, q]$ is a V-chain where $|x(p) - x(q)| < |x(v_1) - x(v_2)|$, contrary to the choice of $[v_1, u, v_2] \in \mathcal{V}$. This confirms that V-shortcut(v_1, u, v_2) satisfies (B1). Henceforth, assume that $\mathcal{P}in = \emptyset$ and $\mathcal{V} = \emptyset$.

Step (3)–(4). By symmetry, we consider only step (3). We distinguish between two cases.

Case 1: the conditions of (2) are satisfied. We need to show that L-shortcut(v, TR) satisfies (B1)–(B3). Since $\mathcal{P}in = \emptyset$, condition (B1) is met. Suppose there is an edge $[p, u_1]$ such that segment pu_1 is in the interior of the wedge $\angle(v, u_1, u_{\min})$. Clearly, $p \in \partial D_b$ is a top node. Then edge $[p, u_1]$ is part of a path $[p, u_1, q]$. However, q must be in the closed wedge $\angle(v, u_1, u_{\min})$ otherwise there would be a node-crossing at u_1. Also, q cannot be a top vertex on ∂D_b since $\mathcal{P}in = \mathcal{V} = \emptyset$, and q cannot be on b by the choice of node v. This confirms (B2). We argue similarly for (B3). Suppose there is an edge $[p, q]$ such that $p \in \partial D_b$ is a top vertex and $q \in b$, $x(u_1) < x(q) < x(u_{\max})$. This edge is part of a path $[p, q, r]$. Node r must be on or above b, otherwise there would be a node-crossing at q. It cannot be a top vertex, since $\mathcal{P}in = \mathcal{V} = \emptyset$. It cannot be to the left of q, otherwise the conditions of (2) are satisfied; and it cannot be to the right of q, otherwise $L_p^{TR} \neq \emptyset$ with $x(v) < x(p)$, contrary to the choice of v. This confirms that L-shortcut(v, TR) satisfies (B2)–(B3) and can be performed.

Case 2: the conditions of (2) are not satisfied. Let the path $[v', u'_1, u'_{\min}] \in L_{v'}^{TL}$ be selected in L-shortcut(v', TL) by the algorithm. Condition (B2) is satisfied similar to Case 1. If (B3) fails, there is an edge $[p, q]$ such that $p \in \partial D_b$ is a top vertex and $q \in b$, $x(u'_{\max}) < x(q) < x(u_{\max})$ (Recall that left and right are interchanged in L^{TL}). Edge $[p, q]$ is part of a path $[p, q, r]$, where $r \in b$, similar to Case 1. This implies $[p, q, r] \in L_p^{TR} \cup L_p^{TL}$. If $x(v) < x(p) < x(v')$, then either $L_p^{TR} \neq \emptyset$, which contradicts the choice of v, or $L_p^{TL} \neq \emptyset$, which contradicts the choice of v'. Consequently, $x(p) \leq x(v)$. This implies $x(u'_{\max}) < x(p) \leq x(v)$, so the paths $[v, u_1, u_{\max}]$ and $[v', u'_1, u'_{\max}]$ cross. Therefore the algorithm correctly finds that P is not weakly simple.

Steps (5)–(6). If steps (1)–(4) do not apply, then $\widehat{L_v}^{TR} \cup L_v^{TL} = \emptyset$. That is, for every path $[v, u_1, u_2] \in L^{TR}$, we have $[u_1, u_2] \in M_{cr}$. In particular, there are no top chains. The operations in (5)–(6) do not change these properties. Consequently, once steps (5)–(6) are executed for the first time, steps (3)–(4) are never executed again. By a symmetric argument steps (5)–(6) eliminate all paths in $\widehat{L_v}^{BL} \cup L_v^{BR}$. If the while loop terminates, every edge in

b is also in M_{cr} and $L_v^{TL} \cup L_v^{BR} = \emptyset$. Consequently, by Lemma 2, b contains no spurs and old-bar-expansion is ws-equivalent. This eliminates all nodes in the interior of D_b.

Termination. Each pin-extraction and V-shortcut operation reduces the number of vertices of P within D_b. Operation L-shortcut(v, X), $X \in \{TR, TL, BR, BL\}$, either reduces the number of interior vertices, or produces a crimp if edge $[u_1, u_2]$ is a longest edge of a cross-chain. For termination, it is enough to show that, for each cross-chain $c \in \mathcal{P}$, the algorithm introduces a crimp at most once in steps (3)–(4), and at most once in steps (5)–(6). W.l.o.g., consider step (3). We apply an L-shortcut in two possible cases. We show that it does not introduce crimps in Case 2. In step (3), we only perform L-shortcut(v', TL) if (B3) is satisfied and $x(u_1') < x(u_{\max})$. So for all $[v', u_1', u_2'] \in L_{v'}^{TL}$, we have $x(u_1) < x(u_2')$. Suppose, for contradiction, that $[u_1', u_2']$ is the only longest edge of some cross chain (and hence L-shortcut would introduce a crimp). Then, $[u_1', u_2'] \in M_{cr}$ is inside $[u_1, u_{\max}]$, contradicting (A3).

Consider Case 1. Notice that L-shortcut(v, TR) is executed only if there exists a top node p with $x(p) < x(u_1)$ such that $\widehat{L_p}^{TR} \neq \emptyset$. Suppose that L-shortcut(v, TR) introduces a crimp in the path $[v, u_1, u_2] \in L_v^{TR}$. This operation removes this subpath of a cross chain from L_v^{TR}, but introduces $[v^*, u_2, u_1]$ into $L_{v^*}^{TL}$. By the time the algorithm executes L-shortcut(v^*, TL), we know that for every top vertex p with $x(p) < x(u_1)$, $\widehat{L_p}^{TR} = L_p^{TL} = \emptyset$. This implies that, after L-shortcut(v^*, TL) is performed, although a path $[v^{**}, u_1, u_2]$ is introduced in $L_{v^{**}}^{TR}$, L-shortcut(v^{**}, TR) can never be performed. The same arguments apply to steps (5)–(6). ◀

▶ **Lemma 5.** *Algorithm* **bar-simplification**(P, b) *takes* $O(m \log m)$ *time, where* m *is the number of vertices in* b.

Proof. pin-extraction, V-shortcut, and L-shortcut each make $O(1)$ changes in the image graph. pin-extraction and V-shortcut decrease the number of vertices inside D_b. Each L-shortcut does as well, but they may jointly create $2|\mathcal{P}| = O(m)$ crimps, by Lemma 3. So the total number of operations is $O(m)$. When $[v, u_1, u_2] \in L_v^{TR}$ and $u_2 \neq u_{\min}$, L-shortcut replaces $[v, u_1, u_2]$ by $[v^*, u_{\min}, u_2]$: $[u_1]$ shifts to $[u_2]$, but no vertex is eliminated. In the worst case, one L-shortcut modifies $\Theta(m)$ paths, so in $\Theta(m)$ operations the total number of vertex shifts is $\Theta(m^2)$. Our implementation does not maintain the paths in $\mathcal{P}$ explicitly. Instead, we use set operations. We maintain the sets $\mathcal{P}in$, $\mathcal{V}$, and L_v^X, with $v \in \partial D_b$ and $X \in \{TR, TL, BR, BL\}$, in sorted lists. The pins $[v, u, v] \in \mathcal{P}in$ are sorted by $x(v)$; the wedges $[v_1, u, v_2] \in \mathcal{V}$ are sorted by $|x(v_1) - x(v_2)|$. In every set L_v^X, the first two nodes in the paths $[v, u_1, u_2] \in L_v^X$ are the same by (I3), and so it is enough to store vertex $[u_2]$; these vertices are stored in a list sorted by $x(u_2)$. We also maintain binary variables to indicate for each path $[v, u_1, u_2] \in L_v^X$ whether it is part of a cross chain, and whether $[u_1, u_2]$ is the only longest edge of that chain.

Steps (1)-(2) remove pins and V-chains, taking linear time in the number of removed vertices, without introducing any path in any set. Consider L-shortcut(v, TR), executed in one of steps (3)–(4) which can be generalized to other occurrences of the L-shortcut operation. The elements $[v, u_1, u_{\min}] \in L_v^{TR}$ are simplified to $[v^*, u_{\min}]$. For each of these paths, say that the next edge along P is $[u_{\min}, u_3]$. Then, the paths $[v^*, u_{\min}, u_3]$ are inserted into either $\mathcal{P}in \cup \mathcal{V}$ if $u_3 \in \partial D_b$ is a top vertex, or $L_{v^*}^{TL}$ if $u_3 \in b$. We can find each chain $[v, u_1, u_{\min}] \in L_v^{TR}$ in $O(1)$ time since L_v^{TR} is sorted by $x(u_2)$. Finally, all other paths $[v, u_1, u_2] \in L_v^{TR}$, where $u_2 \neq u_{\min}$, become $[v^*, u_{\min}, u_2]$ and they form the new set $L_{v^*}^{TR}$. Since we store only the last vertex $[u_2]$, which is unchanged, we create $L_{v^*}^{TR}$ at no cost.

This representation allows the manipulation of $O(m)$ vertices with one set operation. The number of insert and delete operations in the sorted lists is proportional to the number

of vertices that are removed from the interior of D_b, which is $O(m)$. Each insertion and deletion takes $O(\log m)$ time, and the overall time complexity is $O(m \log m)$. ◀

5 Spur-elimination

When there are no forks in the polygon, we can decide weak simplicity using [4][Theorem 5.1], but a naïve implementation runs in $O(n^2 \log n)$ time: successive applications of spur-reduction would perform an operation at each dummy vertex. Here, we show how to eliminate spurs in $O(n \log n)$ time. After the bar simplification phase, each vertex of P belongs to a cluster.

Formation of Groups. Recall that by (I1) each cluster induces a tree. We first modify the image graph, transforming each tree into a binary tree by adding children to nodes with degree higher than 3. This does not affect the benchmark representation and is ws-equivalent (it can be reversed by node-splits and merges). By construction, if a segment uv connects nodes in different clusters, both u and v are leaves or both are root nodes. We define a *group* G_{uv} as the set of two clusters $C(u)$ and $C(v)$ if their roots are connected by the segment uv.

Recall that we only store benchmark vertices in each cluster. We denote by $[u_1; \ldots; u_k]$ (using semicolons) a path inside a group defined by the benchmark vertices $u_1, \ldots, u_k$. Let $\mathcal{B}$ be the set of paths between two consecutive benchmark vertices in G_{uv}. By invariants (I1), (I2) and (I4), every path in $\mathcal{B}$ has one endpoint in $T[u]$ and one in $T[v]$ and every spur in G_{uv} is incident to two paths in $\mathcal{B}$.

Overview. Assume that $\mathcal{G}$ is a partition of the nodes of the image graph into groups satisfying (I1)–(I5). We consider one group at a time, and eliminate all spurs from one cluster of that group. When we process one group, we may split it into two groups, create a new group, or create a new spur in an adjacent group (similar to pin-extraction in Section 4). The latter operation implies that we may need to process a group several times. Termination is established by showing that each operation reduces the total number of benchmark vertices.

Algorithm spur-elimination$(P, \mathcal{G})$.
While P contains a spur, do:

1. Choose a group $G_{uv} \in \mathcal{G}$ that contains a spur, *w.l.o.g.* contained in $T[u]$.
2. While $T[u]$ contains an interior node, do:
   a. If u contains no spurs and is incident to only two edges uv and uw, eliminate u with a merge operation. The node w is the new root of the tree.
   b. If u contains spurs, eliminate them as described below.
   c. If u contains no spurs, split G_{uv} into two groups along a chain of segments starting with uv as described below. Rename a largest resulting group to G_{uv}.

The detailed description of steps 2b and 2c, as well as the analysis of the algorithm and the supporting data structures are in the full paper [1]. Here we give a brief summary. Step 2b first replaces every path of the form $[t_1; u; t_2]$ by a path $[t_1; t_2]$ (Fig. 17(a)–(b)). The resulting new path $[t_1; t_2]$ passes through the lowest common ancestor of t_1 and t_2, denoted $\mathrm{lca}(t_1, t_2)$. Notice that t_1 and t_2 belong to $T[v]$, therefore $[t_1; t_2]$ does not satisfy (I2). We complete Step 2b by a sequence of "repair" operations that restore (I2). One is analogous to pin-extraction, moving a spur from a leaf of $T[v]$ into an adjacent group (Fig. 17(b)–(c)). The other is analogous to V-shortcut: it creates a new group for each node ℓ where $\ell = \mathrm{lca}(t_1, t_2)$ for some path $[t_1; t_2]$ that violates (I2). The set of all $[t_1; t_2]$ for which $\mathrm{lca}(t_1, t_2) = \ell$ induces

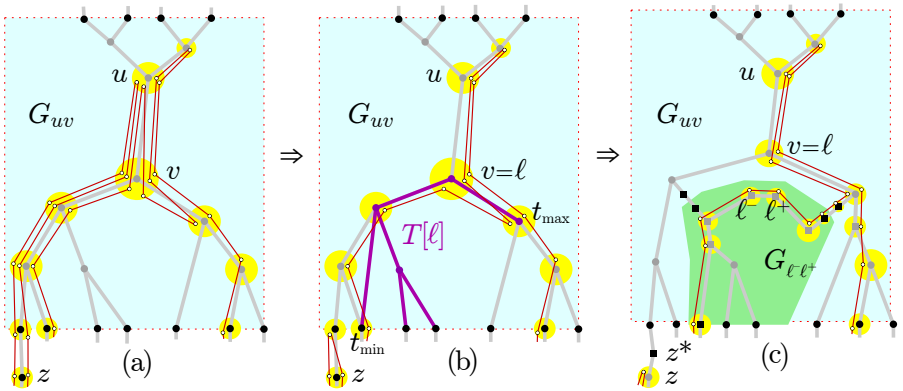


Figure 17 (a) u contains spurs. (b) After eliminating spurs, $T[v]$ does not satisfy (I2). (c) The analogues of pin-extraction and V-shortcut. Leaf nodes are shown black.

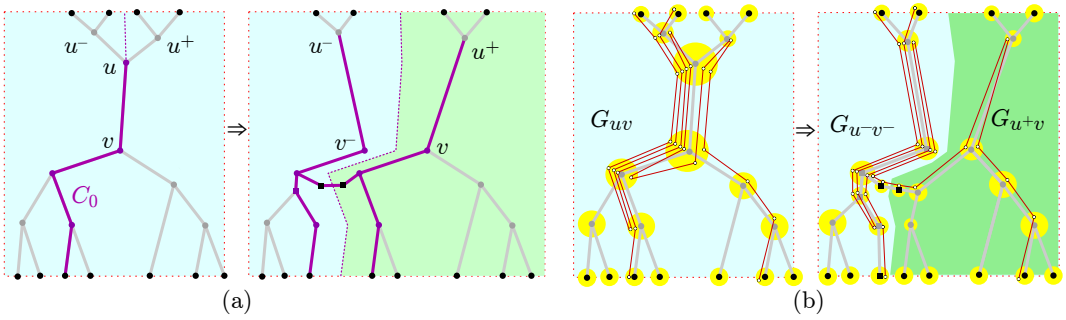


Figure 18 Splitting group G_{uv}. (a) Changes in the image graph. (b) Changes in the polygon.

a tree $T[\ell]$, in which t_1 and t_2 are in the left and right subtree of ℓ, respectively. We then remove all such paths from $T[v]$ and create a new group that satisfies (I1)–(I5).

Step 2c splits G_{uv} into two groups along a chain of segments C_0 (Fig. 18(a)–(b)). The tree $T[u]$ naturally splits into the left and right subtrees, $T[u^-]$ and $T[u^+]$, but splitting $T[v]$ is a nontrivial task. $\mathcal{B}$ is partitioned into $\mathcal{B}^-$ and $\mathcal{B}^+$, paths with one endpoint in $T[u^-]$ and $T[u^+]$, respectively. C_0 represents the shared boundary between $\mathcal{B}^-$ and $\mathcal{B}^+$ and can be found in $O(\log|\mathcal{B}|)$ time. W.l.o.g., $|\mathcal{B}^-| < |\mathcal{B}^+|$. We build a tree $T[v^-]$ induced by $\mathcal{B}^-$ and adjust $T[v]$ so that it is induced by $\mathcal{B}^+$. This is accomplished in $O(|\mathcal{B}^-|)$ time by deleting from $T[v]$ nodes unique to $T[v^-]$. The spurs connected to one path in $\mathcal{B}^-$ and one in $\mathcal{B}^+$ are replaced by a pair of benchmarks on the boundary of the two groups. By a heavy path decomposition argument, the overall time spent in this step is $O(n \log n)$.

6 Conclusion

We presented an $O(n \log n)$-time algorithm for deciding whether a polygon with n vertices is weakly simple. The problem has a natural generalization for planar graphs [4]. It is an open problem to decide efficiently whether a drawing of a graph H is weakly simple, i.e., whether a drawing P of H is within ε Fréchet distance from an embedding Q of H, for all $\varepsilon > 0$.

We can also generalize the problem to higher dimensions. A polyhedron can be described as a map $\gamma : \mathbb{S}^2 \to \mathbb{R}^3$. A simple polyhedron is an injective function. A polyhedron P is weakly simple if there exists a simple polyhedron within ε Fréchet distance from P for all $\varepsilon > 0$. This problem can be reduced to origami flat foldability. The results of [3] imply

that, given a convex polygon P and a piecewise isometric function $f : P \to \mathbb{R}^2$ (called *crease pattern*), it is NP-hard to decide if there exists an injective embedding of P in three dimensions $\lambda : P \to \mathbb{R}^3$ within ε Fréchet distance from f for all $\varepsilon > 0$, i.e., if f is *flat foldable*. Given P and f, we can construct a continuous function $g : \mathbb{S}^2 \to P$ mapping each hemisphere of $\mathbb{S}^2$ to P ($g^{-1}(x)$, for a point $x \in P$, maps to two points in different hemispheres of $\mathbb{S}^2$). Then, the polyhedron $\gamma = g \circ f$ is weakly simple if and only if f is flat foldable. Therefore, it is also NP-hard to decide whether a polyhedron is weakly simple.

References

1 Hugo A. Akitaya, Greg Aloupis, Jeff Erickson, and Csaba D. Tóth. Recognizing weakly simple polygons. Preprint, 2016. `arXiv:1603.07401`.

2 Esther M. Arkin, Michael A. Bender, Erik D. Demaine, Martin L. Demaine, Joseph S.B. Mitchell, Saurabh Sethia, and Steven S. Skiena. When can you fold a map? *Computational Geometry*, 29(1):23–46, 2004.

3 Marshall Bern and Barry Hayes. The complexity of flat origami. In *Proc. 7th ACM-SIAM Sympos. on Discrete Algorithms*, pages 175–183, 1996.

4 Hsien-Chih Chang, Jeff Erickson, and Chao Xu. Detecting weakly simple polygons. In *Proc. 26th ACM-SIAM Sympos. on Discrete Algorithms*, pages 1655–1670, 2015.

5 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, 6(3):485–524, 1991.

6 Pier Francesco Cortese, Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia. On embedding a cycle in a plane graph. *Discrete Math.*, 309(7):1856–1869, 2009.

7 Ares Ribó Mor. *Realization and counting problems for planar structures*. PhD thesis, Freie Universität Berlin, Department of Mathematics and Computer Science, 2006.

8 Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *Proc. 17th IEEE Sympos. Foundations of Computer Science*, pages 208–215, 1976.

Tight Lower Bounds for Data-Dependent Locality-Sensitive Hashing

Alexandr Andoni[*1] **and Ilya Razenshteyn**[2]

1 Columbia University, New York, USA
2 MIT CSAIL, Cambridge, USA

Abstract

We prove a tight lower bound for the exponent ρ for data-dependent Locality-Sensitive Hashing schemes, recently used to design efficient solutions for the c-approximate nearest neighbor search. In particular, our lower bound matches the bound of $\rho \leq \frac{1}{2c-1} + o(1)$ for the ℓ_1 space, obtained via the recent algorithm from [Andoni-Razenshteyn, STOC'15].

In recent years it emerged that data-dependent hashing is strictly superior to the classical Locality-Sensitive Hashing, when the hash function is data-*independent*. In the latter setting, the best exponent has been already known: for the ℓ_1 space, the tight bound is $\rho = 1/c$, with the upper bound from [Indyk-Motwani, STOC'98] and the matching lower bound from [O'Donnell-Wu-Zhou, ITCS'11].

We prove that, even if the hashing is data-dependent, it must hold that $\rho \geq \frac{1}{2c-1} - o(1)$. To prove the result, we need to formalize the exact notion of data-dependent hashing that also captures the complexity of the hash functions (in addition to their collision properties). Without restricting such complexity, we would allow for obviously infeasible solutions such as the Voronoi diagram of a dataset. To preclude such solutions, we require our hash functions to be succinct. This condition is satisfied by all the known algorithmic results.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Similarity search, high-dimensional geometry, LSH, data structures, lower bounds

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.9

1 Introduction

We study lower bounds for the high-dimensional nearest neighbor search problem, which is a problem of major importance in several areas, such as databases, data mining, information retrieval, computer vision, computational geometry, signal processing, etc. This problem suffers from the "curse of dimensionality" phenomenon: either space or query time are exponential in the dimension d. To escape this curse, researchers proposed *approximation* algorithms for the problem. In the (c, r)-approximate near neighbor problem, the data structure may return any data point whose distance from the query is at most cr, for an approximation factor $c > 1$ (provided that there exists a data point within distance r from the query). Many approximation algorithms are known for this problem: e.g., see surveys [22, 3, 1, 24].

An influential algorithmic technique for the approximate near neighbor search (ANN) is the *Locality Sensitive Hashing* (LSH) [13, 12]. The main idea is to hash the points so that the

[*] Work done in part while the first author was at the Simons Institute for the Theory of Computing, Berkeley University.


© Alexandr Andoni and Ilya Razenshteyn;
licensed under Creative Commons License CC-BY

probability of collision is much higher for points that are close to each other (at distance $\leq r$) than for those which are far apart (at distance $> cr$). Given such hash functions, one can retrieve near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. If the probability of collision is at least p_1 for the close points and at most p_2 for the far points, the algorithm solves the (c, r)-ANN using essentially $O(n^{1+\rho}/p_1)$ extra space and $O(dn^\rho/p_1)$ query time, where $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$ [12]. The value of the exponent ρ thus determines the "quality" of the LSH families used.

Consequently, a lot of work focused on understanding the best possible value ρ for LSH, including the sequence of upper bounds [13, 9, 2] and lower bounds [18, 19]. Overall, they established the precise bounds for the best value of ρ: for ℓ_1 the tight bound is $\rho = \frac{1}{c} \pm o(1)$. In general, for ℓ_p, where $1 \leq p \leq 2$, the tight bound is $\rho = \frac{1}{c^p} \pm o(1)$.

Surprisingly, it turns out there exist *more efficient* ANN data structures, which step outside the LSH framework. Specifically, [5, 6] design algorithms using the concept of *data-dependent hashing*, which is a randomized hash family that itself adapts to the actual given dataset. In particular, the result of [6] obtains an exponent $\rho = \frac{1}{2c^p-1} + o(1)$ for the ℓ_p space, thus improving upon the best possible LSH exponent essentially by a factor of 2 for both ℓ_1 and ℓ_2 spaces.

Our result. Here we prove that the exponent $\rho = \frac{1}{2c^p-1}$ from [6] is essentially optimal even for data-dependent hashing, and cannot be improved upon. Stating the precise theorem requires introducing the precise model for the lower bound, which we accomplish below. For now, we state our main theorem informally:

▶ **Theorem 1** (Main, informal). *Any data-dependent hashing scheme for ℓ_1 that achieves probabilities p_1 and p_2 must satisfy*

$$\rho = \frac{\log 1/p_1}{\log 1/p_2} \geq \frac{1}{2c-1} - o(1),$$

as long as the description complexity of the hash functions is sufficiently small.

An immediate consequence is that $\rho \geq \frac{1}{2c^p-1} - o(1)$ for all ℓ_p with $1 \leq p \leq 2$, using the embedding from [17].

1.1 Lower Bound Model

To state the precise theorem, we need to formally describe what is data-dependent hashing. First, we state the definition of (data-independent) LSH, as well as define LSH for a fixed dataset P.

▶ **Definition 2** (Locality-Sensitive Hashing). We say that a hash family $\mathcal{H}$ over $\{0,1\}^d$ is (r_1, r_2, p_1, p_2)-*sensitive*, if for every $u, v \in \{0,1\}^d$ one has:

- if $\|u - v\|_1 \leq r_1$, then $\Pr_{h\sim\mathcal{H}}[h(u) = h(v)] \geq p_1$;
- if $\|u - v\|_1 > r_2$, then $\Pr_{h\sim\mathcal{H}}[h(u) = h(v)] \leq p_2$.

We now refine the notion of data-independent LSH, where we require the distribution to work only for a particular dataset P.

▶ **Definition 3** (LSH for a dataset P). A hash family $\mathcal{H}$ over $\{0,1\}^d$ is said to be (r_1, r_2, p_1, p_2)-*sensitive for a dataset $P \subseteq \{0,1\}^d$*, if:

- for every $v \in \{0,1\}^d$ and every $u \in P$ with $\|u - v\|_1 \leq r_1$ one has

$$\Pr_{h\sim\mathcal{H}}[h(u) = h(v)] \geq p_1;$$

- $\Pr_{\substack{h\sim\mathcal{H}\\u,v\sim P}}[h(u)=h(v) \text{ and } \|u-v\|_1 > r_2] \le p_2$.

Note that the second definition is less stringent than the first one: in fact, if all the points in a dataset are at distance more than r_2 from each other, then an LSH family $\mathcal{H}$ is also LSH for P, but not necessarily vice versa! Furthermore, in the second definition, we require the second property to hold only *on average* (in contrast to *every* point as in the first definition). This aspect means that, while Definition 3 is certainly necessary for an ANN data structure, it is not obviously *sufficient*. Indeed, the algorithm from [6] requires proving additional properties of their partitioning scheme, and in particular analyzes *triples* of points. Since here we focus on lower bounds, this aspect will not be important.

We are now ready to introduce data-dependent hashing.

1.1.1 Data-dependent hashing

Intuitively, a data-dependent hashing scheme is one where we can pick the family $\mathcal{H}$ *as a function of* P, and thus be sensitive for P. An obvious solution would hence be to choose $\mathcal{H}$ to consist of a single hash function h which is just the Voronoi diagram of the dataset P: it will be $(r, cr, 1, 0)$-sensitive for P and hence $\rho = 0$. However this does not yield a good data strcture for ANN since *evaluating* such a hash function on a query point q is as hard as the original problem!

Hence, ideally, our lower bound model would require that the hash function is *computationally efficient* to evaluate. We do not know how to formulate such a condition which would not make the question as hard as circuit lower bounds or high cell-probe lower bounds, which would be well beyond the scope of this paper.

Instead, we introduce a condition on the hash family that can be roughly summarized as "hash functions from the family are succinct". For a precise definition and discussion see below. For now, let us point out that all the known algorithmic results satisfy this condition.

Finally, we are ready to state the main result formally.

▶ **Theorem 4** (Main theorem, full). *Fix the approximation $c > 1$ to be a constant. Suppose the dataset P lives in the Hamming space $\{0,1\}^d$, where the dataset size $n = |P|$ is such that $d = \omega(\log n)$ as n tends to infinity. There exist distance thresholds r and $(c - o(1))r$ with the following property.*

Suppose there exist T hash functions $\{h_i\}_{1\le i\le T}$ over $\{0,1\}^d$ such that for every n-point dataset P there exists a distribution $\mathcal{H}_P$ over h_i's such that the corresponding hash family is $(r, (c - o(1))r, p_1, p_2)$-sensitive for P, where $0 < p_1, p_2 < 0.99$. For any such data-dependent scheme it must either hold that $\rho = \frac{\log 1/p_1}{\log 1/p_2} \ge \frac{1}{2c-1} - o(1)$ or that $\frac{\log T}{p_1} \ge n^{1-o(1)}$.

1.1.2 Interpreting Theorem 4

Let us explain the conditions and the conclusions of Theorem 4 in more detail.

We start by interpreting the conclusions. As explained above, the bound $\rho = \frac{\log 1/p_1}{\log 1/p_2} \ge \frac{1}{2c-1} - o(1)$ directly implies the lower bound on the query time $n^{\frac{1}{2c-1}-o(1)}$ for any scheme that is based on data-dependent hashing.

The second bound $\frac{\log T}{p_1} \ge n^{1-o(1)}$ is a little bit more mysterious. Let us now explain what it means precisely. The quantity $\log T$ can be interpreted as the *description complexity* of a hash function sampled from the family. At the same time, if we use a family with collision probability p_1 for close points, we need at least $1/p_1$ hash tables to achieve constant probability of success. Since in each hash table we evaluate at least one hash function, the

quantity $\frac{\log T}{p_1}$ can be interpreted as the lower bound for the *total space occupied by hash functions we evaluate during each query.* In all known constructions of (data-independent or data-dependent) LSH families [13, 9, 2, 23, 5, 6] the *evaluation time* of a single hash function is comparable to the space it occupies (for discussion regarding why is it true for [6], see Appendix A), thus, under this assumption, we can not achieve query time $n^{1-\Omega(1)}$, unless $\rho \geq \frac{1}{2c-1} - o(1)$. On the other hand, we can achieve $\rho = 0$ by considering a data-dependent hash family that consists only of the Voronoi diagram of a dataset (trivially, $p_1 = 1$ and $p_2 = 0$ for this case), thus the conclusion $\frac{\log T}{p_1} \geq n^{1-o(1)}$ can not be omitted in general[1]. Note that the "Voronoi diagram family" is very slow to evaluate: to locate a point we need to solve an instance of exact Nearest Neighbor Search, that is unlikely to be possible to do in strongly sublinear time. Thus, this family satisfies the above assumption "evaluation time is comparable to the space".

We now turn to interpreting the conditions. We require that $d = \omega(\log n)$. We conjecture that this requirement[2] is necessary and for $d = O(\log n)$ there is an LSH family that gives a better value of ρ (the improvement, of course, would depend on the hidden constant in the expression $d = O(\log n)$). In fact, some evidence towards this conjecture is given in a recent paper [7], where an improved data structure for ANN for the case $d = O(\log n)$ is presented, albeit by stepping outside of the pure (data-dependent) LSH framework.

1.2 Techniques and Related Work

There are two components to our lower bound, i.e., Theorem 4.

The first component is a lower bound for *data-independent* LSH for a *random dataset.* We show that in this case, we must have $\rho \geq \frac{1}{2c-1} - o(1)$. This is in contrast to the lower bound of [19], who achieve a higher lower bound but for the case when the (far) points are correlated. Our lower bound is closer in spirit to [18], who also consider the case when the far points are random uncorrelated. In fact, this component is a strengthening of the lower bound from [18].

We mention that, in [10], Dubiner has also considered the setting of a random dataset for a related problem – finding the closest pair in a given dataset P. Dubiner sets up a certain related "bucketing" model, in which he conjectures the lower bound, which would imply a $\rho \geq \frac{1}{2c-1}$ lower bound for data-independent LSH for a random set. Dubiner verifies the conjecture computationally and claims it is proved in a different manuscript.[3]

We also point out that, for the ℓ_2 case, the optimal data-independent lower bound $\rho \geq \frac{1}{2c^2-1} - o(1)$ follows from a recent work [4]. In fact, it shows almost exact trade-off between p_1 and p_2 (not only the lower bound on $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$). Unfortunately, the techniques there are really tailored to the Euclidean case (in particular, a powerful isoperimetric inequality of Feige and Schechtman is used [11]) and it is unclear how to extend it to ℓ_1, as well as, more generally, to ℓ_p for $1 \leq p < 2$.

Our second component focuses on the data-dependent aspect of the lower bound. In particular, we prove that if there exists a data-dependent hashing scheme for a random dataset with a better ρ, then in fact there is also a data-independent such hashing scheme.

---

[1] For the Voronoi diagram, $\log T \geq n$, since to specify it, one needs at least n bits.

[2] When ANN for the general dimension d is being solved, one usually first performs some form of the dimensionality reduction [14, 16, 8]. Since at this stage we do not want distances to be distorted by a factor more than $1 + o(1)$, the target dimension is precisely $\omega(\log n)$. So, the assumption $d = \omega(\log n)$ in Theorem 4 in some sense captures a *truly high-dimensional case.*

[3] This manuscript does not appear to be available at the moment of writing of the present paper.

To accomplish this, we consider the "empirical" average p_1 and p_2 for a random dataset, and prove it is close to the average p_1 and p_2, for which we can deduce a lower bound from the first component.

In terms of related work, we also must mention the papers of [20, 21], who prove cell-probe lower bounds for the same problem of ANN. In particular, their work utilizes the lower bound of [18] as well. Their results are however incomparable to our results: while their results are unconditional, our model allows us to prove a much higher lower bound, in particular matching the best algorithms.

2 Data-independent Lower Bound

In this section we prove the first component of Theorem 4. Overall we show a lower bound of $\rho \geq \frac{1}{2c-1} + o(1)$ for data-independent hash families for random datasets. Our proof is a strengthening and simplification of [18]. The final statement appears as Corollary 7. In the second component, we will use a somewhat stronger statement, Lemma 6).

For $u \in \{0,1\}^d$ and $0 \leq \alpha \leq 1/2$ define a random variable $N_\alpha(u)$ distributed over $\{0,1\}^d$ as follows: we set every bit of $N_\alpha(u)$ to the corresponding bit of u with probability $1-\alpha$ and with the remaining probability α we flip the bit.

The following lemma is proved via a combination of the averaging argument from [18] and an estimate using Hypercontractivity on the Boolean cube that can be found, e.g., in the proof of Lemma 3.6 of the arXiv version of [15].

▶ **Lemma 5.** *For every hash function $h\colon \{0,1\}^d \to \mathbb{Z}$ and every $0 \leq \alpha \leq 1/2$ one has:*

$$\Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}[h(u)=h(v)] \leq \Pr_{u,v\sim\{0,1\}^d}[h(u)=h(v)]^{\frac{\alpha}{1-\alpha}}.$$

Proof. Let $A \subseteq \{0,1\}^d$ be a subset of the hypercube. Then, from the proof of Lemma 3.6 from the arXiv version of [15], we have

$$\Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}[v \in A \mid u \in A] \leq \left(\frac{|A|}{2^d}\right)^{\frac{\alpha}{1-\alpha}}. \tag{1}$$

Low let us finish the proof using (1). For every $h\colon \{0,1\}^d \to \mathbb{Z}$ one has:

$$\begin{aligned}\Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}[h(u)=h(v)] &= \sum_{k\in\mathbb{Z}} \Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}[v \in h^{-1}(k) \mid u \in h^{-1}(k)] \cdot \left(\frac{|h^{-1}(k)|}{2^d}\right) \\ \leq \sum_{k\in\mathbb{Z}} \left(\frac{|h^{-1}(k)|}{2^d}\right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{|h^{-1}(k)|}{2^d}\right) &= \mathop{\mathrm{E}}_{u\sim\{0,1\}^d}\left[\left(\frac{|h^{-1}(h(u))|}{2^d}\right)^{\frac{\alpha}{1-\alpha}}\right] \\ &\leq \mathop{\mathrm{E}}_{u\sim\{0,1\}^d}\left[\frac{|h^{-1}(h(u))|}{2^d}\right]^{\frac{\alpha}{1-\alpha}} = \Pr_{u,v\sim\{0,1\}^d}[h(u)=h(v)]^{\frac{\alpha}{1-\alpha}},\end{aligned}$$

where the second step follows from (1), and the fourth step follows from the Jensen's inequality. ◀

We are now ready to prove the main lemma of this section, which will be used in the later section on data-dependent hashing. We need to introduce two more definitions. We

define "average p_1" as:

$$\zeta(c,d,\alpha,h) := \Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}\left[h(u)=h(v)\middle|\|u-v\|_1 \le \frac{d}{2c}\right]$$

for $c>1$, positive integer d, $\alpha>0$ and a hash function $h\colon \{0,1\}^d \to \mathbb{Z}$. Similarly, we define the "average p_2" as

$$\eta(d,\beta,h) := \Pr_{u,v\sim\{0,1\}^d}\left[h(u)=h(v), \|u-v\|_1 > \left(\frac{1}{2}-\beta\right)\cdot d\right]$$

for positive integer d, $\beta>0$ and a hash function $h\colon \{0,1\}^d \to \mathbb{Z}$.

▶ **Lemma 6.** *Let $c>1$ be a fixed constant. Suppose that $\gamma=\gamma(d)>0$ is such that $\gamma=o(1)$ as $d\to\infty$. Then, there exist $\alpha=\alpha(d)$, $\beta=\beta(d)$ and $\rho=\rho(d)$ such that:*

$$\alpha = \frac{1}{2c} - o_{c,\gamma}(1),$$
$$\beta = o_\gamma(1),$$
$$\rho = \frac{1}{2c-1} - o_{c,\gamma}(1)$$

as $d\to\infty$ such that, for every d and every hash function $h\colon \{0,1\}^d\to\mathbb{Z}$, one has

$$\zeta(c,d,\alpha,h) \le \eta(d,\beta,h)^\rho + 2^{-\gamma\cdot d}.$$

Proof. We can choose $\alpha = \frac{1}{2c} - o_{c,\gamma}(1)$ such that

$$\Pr\left[\|N_\alpha(u)-u\|_1 \ge \frac{d}{2c}\right] < \frac{2^{-\gamma\cdot d}}{2}. \tag{2}$$

We can choose $\beta=o_\gamma(1)$ so that

$$\Pr_{u,v\sim\{0,1\}^d}\left[\|u-v\| < \left(\frac{1}{2}-\beta\right)\cdot d\right] < \frac{2^{-\gamma\cdot d}}{2}. \tag{3}$$

(Both follow from the standard Chernoff-type bounds.)

Now we apply Lemma 5 and get

$$\Pr_{\substack{u\sim\{0,1\}^d \\ v\sim N_\alpha(u)}}[h(u)=h(v)] \le \Pr_{u,v\sim\{0,1\}^d}[h(u)=h(v)]^\rho, \tag{4}$$

where

$$\rho = \frac{\alpha}{1-\alpha} = \frac{1}{2c-1} - o_{c,\gamma}(1).$$

Finally, we combine (4), (2) and (3), and get the desired inequality. ◀

The following corollary shows how the above lemma implies a lower bound on data-independent LSH.

▶ **Corollary 7.** *For every $c>1$ and $\gamma=\gamma(d)>0$ such that $\gamma=o(1)$ there exists $\beta=\beta(d)>0$ with $\beta=o_\gamma(1)$ such that if $\mathcal{H}$ is a data-independent $(d/(2c),(1/2-\beta)d,p_1,p_2)$-sensitive family, then*

$$p_1 \le p_2^{\frac{1}{2c-1}-o_{c,\gamma}(1)} + 2^{-\gamma\cdot d}.$$

Proof. We observe that for every $\alpha, \beta > 0$:

- $p_1 \leq \mathop{\mathrm{E}}_{h\sim\mathcal{H}}[\zeta(c,d,\alpha,h)]$;
- $p_2 \geq \mathop{\mathrm{E}}_{h\sim\mathcal{H}}[\eta(d,\beta,h)]$.

Now we apply Lemma 6 together with the following application of Jensen's inequality:

$$\mathop{\mathrm{E}}_{h\sim\mathcal{H}}[\eta(d,\beta,h)^\rho] \leq \mathop{\mathrm{E}}_{h\sim\mathcal{H}}[\eta(d,\beta,h)]^\rho,$$

since $0 < \rho \leq 1$. ◀

3 Data-Dependent Hashing

We now prove the second component of the main Theorem 4 proof. In particular we show that a very good data-dependent hashing scheme would refute Lemma 6 from the previous section.

3.1 Empirical Probabilities of Collision

For a particular dataset P, we will be interested in *empirical* probabilities p_1, p_2 – i.e., the equivalents of ζ, η for a given set P – defined as follows. Let $0 < \delta(d) < 1/3$ be some function. Let P be a random set of points from $\{0,1\}^d$ of size $2^{\delta(d)\cdot d}$. The *empirical* versions of ζ and η with respect to P are:

$$\widehat{\zeta}(c,d,\alpha,h,P) := \Pr_{\substack{u\sim P\\ v\sim N_\alpha(u)}}\left[h(u)=h(v)\middle|\|u-v\|_1 \leq \frac{d}{2c}\right]$$

$$\widehat{\eta}(d,\beta,h,P) := \Pr_{u,v\sim P}\left[h(u)=h(v), \|u-v\|_1 > \left(\frac{1}{2}-\beta\right)\cdot d\right].$$

We now want to prove that, for a random dataset P, the empirical ζ, η are close to the true averages. For this we will need the following auxiliary lemma.

▶ **Lemma 8.** *Let M be an $n \times n$ symmetric matrix with entries from $[0;1]$ and average ε. Let M' be a principal $n^\delta \times n^\delta$ submatrix of M sampled uniformly with replacement. Then, for every $\theta > 0$, the probability that the maximum of the average over M' and θ does not lie in $[1/2;2]\cdot\max\{\varepsilon,\theta\}$ is at most $n^\delta \cdot 2^{-\Omega(\theta n^\delta)}$.*

Proof. We need the following version of Bernstein's inequality.

▶ **Lemma 9.** *Suppose that $X_1,\ldots,X_n$ are i.i.d. random variables that are distributed over $[0;1]$. Suppose that $\mathrm{E}[X_i] = \varepsilon$. Then, for every $0 < \theta < 1$, one has*

$$\Pr\left[\max\left\{\frac{1}{n}\sum_{i=1}^n X_i, \theta\right\} \in \left[\frac{1}{2};2\right]\cdot\max\{\varepsilon,\theta\}\right] \geq 1 - 2^{-\Omega(\theta n)}.$$

We just apply Lemma 9 and take union bound over the rows of M'. ◀

The following two lemmas are immediate corollaries of Lemma 9 and Lemma 8, respectively.

▶ **Lemma 10.** *For every $c > 1$, $\alpha > 0$, positive integer d, $\theta > 0$ and a hash function $h\colon \{0,1\}^d \to \mathbb{Z}$, one has*

$$\Pr_P\left[\max\{\widehat{\zeta}(c,d,\alpha,h,P),\theta\} \in [1/2;2]\cdot\max\{\zeta(c,d,\alpha,h),\theta\}\right] \geq 1 - 2^{-\Omega\left(\theta\cdot 2^{\delta(d)\cdot d}\right)}.$$

▶ **Lemma 11.** *For every $\beta > 0$, positive integer d, $\theta > 0$ and a hash function $h\colon \{0,1\}^d \to \mathbb{Z}$, one has*

$$\Pr_P\left[\max\{\widehat{\eta}(d,\beta,h,P),\theta\} \in [1/2;2]\cdot\max\{\eta(d,\beta,h),\theta\}\right] \geq 1 - 2^{\delta(d)\cdot d}\cdot 2^{-\Omega\left(\theta\cdot 2^{\delta(d)\cdot d}\right)}.$$

3.2 Proof of Theorem 4

We are finally ready to complete the proof of the main result, Theorem 4.

Let us first assume that $p_1 = o(1)$ and then show how to handle the general case. Suppose that $n = |P| = 2^{\delta\cdot d}$. By the assumption of Theorem 4, $\delta = o(1)$. Let $\{h_1, h_2, \ldots, h_T\}$ be a set of hash functions. We can assume that $\frac{\log T}{p_1} \leq n^{1-\Omega(1)}$, since otherwise we are done. Let us fix $\gamma = \gamma(d) > 0$ such that $\gamma = o(1)$ and $2^{-\gamma d} \ll p_1^{\omega(1)}$. We can do this, since if $p_1 = 2^{-\Omega(d)}$, then $1/p_1 = n^{\omega(1)}$, and the desired statement is true. Then, by Lemma 6, there is $\alpha = \alpha(d)$ and $\beta = \beta(d) = o_\gamma(1)$ such that for every $1 \leq i \leq T$

$$\zeta(c,d,\alpha,h_i) \leq \eta(d,\beta,h_i)^{\frac{1}{2c-1}-o_{c,\gamma}(1)} + 2^{-\gamma d}. \tag{5}$$

Let us choose $\theta > 0$ such that

$$T\cdot 2^{\delta\cdot d} \ll 2^{\theta\cdot 2^{\delta\cdot d}}$$

and $\theta \ll p_1$. We can do it since, by the above assumption, $\frac{\log T}{p_1} \leq n^{1-\Omega(1)} = 2^{\delta\cdot d(1-\Omega(1))}$.

Then, from Lemma 10 and Lemma 11 we get that, with high probability over the choice of P, one has for every $1 \leq i \leq T$:

- $\max\{\widehat{\zeta}(c,d,\alpha,h_i),\theta\} \in [1/2;2]\cdot\max\{\zeta(c,d,\alpha,h_i),\theta\}$;
- $\max\{\widehat{\eta}(d,\beta,h_i),\theta\} \in [1/2;2]\cdot\max\{\eta(d,\beta,h_i),\theta\}$.

Suppose these conditions hold and assume there exist a distribution $\mathcal{D}$ over $[T]$ such that the corresponding hash family is $(d/2c, (1/2-\beta(d))d, p_1, p_2)$-sensitive for P. Then,

$$\begin{aligned} p_1 \leq \mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\widehat{\zeta}(c,d,\alpha,h_i)] \leq \mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\max\{\widehat{\zeta}(c,d,\alpha,h_i),\theta\}] &\leq 2\cdot\mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\max\{\zeta(c,d,\alpha,h_i),\theta\}] \\ &\leq 2\cdot\mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\zeta(c,d,\alpha,h_i)] + \theta. \end{aligned} \tag{6}$$

Similarly,

$$\begin{aligned} p_2 \geq \mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\widehat{\eta}(d,\beta,h_i)] \geq \mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\max\{\widehat{\eta}(d,\beta,h_i),\theta\}] - \theta &\geq \frac{1}{2}\cdot\mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\max\{\eta(d,\beta,h_i),\theta\}] - \theta \\ &\geq \frac{1}{2}\cdot\mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\eta(d,\beta,h_i)] - \theta. \end{aligned} \tag{7}$$

Averaging (5) and applying Jensen's inequality, we have

$$\mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\zeta(c,d,\alpha,h_i)] \leq \mathop{\mathrm{E}}_{i\sim\mathcal{D}}[\eta(d,\beta,h_i)]^{\frac{1}{2c-1}-o(1)} + 2^{-\gamma(d)\cdot d}. \tag{8}$$

Thus, substituting (6) and (7) into (8)

$$\frac{p_1-\theta}{2} \leq (2(p_2+\theta))^{\frac{1}{2c-1}-o(1)} + 2^{-\gamma d},$$

which proves the theorem, since $\theta \ll p_1$, $2^{-\gamma d} \ll p_1^{\omega(1)}$, and $p_1 = o(1)$.

Now let us deal with the case $p_1 = \Omega(1)$. This can be reduced to the case $p_1 = o(1)$ by choosing a slowly-growing super constant k and replacing the set of T functions with the set of T^k *tuples* of length k. This replaces p_1 and p_2 with p_1^k and p_2^k, respectively. In the same time, we choose k so that $T' = T^k$ still satisfy the hypothesis of the theorem. Then, we just apply the above proof.

References

1. Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, Massachusetts Institute of Technology, 2009.
2. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 459–468, 2006.
3. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.
4. Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. Practical and optimal LSH for angular distance. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS'15)*, 2015.
5. Alexandr Andoni, Piotr Indyk, Huy L. Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1018–1028, 2014.
6. Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the ACM Symposium on the Theory of Computing (STOC'15)*, 2015.
7. Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA'16)*, 2016.
8. Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.
9. Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG'04)*, pages 253–262, 2004.
10. Moshe Dubiner. Bucketing coding and information theory for the statistical highdimensional nearest-neighbor problem. *IEEE Transactions on Information Theory*, 56(8):4166–4179, 2010.
11. Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for MAX CUT. *Random Structures and Algorithms*, 20(3):403–440, 2002.
12. Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.
13. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC'98)*, pages 604–613, 1998.
14. William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Connecticut, 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. 1984.
15. Subhash Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative-type metrics into l_1. *J. ACM*, 62(1):8:1–8:39, 2015.
16. Eyal Kushilevitz, Rafail Ostrovky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.
17. Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.
18. Rajeev Motwani, Assaf Naor, and Rina Panigrahy. Lower bounds on locality sensitive hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007.

19 Ryan O'Donnell, Yi Wu, and Yuan Zhou. Optimal lower bounds for locality sensitive hashing (except when q is tiny). In *Proceedings of Innovations in Computer Science (ICS'11)*, pages 275–283, 2011.

20 Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*, pages 414–423, 2008.

21 Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 805–814, 2010.

22 Hannan Samet. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, 2006.

23 Kengo Terasawa and Yuzuru Tanaka. Spherical LSH for approximate nearest neighbor search on unit hypersphere. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 27–38, 2007.

24 Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014.

A Upper Bounds Are in the Model

In this section, we show how the data-dependent hash family from [6] fits into the model of the lower bound from Section 1.1.

Let us briefly recall the hash family construction from [6]. For simplicity, assume that all points and queries lie on a sphere of radius $R \gg cr$. First, consider a *data-independent* hash family from [5, 6], called *Spherical LSH*. It gives a good exponent $\rho \leq \frac{1}{2c^2-1} + o(1)$ for distance thresholds r vs $\sqrt{2}R$ (the latter corresponds to a typical distance between a pair of points from the sphere). The main challenge that arises is how to handle distance thresholds r vs cr, where the latter may be much smaller than $\sqrt{2}R$. Here comes the main insight of [6].

We would like to process the dataset so that the distance between a typical pair of *data points* is around $\sqrt{2}R$, so to apply Spherical LSH. To accomplish this, we remove all the clusters of radius $(\sqrt{2}-\varepsilon)R$ that contain lots of points (think of $\varepsilon > 0$ being barely sub-constant). We will treat these clusters separately, and will focus on the remainder of the pointset for now. So for the remainder of the pointset, we just apply the Spherical LSH to it. This scheme turns out to satisfy the definition of the data-dependent hash family for the remaining points and for distance thresholds r vs. cr; in particular, the hash function is sensitive for the remaining set only! The intuition is that, for any potential query point, there is only a small number of data points within distance $(\sqrt{2}-\varepsilon)R$ – otherwise, they would have formed yet another cluster, which we would have removed – and the larger distances are handled well by the Spherical LSH. Thus, *for a typical pair of data points*, Spherical LSH is sensitive for P (see Definition 3).

How does [6] deal with the clusters? The main observation is that one can enclose such a cluster in a ball of radius $(1-\Omega(\varepsilon^2))R$, which intuitively makes our problem a little bit simpler (after we reduce the radius enough times, the problem becomes trivial). To answer a query, we query *every cluster*, as well as *one part* of the remainder (partitioned by the Spherical LSH). This can be shown to work overall, in particular, we can control the overall branching and depth of the recursion (see [6] for the details).

For both the clusters, as well as the parts obtained from the Spherical LSH, the algorithm recurses on the obtained point subsets. The overall partitioning scheme from [6] can be seen as a tree, where the root corresponds to the whole dataset, and every node either corresponds to a cluster or to an application of the Spherical LSH. One nuance is that, in

different parts of the tree the Spherical LSH partitioning obtains different p_1, p_2 (depending on R). Nonetheless, each time it holds that $p_1 \geq p_2^\rho$ for $\rho \leq \frac{1}{2c^2-1} + o(1)$. Hence, a node terminates as a leaf once its accumulated p_2 (product of p_2's of "Spherical LSH" nodes along the path from the root) drops below $1/n$.

We now want to argue that the above algorithm can be recast in the framework of data-dependent hashing as per Definition 3. We consider a subtree of the overall tree that contains the root and is defined as follows. Fix a parameter $l = n^{-o(1)}$ (it is essentially the target p_2 of the partition). We perform DFS of the tree and cut the tree at any "Spherical LSH" node where the cumulative p_2 drops below l. This subtree gives a partial partition of the dataset P as follows: for "Spherical LSH" nodes we just apply the corresponding partition, and for cluster nodes we "carve" them in the random order. It turns out that if we choose $l = n^{-o(1)}$ carefully, the partition will satisfy Definition 3 and the preconditions of Theorem 4. In particular, the description complexity of the resulting hash function is $n^{o(1)}$.

Let us emphasize that, while Definition 3 is certainly necessary for an ANN data structure based on data-dependent hashing, it is not *sufficient*. In fact, [6] prove additional properties of the above partitioning scheme, essentially because the "p_2 property" is "on average" one (thus, we end up having to understand how this partitioning scheme treats *triples* of points).

The Number of Holes in the Union of Translates of a Convex Set in Three Dimensions

Boris Aronov[*1], **Otfried Cheong**[†2], **Michael Gene Dobbins**[‡3], **and Xavier Goaoc**[4]

1 **Department of Computer Science and Engineering, Tandon School of Engineering, New York University, New York, USA**
boris.aronov@nyu.edu
2 **KAIST, Daejeon, South Korea**
otfried@kaist.edu
3 **Department of Mathematical Sciences, Binghamton University, Binghamton, USA**
mdobbins@binghamton.edu
4 **Université Paris Est, LIGM (UMR 8049), Marne la Vallée, France**
goaoc@u-pem.fr

Abstract

We show that the union of n translates of a convex body in $\mathbb{R}^3$ can have $\Theta(n^3)$ holes in the worst case, where a *hole* in a set X is a connected component of $\mathbb{R}^3 \setminus X$. This refutes a 20-year-old conjecture. As a consequence, we also obtain improved lower bounds on the complexity of motion planning problems and of Voronoi diagrams with convex distance functions.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Union complexity, Convex sets, Motion planning

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.10

1 Introduction

From path planning in robotics [17] to the design of epsilon-nets [7] to analyzing vulnerabilities in networks [1], a variety of combinatorial and algorithmic problems in computational geometry involve understanding the complexity of the union of n elementary objects. An abundant literature studies how this union complexity depends on the geometry of the objects, and we refer the interested reader to the survey of Agarwal, Pach and Sharir [2]. In the plane, two important types of conditions were shown to imply near-linear union complexity: restrictions on the number of boundary intersections [18, 12] and fatness [8, 16]. In three dimensions, the former are less relevant as they do not apply to important examples such as motion planning problems. Whether fatness implies low union complexity in $\mathbb{R}^3$ has been identified as an important open problem in the area [14, Problem 4]; to quote Agarwal, Pach and Sharir [2, Section 3.1, §2],

> "*A prevailing conjecture is that the maximum complexity of the union of such fat objects is indeed at most nearly quadratic. Such a bound has however proved quite*

* B. Aronov is supported by NSF grants CCF-11-17336 and CCF-12-18791.
† O. Cheong is supported by NRF grant 2011-0030044 (SRC-GAIA) from the government of Korea.
‡ M. G. Dobbins is supported by NRF grant 2011-0030044 (SRC-GAIA) from the government of Korea.

© Boris Aronov, Otfried Cheong, Michael Gene Dobbins, and Xavier Goaoc;
licensed under Creative Commons License CC-BY

elusive to obtain for general fat objects, and this has been recognized as one of the major open problems in computational geometry."

A weaker version of this conjecture asserts that the number of holes in the union of n translates of a fixed convex body in $\mathbb{R}^3$ is at most nearly quadratic in n. By a *hole* in a subset $X \subseteq \mathbb{R}^d$ we mean a connected component of $\mathbb{R}^d \setminus X$. This is indeed a weakening since the number of holes is a lower bound on the complexity and a family of translates can be made quite fat by applying a suitable affine transformation. Evidence in support of the weaker conjecture is that in two dimensions, the union of n translates has at most a linear number of holes [12], and in three dimensions, the union of n translates of a convex polytope with k facets has at most $O(kn^2)$ holes [4], so the number of holes grows only quadratically for any fixed convex polytope.

Remarkably, we refute the conjecture even in this weaker form. We construct a convex body in $\mathbb{R}^3$ that has, for any n, a family of n translates with $\Theta(n^3)$ holes in its union. This matches the upper bound for families of arbitrary convex bodies in $\mathbb{R}^3$ [13].

▶ **Theorem 1.** *The maximum number of holes in the union of n translates of a compact, convex body in $\mathbb{R}^3$ is $\Theta(n^3)$.*

We start with a warm-up example illustrating the idea behind our construction (Section 2). We then construct a polytope K_m, tailored to the value of m considered and with $\Theta(m^2)$ faces; we give a family of $3m$ translates of K_m (in Section 3) whose union has $\Theta(m^3)$ holes. The final step of our construction is to turn the family of polytopes (K_m) into a limiting "universal" convex body K that, for any m, admits $3m$ translates whose union has $\Theta(m^3)$ holes. We prove this formally using arguments from algebraic topology, developed in Section 5.

Further consequences. We conclude this introduction with two examples of problems in computational geometry whose underlying structure involves a union of translates of a convex body, and on which Theorem 1 casts some new light.

A *motion-planning problem* asks whether an object, typically in $\mathbb{R}^2$ or $\mathbb{R}^3$, can move from an initial position to a final position by a sequence of elementary motions while remaining disjoint from a set of obstacles (and to compute such a motion when it exists). This amounts to asking whether two given points lie in the same connected component of the *free space*; that is, the set of positions of the object where it intersects no obstacle. When the motions are restricted to translations, the free space can be obtained by taking the complement of the union of the "expansion" (formally: the *Minkowski sum*) of every obstacle by the reflection of the object through the origin. In the simplest case the mobile object is convex, the obstacles consist of n points and the free space is the complement of the union of n translates of a convex body; Theorem 1 implies that already in this case the free space can have large complexity:

▶ **Corollary 2.** *There exists a set P of n point obstacles and a convex body K in $\mathbb{R}^3$ such that the free space for moving K by translations while avoiding P has $\Theta(n^3)$ connected components.*

The *Voronoi diagram* of a family P of points $p_1, p_2, \ldots, p_n$, called *sites*, in a metric space X is the partition of X according to the closest p_i. A subset $Q \subseteq P$ *defines a face* of the diagram if there exist some point $x \in X$ at equal distance from all sites of Q, and strictly further away from all sites in $P \setminus Q$. A case of interest is when X is $\mathbb{R}^d$ equipped with a *convex distance function* d_U defined by a convex unit ball U (in general, d_U is not a metric, as U needs not be centrally symmetric). A *face* of the Voronoi diagram with respect to d_U is

defined to be a connected component of the set $\{x : d(x,q) \leq d(x,p), \text{ for all } q \in Q, p \in P\}$ for some $Q \subset P$ defining a face; the *complexity* of a Voronoi diagram is measured by the number of its faces of all dimensions. In $\mathbb{R}^2$, the complexity of the Voronoi diagram of n point sites with respect to d_U is $O(n)$, independent of the choice of U [5, 9].

The state of knowledge is less satisfactory in three dimensions where we know near-quadratic complexity bounds for the Voronoi diagram with respect to d_U if U is a constant complexity polytope [19], and that by making U sufficiently complicated one can have four point sites define arbitrarily many Voronoi vertices (that is, isolated points at equal distance from these four sites) [11]. If we grow equal-radii balls (for d_U) centered in each of the p_i simultaneously, every hole in the union of these balls must contain a Voronoi vertex. Theorem 1 therefore implies that one can also find $\Omega(n^3)$ different quadruples of point sites defining Voronoi vertices.

▶ **Corollary 3.** *There exist a convex distance function d_U and a set P of n points in $\mathbb{R}^3$ such that $\Omega(n^3)$ different quadruples of P define a Voronoi vertex in the Voronoi diagram of P with respect to d_U.*

Notation. For an integer n we let $[n]$ denote the set $\{1, \ldots, n\}$. If A and B are two subsets of $\mathbb{R}^d$ we denote by $A + B$ their *Minkowski sum*

$$A + B = \{a + b : a \in A, b \in B\}$$

and by $\text{conv}(A)$ the *convex hull* of A. We refer to coordinates in three space by x, y, z. We refer to the positive z-direction as "upward," and to the positive y-direction as "forward." For an object $A \subset \mathbb{R}^3$, the "front" boundary of A is the upper y-envelope. Similarly, the "back" boundary is the lower y-envelope.

Remark on figures. The reader should take heed that in several places we give explicit coordinates for a construction and provide a figure, where the figure depicts the qualitative geometric features of interest using different coordinates. Using the coordinates chosen for convenience of computation would have resulted in figures where features are too small to see.

2 Many holes touching a single facet

We first introduce the key idea of our constructions with a simpler goal: constructing $2m + 1$ translates of a convex polytope with $\Omega(m^2)$ holes in the union, all incident to a common facet.

Let K be the polytope depicted in Figure 1, the convex hull of the following seven points:

$$a = (0,0,0),\ b = (1,0,0),\ c = (0,0,-1), d = (1,0,-1),$$
$$e = (1/2, 1/2, 1/2),\ f = (0,1,0), \text{ and } g = (1,1,0).$$

We fix an integer $m > 0$ and let $C = K$ be the trivial translate of K. Let F denote the facet of C with vertices a, b, c, d. We then pick m translates of K, denoted $B_1, B_2, \ldots, B_m$, whose top-most vertices (corresponding to vertex e) are placed regularly along the edge ab of F. The top part of the intersection $B_j \cap F$ is a triangular region, shown in blue in Figure 1 (bottom). The union $\bigcup_{j=1}^{m} B_j$ bounds $m - 1$ regions below the edge ab of F. Let ℓ denote the height of one such region; that is the distance between $B_1 \cap B_2$ and the edge ab of F. Let c' denote the point on the segment ac of F at distance ℓ from a (see again

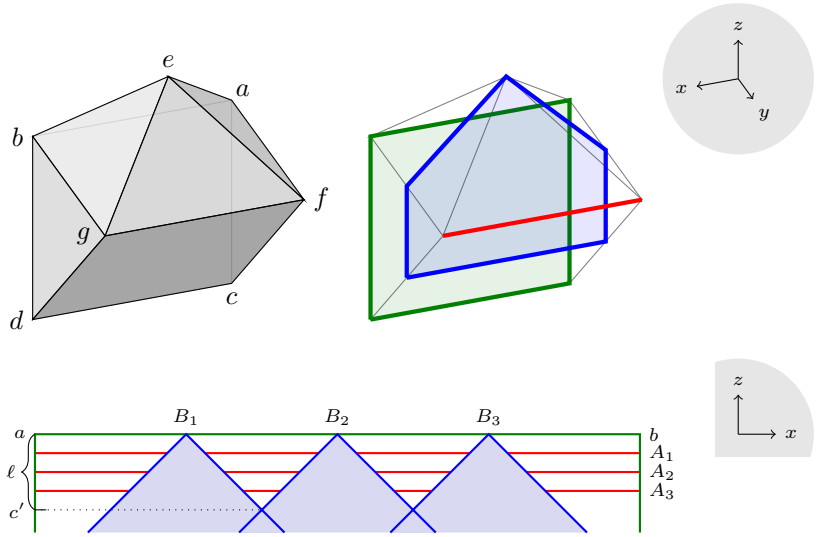


Figure 1 *Top left*: The convex polytope K for the construction of Section 2; *Top right*: three important (x, z) cross-sections of K. The cross section in the back (green) is the facet F that touches many holes, the one in the center (blue) is used to produce vertical cones, and the one in the front (red) is used to produce horizontal edges. *Bottom*: The grid formed on the facet F by horizontal segments and vertical cones.

Figure 1 (bottom)). We next pick m translates of K, denoted $A_1, A_2, \ldots, A_m$, whose vertices corresponding to f are placed regularly along the segment ac'.

The intersections $F \cap B_j$ and $F \cap A_i$, for $i, j \in [m]$, form a grid in F with $m(m-1)$ holes on F. Since two consecutive A_i leave only a narrow tunnel incident to F, and each B_j entirely cuts this tunnel, each of the holes on F is indeed on the boundary of a distinct hole in the union of all translates in $\mathbb{R}^3$.

▶ **Claim 4.** *The union* $\bigcup_{i=1}^{m} A_i \cup \bigcup_{j=1}^{m} B_j \cup C$ *has* $\Omega(m^2)$ *holes, all touching the facet* F *of* C.

Since this is only a warm-up example, we do not include a formal proof of the claim.

3 The construction of K_m

The construction of Section 2 uses three features of K: a portion of a cone with apex e, a portion of a prism with edge fg, and a facet F. We combined the A_i's and the B_j's in a grid-like structure that created $\Theta(m^2)$ local minima in the y-direction on the front boundary of the union $\bigcup A_i \cup \bigcup B_j$. Each of these minima is the bottom of a "pit", and C acts as a "lid" to turn each pit into a separate hole. The construction we use to prove Theorem 1 is based on a similar principle, but before giving this construction, we first fix a value $m > 0$, and then build a convex polytope K_m depending on m and a family of $3m$ translates with $\Theta(m^3)$ holes. The construction of K_m consists of two parts, which we refer to as Front and Back.

The auxiliary paths. To construct our polytope K_m we use two auxiliary polygonal paths η and γ. They both start at the origin 0. The path η has m edges, lies in the (y, z)-plane, and

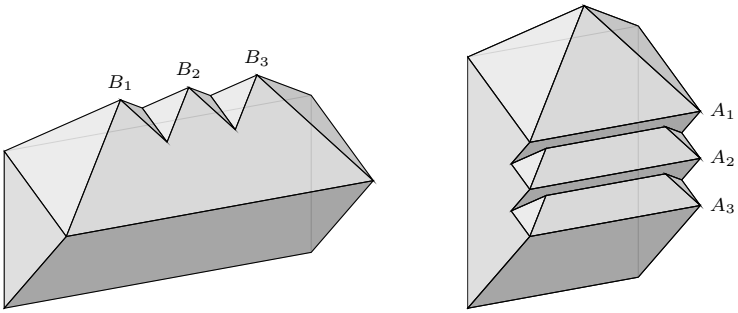


Figure 2 The union of the translates $B_1, B_2, \ldots, B_m$ (left) and $A_1, A_2, \ldots, A_m$ (right). The spacing between the translates is exaggerated for clarity.

is convex[1] in both directions $(0, -1, 0)$ and $(0, 0, -1)$. The path γ has $m + 2$ edges, lies in the (x, y)-plane, and is convex in direction $(0, -1, 0)$. We denote the jth vertex of η by $w_{j,0}$ and the kth vertex of γ by $w_{0,k}$, see the top left of Figure 3.

The front part. We define a "grid" of $(m+1) \times (m+3)$ points, which are the vertices of the polyhedral surface $\eta + \gamma$ (see the top right of Figure 3), by putting

$$w_{j,k} = w_{j,0} + w_{0,k} \quad \text{for } j \in \{0, 1, \ldots, m\} \text{ and } k \in \{0, 1, \ldots, m+2\}.$$

We then add a point $v_{j,k}$ on the edge $w_{j,k}w_{j,k+1}$ as follows (see the bottom left of Figure 3):

$$v_{j,k} = \left(1 - \tfrac{j}{m+1}\right)w_{j,k} + \tfrac{j}{m+1}w_{j,k+1} \quad \text{for } j \in \{0, 1, \ldots, m\} \text{ and } k \in \{0, 1, \ldots, m+1\}.$$

For $j \in \{0, \ldots, m\}$ we define two polygonal paths

$$\gamma_j = w_{j,0} + \gamma = w_{j,0}w_{j,1}w_{j,2} \ldots w_{j,m+1}w_{j,m+2} \text{ and}$$
$$\xi_j = w_{j,0}v_{j,0}v_{j,1}v_{j,2}v_{j,3} \ldots v_{j,m}v_{j,m+1}w_{j,m+2}.$$

Note that $\xi_0 = \gamma_0 = \gamma$, that the paths γ_j are simply translates of γ, and that the path ξ_j lies entirely in the convex region $\gamma_j + (0, \mathbb{R}^-, 0)$ and the vertices of ξ_j lie on γ_j; see Figure 4.

The front part FRONT of K_m is the convex hull of the paths ξ_j (see the bottom right of Figure 3):

$$\text{FRONT} = \text{conv}\left(\xi_0 \cup \xi_1 \cup \cdots \cup \xi_m\right).$$

Observe that the paths η and γ can be chosen so that, for each $j \in [m]$, the path ξ_j lies entirely on the front boundary as well as on the upper boundary of FRONT.

Let $E_{j,k}$ denote the edge $w_{j,k}w_{j,k+1}$ and let $E_k = E_{0,k}$. Consider the point $v_{j,k}$ on $E_{j,k}$. Since ξ_j lies entirely on the upper boundary of FRONT, no part of FRONT appears above $E_{j,k}$, and since $v_{j,k}$ is the only point where the segment $E_{j,k}$ intersects FRONT for $j, k \in [m]$, the vertical plane containing $E_{j,k}$ intersects FRONT in a downward extending cone with apex $v_{j,k}$; see Figure 5. Note that this fails for $k = 0$ and for $k = m+1$.

[1] We say a path π is convex in a *direction* u if the orthogonal projection of π onto $u^\perp$ is injective and the set $\pi + \mathbb{R}^+u$ is convex.

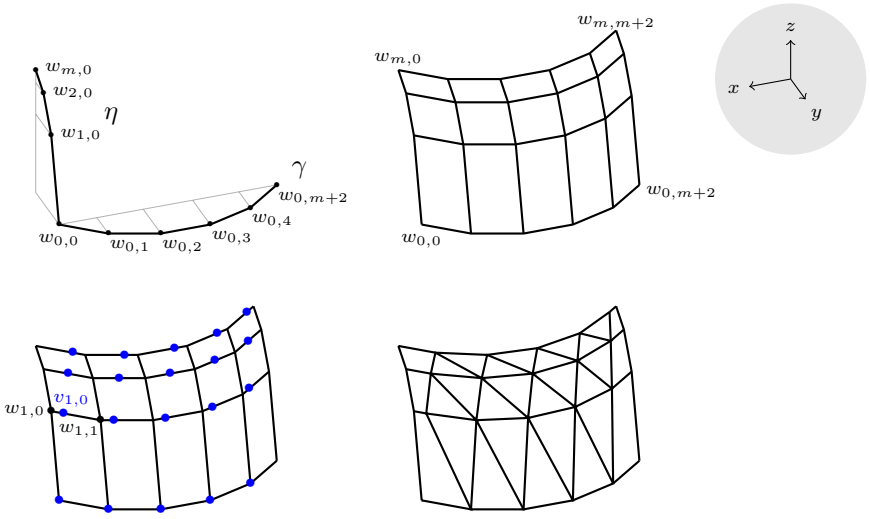


Figure 3 Design of FRONT. *Top left*: The auxiliary paths η and γ. *Top right*: The surface $\eta + \gamma$ defining the "grid" of vertices $w_{j,k}$. *Bottom left*: The points $v_{j,k}$. *Bottom right*: The front boundary of FRONT.

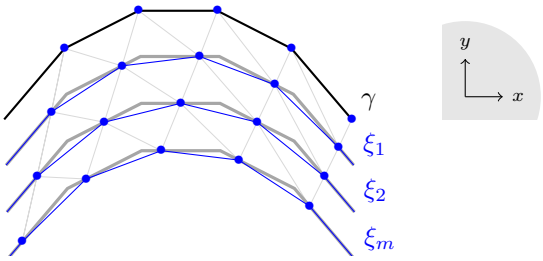


Figure 4 View of the paths ξ_j from above (with the paths γ_j represented in grey).

The back part. We now define two vectors, $u_1 = w_{m,m+2} - (0, t, 0)$, where t is some positive real number, and u_0, the orthogonal projection of u_1 on the (x, y)-plane. We define BACK as the Minkowski sum of the segment u_0u_1 and $-\gamma$, the reflection of γ with respect to the origin (see Figure 6). The value of t is adjusted so that FRONT and BACK have disjoint convex hulls.

By construction, BACK consists of $m+2$ rectangles orthogonal to the (x, y)-plane, namely the rectangles $u_0u_1 - E_k$, for $k \in \{0, \ldots, m+1\}$. The top and bottom edges of each rectangle are $u_1 - E_k$ and $u_0 - E_k$ respectively.

The polytope K_m and its translates. We now let $K_m = \text{conv}(\text{FRONT} \cup \text{BACK})$, see Figure 7, and define three families of its translates. First, for $k \in [m]$, we define a translate C_k such that the edge $u_1 - E_k$ of C_k coincides with the edge E_k of K_m. Formally, we have

$$C_k = K_m + c_k, \quad \text{where} \quad c_k = w_{0,k} - (u_1 - w_{0,k+1}).$$

See Figure 8 (right). The vertical facet formed by $u_1 - E_k$ and $u_0 - E_k$ of C_k will be incident to a quadratic number of holes, playing the same role as the facet F in the previous section.

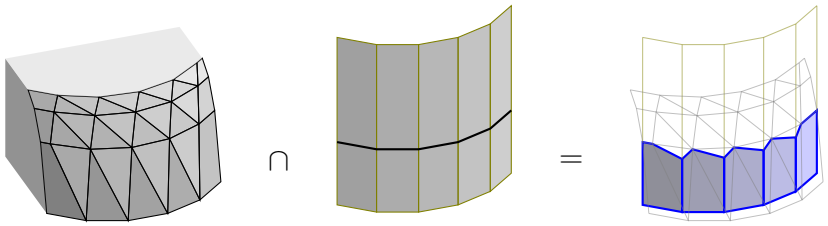


Figure 5 Intersection of vertical panels through the path γ_1 with FRONT.

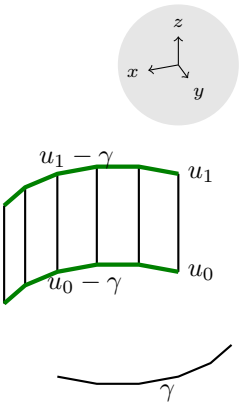


Figure 6 Design of BACK.

Let us denote this facet as R_k.

Next, for $j \in [m]$, we define a translate B_j of K_m as follows:

$$B_j = K_m + b_j, \quad \text{where} \quad b_j = -w_{j,0}.$$

In other words, the path ξ_j of B_j lies in the (x, y)-plane and the vertex $v_{j,k}$ of B_j lies on E_k. See Figure 8 (middle).

Consider now the edge E_k, for $k \in [m]$. For each $j \in [m]$, the edge E_k contains the vertex $v_{j,k}$ of B_j. By the argument above, the intersection of B_j with the facet R_k is a vertical cone with apex $(1 - j/m+1)w_{0,k} + (j/m+1)w_{0,k+1}$. These apices are regularly spaced along E_k, and we obtain a configuration similar to Figure 1. In other words, the union $\bigcup_{j\in[m]} B_j$ bounds $m - 1$ triangular regions below the edge E_k on the facet R_k of C_k.

We now pick a suffiently small number $\varepsilon > 0$ and define our final family of translates. For $i \in [m]$, let

$$A_i = K_m + a_i, \quad \text{where } a_i = \left(0, 0, -\tfrac{i}{m}\varepsilon\right).$$

The translates A_i are defined by translating K_m vertically such that for every $i \in [m]$ and $k \in [m]$, the edge E_k of A_i appears as a horizontal edge on R_k, cutting each of the $m - 1$ triangular regions. See Figure 8 (left).

The family $\mathcal{F}$. We now finally set $\mathcal{F} = \{A_1, \ldots, A_m, B_1, \ldots, B_m, C_1, \ldots, C_m\}$. The union of the family $\mathcal{F}$ has at least $m^2(m - 1)$ holes: For $k \in [m]$, we consider the facet R_k of C_k. On this facet, the union of the B_j and A_i forms a grid with $m(m - 1)$ holes. As in Section 2, we argue that each of these holes is incident to a distinct component of the complement of the union of all the translates.

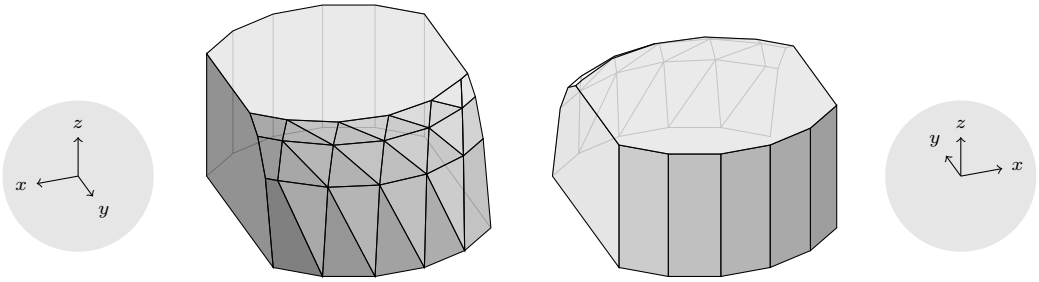


Figure 7 *Left*: The polytope K_m. *Right*: K_m as viewed from behind.

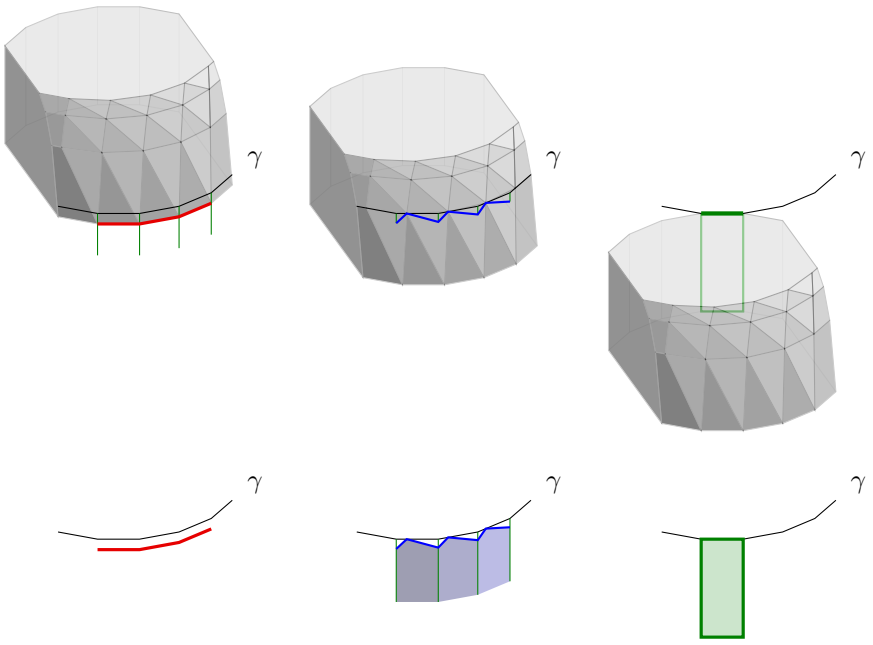


Figure 8 *Top from left*: A translate A_i, B_j, and C_k. *Bottom from left*: The intersection of the respective translates A_i, B_j, and C_k above with the facets $R_1, \ldots, R_m$.

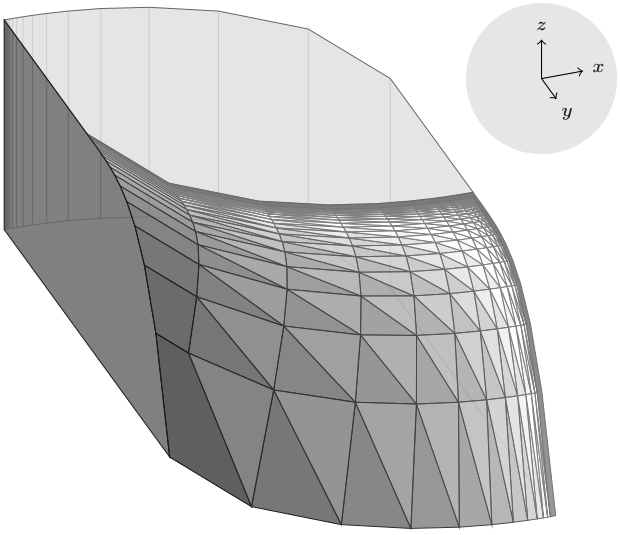


Figure 9 A convex body K, not depending on n, with translates forming $\Theta(n^3)$ holes.

We will not give a formal proof of this fact, since we will give an even stronger construction in Section 4, and we will include a formal, algebraic argument for the correctness of that construction.

Explicit coordinates for K_m. The reader not satisfied with the qualitative description of $\mathcal{F}$ may enjoy verifying that the following coordinates satisfy the properties we needed for our construction:

$$\begin{aligned} w_{j,0} &= (0, -3j, 1 - 2^{-j}) && \text{for } 0 \le j \le m, \\ w_{0,k} &= (\cos\theta_k - 1, \sin\theta_k, 0) && \text{for } \theta_k = \tfrac{\pi}{3}\left(\tfrac{k-1}{m} + 1\right), 1 \le k \le m+1, \\ w_{0,m+2} &= (-2, 0, 0), \\ u_1 &= (-2, -3m - 3, 1 - 2^{-m}), \\ u_0 &= (-2, -3m - 3, 0). \end{aligned}$$

4 Constructing a universal convex body

The family of translates constructed in Section 3 uses a convex polytope K_m that depends on m. In this section we construct a single convex body K that allows the formation of families of n translates of K, for arbitrarily large n, with a cubic number of holes. (Note that the *position* of the translates in the family will depend on n.)

The convex body K. The finite polygonal paths γ and η are replaced by infinite polygonal paths. We must also redefine the vertices $v_{j,k}$, but the rest of the construction remains largely the same as before. (Figure 9 illustrates the construction.)

Using $\zeta_s = \sum_{t=1}^{\infty} t^{-s}$ for $s \in \{2, 3\}$, we define vertices as follows:

$$
\begin{aligned}
w_{0,0} &= 0, \\
w_{j,0} &= w_{j-1,0} + \left(0, -j^{-2}, j^{-3}\right) && \text{for } j \in \{1, 2, \dots\}, \\
w_{\infty,0} &= \lim_{j\to\infty} w_{j,0} = (0, -\zeta_2, \zeta_3)\,, \\
w_{0,k} &= w_{0,k-1} + \left(k^{-2}, k^{-3}, 0\right) && \text{for } k \in \{1, 2, \dots\}, \\
w_{0,\infty} &= \lim_{k\to\infty} w_{0,k} = (\zeta_2, \zeta_3, 0)\,, \\
w_{j,k} &= w_{j,0} + w_{0,k} && \text{for } j, k \in \{1, 2, \dots, \infty\}, \\
v_{j,k} &= \tfrac{1}{(j+1)^3} w_{j,k} + \tfrac{(j+1)^3-1}{(j+1)^3} w_{j,k+1} && \text{for } j, k \in \{0, 1, \dots\}, \\
u_1 &= (\zeta_2, -2, \zeta_3), \\
u_0 &= (\zeta_2, -2, 0).
\end{aligned}
$$

We define the convex paths γ as $w_{0,0}w_{0,1}w_{0,2}\dots w_{0,\infty}$, and η as $w_{0,0}w_{1,0}w_{2,0}\dots w_{\infty,0}$. Again we let E_k denote the edge $w_{0,k}w_{0,k+1}$, for $k \in \{0, 1, \dots\}$. The vertex $v_{j,k}$ lies on the edge $w_{j,k}w_{j,k+1} = E_k + w_{j,0}$. For $j \in \{1, 2, \dots\}$, we set $\gamma_j = w_{j,0} + \gamma$, and define the convex path ξ_j as $w_{j,0}v_{j,0}v_{j,1}v_{j,2}v_{j,3}\dots w_{j,\infty}$; note that $\xi_0 = \gamma$. The path ξ_∞ is equal to $\gamma_\infty = \gamma + w_{\infty,0}$; note that $\lim_{j\to\infty} v_{j,k} = w_{\infty,k+1}$.

The front part of K is the convex hull of the paths ξ_j:

$$\textsc{Front} = \operatorname{conv}(\xi_0 \cup \xi_1 \cup \xi_2 \cup \dots \cup \xi_\infty).$$

The back part Back of K is the Minkowski sum of u_1u_0 and $-\gamma$. By construction, it is the union of the rectangles $u_1u_0 - E_k$, for $k \in \{0, 1, 2, \dots\}$. The top and bottom edges of each rectangle are $u_1 - E_k$ and $u_0 - E_k$.

Finally, we define $K = \operatorname{conv}(\textsc{Front} \cup \textsc{Back})$, concluding the description of our convex body K. This body has the following property.

▶ **Lemma 5.** *The polygonal path ξ_j lies entirely on the front boundary of K.*

Proof. We will show that any point in ξ_j lies outside the convex hull of the regions $\xi_{j'} + (0, \mathbb{R}^-, 0)$, for $j' \neq j$. Fix $j \in \mathbb{N}$, and let Γ be the convex hull of the regions $\gamma_{j'} + (0, \mathbb{R}^-, 0)$, for $j' \neq j$. Since the path $\xi_{j'}$ lies in the convex region $\gamma_{j'} + (0, \mathbb{R}^-, 0)$, it suffices to show that ξ_j lies outside Γ. Since the γ_j are just translates of γ, the body Γ is easy to describe. In particular, between the (x, y)-parallel planes containing γ_{j-1} and γ_{j+1}, its front boundary is formed by rectangles that are the convex hull of $E_k + w_{j-1,0}$ and $E_k + w_{j+1,0}$. Let Π be the (x, y)-parallel plane containing ξ_j and γ_j. The front boundary of $\Gamma \cap \Pi$ is again a translate of γ. We first compute this translate of γ, by finding the intersection point p of the segment $w_{j-1,0}w_{j+1,0}$ with Π.

Let $p' = p - w_{j-1,0}$. This makes p' the point at height $1/j^3$ on the line through the origin and the point $w_{j+1,0} - w_{j-1,0}$. Since

$$w_{j+1,0} - w_{j-1,0} = \left(0, -\frac{1}{j^2} - \frac{1}{(j+1)^2}, \frac{1}{j^3} + \frac{1}{(j+1)^3}\right) = \left(0, \frac{-j^2 - (j+1)^2}{j^2(j+1)^2}, \frac{j^3 + (j+1)^3}{j^3(j+1)^3}\right),$$

this gives

$$p' = \left(0, \frac{-(j+1)(j^2 + (j+1)^2)}{j^2(j^3 + (j+1)^3)}, \frac{1}{j^3}\right).$$

Since $p = p' + w_{j-1,0}$ and $w_{j,0} - w_{j-1,0} = (0, -j^{-2}, j^{-3})$, we have

$$p = w_{j,0} - \Big(0,\ \frac{1}{j^3 + (j+1)^3},\ 0\Big).$$

We have just computed the front boundary $\gamma + p$ of $\Gamma \cap \Pi$, and we want to show that the path ξ_j lies farther in front of this boundary. Since $\gamma + p$ and ξ_j are convex paths, it suffices to show that any vertex of $\gamma + p$ lies behind ξ_j. These vertices are the points $w_{0,k} + p$, for $k \in \{0, 1, 2, \dots\}$. That is, we will show $w_{0,k} + p \in \xi_j + (0, \mathbb{R}^-, 0)$.

We fix some $k \in \{1, 2, \dots\}$ and consider $w_{0,k} + p$. The line parallel to the y-axis through $w_{0,k}+p$ intersects the edge $v_{j,k-1}v_{j,k}$ of ξ_j in a point q. We need to show that $q^y \geq (w_{0,k}+p)^y$ (here and in the following, we use superscripts x, y, z to denote the coordinates of a point).

It will be convenient to translate our coordinate system such that $w_{j,k}$ is the origin. This means that Π is the plane $z = 0$. Letting $J = (j+1)^3$, we have:

$$\begin{aligned} w_{0,k} + p - w_{j,k} &= w_{0,k} + p - (w_{j,0} + w_{0,k}) = \Big(0, \frac{-1}{j^3 + J}, 0\Big), \\ v_{j,k} - w_{j,k} &= \tfrac{1-J}{J} w_{j,k} + \tfrac{J-1}{J} w_{j,k-1} = \tfrac{J-1}{J}\big[w_{j,k+1} - w_{j,k}\big], \\ v_{j,k-1} - w_{j,k} &= \tfrac{1}{J} w_{j,k-1} - \tfrac{1}{J} w_{j,k} = \tfrac{1}{J}\big[w_{j,k-1} - w_{j,k}\big]. \end{aligned}$$

Now parameterize the segment $v_{j,k-1}v_{j,k}$ as $E(s)$, for $0 \leq s \leq 1$, using our new coordinate system:

$$E(s) = (1-s)\frac{J-1}{J}\big[w_{j,k+1} - w_{j,k}\big] + \frac{s}{J}\big[w_{j,k-1} - w_{j,k}\big].$$

The x- and y-coordinates of the point $E(s)$ are

$$E(s)^x = (1-s)\frac{J-1}{J}\frac{1}{(k+1)^2} - \frac{s}{J}\frac{1}{k^2}, \qquad E(s)^y = (1-s)\frac{J-1}{J}\frac{1}{(k+1)^3} - \frac{s}{J}\frac{1}{k^3}.$$

We have $q - w_{j,k} = E(t)$, for the $t \in [0,1]$ where $E(t)^x = 0$. This condition is equivalent to $(1-t)(J-1)k^2 = t(k+1)^2$, and therefore

$$\begin{aligned} t &= \frac{(J-1)k^2}{(k+1)^2 + (J-1)k^2}, \\ E(t)^y &= \frac{1-J}{J}\frac{1}{k(k+1)((k+1)^2 + (J-1)k^2)} \geq \frac{1-J}{J}\frac{1}{2(J+3)}, \end{aligned}$$

where we used $k(k+1) \geq 2$, $k^2 \geq 1$, $(k+1)^2 - k^2 \geq 3$, and the fact that $1 - J \leq 0$. We further have

$$(w_{0,k} + p - w_{j,k})^y = \frac{-1}{j^3 + J} < \frac{-1}{2J} < \frac{1-J}{2J(J+3)} \leq E(t)^y = (q - w_{j,k})^y,$$

and therefore $(w_{0,k} + p)^y < q^y$. Thus, $w_{0,k} + p \in \xi_j + (0, \mathbb{R}^-, 0)$, so ξ_j is in front of Γ and as such is on the front boundary of K. ◀

The translates. We now pick a number $m \in \mathbb{N}$ and construct a family $\mathcal{F}$ of $3m$ translates of K such that their union will have a cubic number of holes. This construction is identical to the construction in Section 3:

First, for $k \in [m]$, we define a translate C_k such that the edge $u_1 - E_k$ of C_k coincides with the edge E_k of K, that is

$$C_k = K + c_k, \quad \text{where} \quad c_k = w_{0,k} - (u_1 - w_{0,k+1}).$$

Again we denote the vertical facet formed by $u_1 - E_k$ and $u_0 - E_k$ of C_k as R_k.

Next, for $j \in [m]$, we define a translate B_j of K as follows:

$$B_j = K + b_j, \quad \text{where} \quad b_j = -w_{j,0}.$$

In other words, the path ξ_j of B_j lies in the (x, y)-plane, the vertex $v_{j,k}$ of B_j lies on E_k.

For the third group of translates we need to determine a sufficiently small $\varepsilon > 0$. First, observe that the points $w_{j,k}$, for $j, k \in [m]$, do not lie on K. Let $\varepsilon_1 > 0$ be smaller than the distance of $w_{j,k}$ to K, for all $j, k \in [m]$. Second, consider the m^2 points $v_{j,k} + b_j$ for $j, k \in [m]$ on the path γ. Let ε_2 be the shortest distance between any two of these points.

Consider now the segment $w_{j,k}w_{j,k+1}$, for some $j, k \in [m]$. It touches K in the point $v_{j,k}$, the rest of the segment lies entirely outside K. This implies that there is an $\varepsilon_{j,k} > 0$ such that any line parallel to $w_{j,k}w_{j,k+1}$ at distance less than $\varepsilon_{j,k}$ intersects K only within a neighborhood of $v_{j,k}$ of radius $\varepsilon_2/3$.

We choose $\varepsilon < \varepsilon_1$ and $\varepsilon < \varepsilon_{j,k}$, for all $j, k \in [m]$. With this choice of $\varepsilon > 0$, we can finally define, for $i \in [m]$:

$$A_i = K + a_i, \quad \text{where } a_i = \left(0, 0, -\tfrac{i}{m}\varepsilon\right).$$

Our family $\mathcal{F}$ is

$$\mathcal{F} = \{A_1, \ldots, A_m, B_1, \ldots, B_m, C_1, \ldots, C_m\}.$$

The nerve. To every family $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ of n sets is associated a collection of subfamilies $\mathcal{N}(\mathcal{X})$, called the *nerve* of $\mathcal{X}$, defined as follows:

$$\mathcal{N}(\mathcal{X}) = \left\{ \mathcal{Y} \subseteq \mathcal{X} : \bigcap_{X \in \mathcal{Y}} X \neq \emptyset \right\}.$$

In a sense, the nerve is a natural generalization of the *intersection graph*. In Section 5, we will count the number of holes in the union of $\mathcal{F}$ by computing the rank of certain matrices defined in terms of its nerve. We now give an explicit description of the nerve of the family $\mathcal{F}$. Consider the following subfamilies of $\mathcal{F}$:

$$\begin{aligned}
\Delta_1 &= \{A_1, \ldots, A_m, B_1, \ldots, B_m\}, \\
\Delta_2 &= \{B_1, \ldots, B_m, C_1, \ldots, C_m\}, \\
\Delta_{i,k} &= \{A_i, C_k, C_{k+1}\} && \text{for } (i,k) \in [m] \times [m-1], \\
\Delta_{i,j,k} &= \{A_i, B_j, C_k\} && \text{for } (i,j,k) \in [m]^3.
\end{aligned}$$

▶ **Lemma 6.** *The set of inclusion-maximal subfamilies in $\mathcal{N}(\mathcal{F})$ is*

$$\mathcal{M} = \{\Delta_1, \Delta_2\} \cup \{\Delta_{i,k} : (i,k) \in [m] \times [m-1]\} \cup \{\Delta_{i,j,k} : (i,j,k) \in [m]^3\}.$$

Proof. To check that the subfamilies in $\mathcal{M}$ are in $\mathcal{N}(\mathcal{F})$, we find a point in the intersection of each of them. Specifically, we argue that

$$\begin{aligned}
w_{0,k+1} + a_i &\in A_i \cap C_k \cap C_{k+1}, \\
v_{j,k} + a_i + b_j &\in A_i \cap B_j \cap C_k, \\
(0, -1, 0) &\in \bigcap \{A_1, \ldots, A_m, B_1, \ldots, B_m\}, \\
(\zeta_2, 2, -1) &\in \bigcap \{B_1, \ldots, B_m, C_1, \ldots, C_m\},
\end{aligned}$$

see the full version [3, Appendix A] Now, let σ be a maximal subfamily in $\mathcal{N}(\mathcal{F})$. If σ does not contain any C_k then $\sigma \subseteq \Delta_1$ and by maximality $\sigma = \Delta_1$. Similarly, if σ does not contain any A_i then $\sigma \subseteq \Delta_2$ and by maximality $\sigma = \Delta_2$.

We can therefore assume that $A_i, C_k \in \sigma$. By definition of K and $\mathcal{F}$ we have $A_i \cap C_k = E_k + a_i$. Since $A_i \cap C_k$ and $A_{i'} \cap C_k$ are parallel segments for $i' \neq i$, σ cannot contain $A_{i'}$.

Assume now that σ contains no B_j. The segments E_k and $E_{k'}$ intersect if and only if k and k' differ by one. It follows that σ is either $\Delta_{i,k}$ or $\Delta_{i,k-1}$.

In the final case, σ contains some B_j, for $j \in [m]$. The segment $E_k + a_i - b_j$ is parallel to $w_{j,k}w_{j,k+1}$ at distance at most $\varepsilon < \varepsilon_{j,k}$, and so it intersects K only in a neighborhood of $v_{j,k}$ of radius at most $\varepsilon_2/3$. It follows that $E_k + a_i$ intersects $B_j = K + b_j$ only in a neighborhood of the same radius around the point $v_{j,k} + b_j$. But, the shortest distance between these points is ε_2, and so these neighborhoods are disjoint. It follows that σ contains no other $B_{j'}$, for $j' \neq j$.

Since the point $w_{0,k} + a_i$ lies at distance at most ε from $w_{0,k}$, but the point $w_{j,k} = w_{0,k} - b_j$ has distance larger than $\varepsilon_1 > \varepsilon$ from K, we have $A_i \cap C_{k-1} \cap C_k = w_{0,k} + a_i \notin B_j$, and so $C_{k-1} \notin \sigma$. For the same reason $C_{k+1} \notin \sigma$. It follows that $\sigma = \{A_i, C_k, B_j\} = \Delta_{i,j,k}$. ◀

5 Counting holes in the union via the nerve

In this section we use homology to count the number of holes in a union of convex objects and prove Theorem 1; we first illustrate our arguments by giving a new proof of Kovalev's upper bound [13]. We start by recalling some standard topological machinery.

Homology and Betti numbers. An *abstract simplicial complex* Δ with vertex set V is a set of subsets of V closed under taking subsets: if $\sigma \in \Delta$ and $\tau \subseteq \sigma$ then $\tau \in \Delta$. An element $\sigma \in \Delta$ is called a *simplex*; the *dimension* of a simplex is its cardinality minus 1, so singletons are simplices of dimension 0, pairs are simplices of dimension 1, etc. A simplex of dimension i is called an *i-simplex* for short. The *vertices* of a simplex $\sigma \in \Delta$ are the singletons contained in σ. Note that the nerve of a family of convex sets is an abstract simplicial complex.

Let Δ be an abstract simplicial complex on a *totally ordered* vertex set V. The ith real chain space of Δ, denoted $\mathcal{C}_i(\Delta)$, is the real vector space spanned[2] by the i-simplices of Δ. For $i \in \mathbb{N}$, the *ith boundary map* $\partial_i : \mathcal{C}_i(\Delta) \to \mathcal{C}_{i-1}(\Delta)$ is the linear map defined on a basis of $\mathcal{C}_i(\Delta)$ as follows. For any i-simplex $\sigma = \{v_0, v_1, \ldots, v_i\} \in \Delta$ with $v_0 < v_1 < \ldots < v_i$,

$$\partial_i(\sigma) = \sum_{j=0}^{i} (-1)^j (\sigma \setminus \{v_j\}).$$

That is, ∂_i maps every i-dimensional simplex to an element of $\mathcal{C}_{i-1}(\Delta)$, namely an alternating sum of its facets. Observe that $\partial_i \circ \partial_{i+1} = 0$, so that $\operatorname{im} \partial_{i+1} \subseteq \ker \partial_i$. The ith *simplicial homology group* $H_i(\Delta, \mathbb{R})$ of Δ is defined as the quotient $\ker \partial_i / \operatorname{im} \partial_{i+1}$ and the ith *Betti number* $\beta_i(\Delta)$ of Δ is the dimension of $H_i(\Delta, \mathbb{R})$, hence:

$$\beta_i(S) = \dim \ker \partial_i - \operatorname{rank} \partial_{i+1}. \tag{1}$$

If X is a subset of $\mathbb{R}^d$, one can define, in a similar but more technical way, the *singular homology groups* of X and its *Betti numbers*. We do not recall those definitions (the interested reader is referred to [10, 15]) but emphasize two facts that will be useful:

[2] In other words, the i-dimensional simplices of Δ form a basis of the vector space $\mathcal{C}_i(\Delta)$, which consists of formal sums of i-simplices, each simplex being assigned a real coefficient.

(i) $\beta_0(X)$ is the number of connected components of X, assuming X admits a cell decomposition.

(ii) If $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ is a family of convex objects in $\mathbb{R}^d$ and $U = \bigcup_{i=1}^n X_i$ then $H_i(U, \mathbb{R}) \simeq H_i(\mathcal{N}(\mathcal{X}))$; as a consequence, U and $\mathcal{N}(\mathcal{X})$ have the same Betti numbers. This follows from the classical *Nerve Theorem* of Borsuk [6].

Counting holes. We can now relate the number of holes in the union of a family of convex objects to one particular Betti number of its nerve.

▶ **Lemma 7.** *If $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ is a family of compact convex objects in $\mathbb{R}^d$ then the number of holes of $U = \bigcup_{i=1}^n X_i$ is $\beta_{d-1}(\mathcal{N}(\mathcal{X})) + 1$.*

Proof. The number of holes of U is, by definition, the number of connected components of $\mathbb{R}^d \setminus U$, which is $\beta_0(\mathbb{R}^d \setminus U)$. Assume $d > 1$. For any compact locally contractible subset $T \subseteq \mathbb{S}^d$, Alexander duality gives $\beta_{d-1}(T) = \beta_0(\mathbb{S}^d \setminus T) - 1$. Identifying the d-sphere with the one-point compactification of d-space, $\mathbb{S}^d \simeq \mathbb{R}^d \cup \{\infty\}$, we have that

$$\beta_0(\mathbb{R}^d \setminus U) = \beta_0(\mathbb{S}^d \setminus U) = \beta_{d-1}(U) + 1,$$

and by the Nerve Theorem, $\beta_{d-1}(U) = \beta_{d-1}(\mathcal{N}(\mathcal{X}))$. ◀

As an illustration let us see how a version of the upper bound of Kovalev [13] for compact convex objects immediately follows from Lemma 7:

▶ **Corollary 8.** *The number of holes in the union of n compact convex objects in $\mathbb{R}^d$ is at most $\binom{n}{d} + 1$.*

Proof. Let $\mathcal{X}$ be a family of n compact convex objects in $\mathbb{R}^d$. By Lemma 7, the number of holes of the union of the members of $\mathcal{X}$ is $\beta_{d-1}(\mathcal{N}(\mathcal{X})) + 1$. Let ∂_i denote the ith boundary operator of $\mathcal{N}(\mathcal{X})$. By definition, β_{d-1} is the dimension of the quotient of the vector space $\ker \partial_{d-1}$ by the vector space $\operatorname{im} \partial_d$. Now, $\ker \partial_{d-1}$ is contained in the space $\mathcal{C}_{d-1}(\mathcal{N}(\mathcal{X}))$ spanned by the $(d-1)$-simplices of $\mathcal{N}(\mathcal{X})$; since $\mathcal{N}(\mathcal{X})$ has n vertices it has at most $\binom{n}{d}$ simplices of dimension $d-1$ and $\ker \partial_{d-1}$ therefore has dimension at most $\binom{n}{d}$. This dimension can only go down by taking the quotient by $\operatorname{im} \partial_d$, so $\beta_{d-1}(\mathcal{N}(\mathcal{X})) \leq \binom{n}{d}$. ◀

The number of holes in the union of $\mathcal{F}$. We now prove Theorem 1 by using Lemma 7.

Proof of Theorem 1. Kovalev [13] already established that any union of n convex objects in $\mathbb{R}^3$ has $O(n^3)$ holes. Hence this bound applies to families of translates. It remains to prove that this bound is tight by constructing a family whose union has $\Omega(n^3)$ holes. Let K denote the convex body, and let $\mathcal{F}$ denote the family of $n = 3m$ translates of K constructed above. Let $U = \bigcup_{X \in \mathcal{F}} X$.

Recall that the maximal simplices of $\mathcal{N}(\mathcal{F})$ are identified by Lemma 6. By Lemma 7, the number of holes of U is $\beta_2(\mathcal{N}(\mathcal{F})) + 1$ which equals, by Equation (1), $\dim \ker \partial_2 - \operatorname{rank} \partial_3 + 1$, where ∂_i denotes the ith boundary map of $\mathcal{N}(\mathcal{F})$. We compute $\beta_2(\mathcal{N}(\mathcal{F}))$ by computing explicitly a basis for $\ker \partial_2$ and a basis for $\operatorname{im} \partial_3$.

To compute a basis of $\operatorname{im} \partial_3$, let $\mathcal{S}$ denote the set of 3-simplices of $\mathcal{N}(\mathcal{F})$ containing B_1 and let $\mathcal{T}$ stand for the set of images of the simplices of $\mathcal{S}$ under ∂_3:

$$\mathcal{S} = \{\sigma : |\sigma| = 4 \text{ and } B_1 \in \sigma\} \quad \text{and} \quad \mathcal{T} = \{\partial_3 \sigma : \sigma \in \mathcal{S}\}.$$

Observe that $\mathcal{T}$ is a linearly independent family. Indeed, for any $\sigma \in \mathcal{S}$, the 2-simplex $\sigma \setminus \{B_1\}$ has non-zero coefficient in $\partial_3(\sigma)$ but has zero coefficient in $\partial_3(\tau)$ for every $\tau \in \mathcal{S} \setminus \{\sigma\}$. To

see that $\mathcal{T}$ spans $\operatorname{im}\partial_3$, let σ be a 3-simplex of $\mathcal{N}(\mathcal{F})$. If $B_1 \in \sigma$ then $\sigma \in \mathcal{A}$ and so $\partial_3(\sigma) \in \mathcal{T}$. If $B_1 \notin \sigma$, since $\partial_3 \circ \partial_4 = 0$ we have

$$\partial_3 \circ \partial_4(\sigma \cup \{B_1\}) = \lambda \partial_3(\sigma) + \sum_{X \in \sigma} \lambda_v \partial_3(\sigma \cup \{B_1\} \setminus \{X\}) = 0$$

where λ and the λ_v are in $\{\pm 1\}$. This implies that $\partial_3(\sigma)$ is a sum of $\pm\partial_3(\tau)$ with $\tau \in \mathcal{S}$, and thus lies in the span of $\mathcal{T}$. Therefore, as claimed, $\mathcal{T}$ is a basis of $\operatorname{im}\partial_3$.

Now, let $\mathcal{S}'$ denote the set of 2-simplices in $\mathcal{F}$ that contain B_1 and let $\mathcal{T}' = \{\partial_2\sigma : \sigma \in \mathcal{S}'\}$. The same arguments yield that $\mathcal{T}'$ is a basis of $\operatorname{im}\partial_2$. Also let $\mathcal{S}''$ denote the set of all 2-simplices contained in $\mathcal{F}$.

We can finally compute $\beta_2(\mathcal{N}(\mathcal{F}))$ using the rank-nullity theorem,

$$\beta_2(\mathcal{N}(\mathcal{F})) = \operatorname{nullity}\partial_2 - \operatorname{rank}\partial_3 = \dim \mathcal{C}_2(\mathcal{N}(\mathcal{F})) - \operatorname{rank}\partial_2 - \operatorname{rank}\partial_3 = |\mathcal{S}''| - |\mathcal{T}'| - |\mathcal{T}|.$$

Counting all quadruples in Δ_1 or Δ_2 that contain B_1 (taking care that some quadruples appear both in Δ_1 and Δ_2) we have

$$|\mathcal{T}| = |\mathcal{S}| = 2\tbinom{2m-1}{3} - \tbinom{m-1}{3}.$$

Next, counting all triples in $\mathcal{F}$ that contain B_1 we have

$$|\mathcal{T}'| = |\mathcal{S}'| = \tbinom{3m-1}{2}.$$

Then, counting all triples contained in $\mathcal{N}(\mathcal{F})$, which are the triples in $\Delta_{i,j,k}$ plus $\Delta_{j,k}$ plus the triples in Δ_1 or Δ_2 (accounting for triples belonging to both Δ_1 and Δ_2), we have

$$|\mathcal{S}''| = m^3 + m(m-1) + 2\tbinom{2m}{3} - \tbinom{m}{3}.$$

Finally, using $\binom{a}{b} - \binom{a-1}{b} = \binom{a-1}{b-1}$, we have

$$\begin{aligned}
\beta_2(\mathcal{N}(\mathcal{F})) &= |\mathcal{S}''| - |\mathcal{T}| - |\mathcal{T}'| \\
&= m + m(m-1) + 2\tbinom{2m}{3} - \tbinom{m}{3} - 2\tbinom{2m-1}{3} + \tbinom{m-1}{3} - \tbinom{3m-1}{2} \\
&= m^3 + m(m-1) + 2\tbinom{2m-1}{2} - \tbinom{m-1}{2} - \tbinom{3m-1}{2} \\
&= m^3 - m.
\end{aligned}$$

So $\mathcal{F}$ is a family of $3m$ translates of a convex body, and the union has $m^3 - m + 1$ holes. ◀

References

1. Pankaj K. Agarwal, Sariel Har-Peled, Haim Kaplan, and Micha Sharir. Union of random Minkowski sums and network vulnerability analysis. *Discrete Comput. Geom.*, 52(3):551–582, 2014.
2. Pankaj K. Agarwal, János Pach, and Micha Sharir. State of the union (of geometric objects). In *Proc. Joint Summer Research Conf. on Discrete and Computational Geometry: 20 Years Later*, volume 452 of *Contemp. Math.*, pages 9–48. AMS, 2008.
3. Boris Aronov, Otfried Cheong, Michael G. Dobbins, and Xavier Goaoc. The number of holes in the union of translates of a convex set in three dimensions. To appear in Discrete Comput. Geom., 2015. URL: `http://arxiv.org/abs/1502.01779`.
4. Boris Aronov and Micha Sharir. On translational motion planning of a convex polyhedron in 3-space. *SIAM J. Comput.*, 26(6):1785–1803, 1997.
5. Franz Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

6 Karol Borsuk. On the imbedding of systems of compacta in simplicial complexes. *Fundamenta Mathematicae*, 35:217–234, 1948.

7 Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete Comput. Geom.*, 37(1):43–58, 2007.

8 Alon Efrat and Micha Sharir. On the complexity of the union of fat convex objects in the plane. *Discrete Comput. Geom.*, 23(2):171–189, 2000.

9 Steven Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 23, pages 513–528. CRC Press LLC, Boca Raton, FL, 2 edition, 2004.

10 Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, UK, 2002.

11 Christian Icking, Rolf Klein, Ngoc-Minh Lé, and Lihong Ma. Convex distance functions in 3-space are different. *Fund. Inform.*, 22:331–352, 1995. doi:10.3233/FI-1995-2242.

12 Klara Kedem, Ron Livne, János Pach, and Micha Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal ostacles. *Discrete Comput. Geom.*, 1(1):59–71, 1986.

13 Mikhail D. Kovalev. Svoistvo vypuklykh mnozhestv i ego prilozhenie (A property of convex sets and its application). *Mat. Zametki*, 44:89–99, 1988. In Russian.

14 Joseph S. B. Mitchell and Joseph O'Rourke. Computational geometry column 42. *Int. J. Comput. Geom. Ap.*, 11(05):573–582, 2001.

15 James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, Menlo Park, CA, 1984.

16 János Pach and Gábor Tardos. On the boundary complexity of the union of fat triangles. *SIAM J. Comput.*, 31(6):1745–1760, 2002.

17 Micha Sharir. Algorithmic motion planning. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 47, pages 1037–1064. CRC Press LLC, Boca Raton, FL, 2 edition, 2004.

18 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, NY, USA, 2010.

19 Boaz Tagansky. *The Complexity of Substructures in Arrangments of Surfaces*. PhD thesis, Tel Aviv University, 1996.

On the Combinatorial Complexity of Approximating Polytopes

Sunil Arya[*1], **Guilherme D. da Fonseca**[2], **and David M. Mount**[†3]

1 **Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Kowloon, Hong Kong**
arya@cse.ust.hk

2 **Université d'Auvergne and LIMOS, Clermont-Ferrand, France**
guilherme.dias_da_fonseca@udamail.fr

3 **Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, USA**
mount@cs.umd.edu

Abstract

Approximating convex bodies succinctly by convex polytopes is a fundamental problem in discrete geometry. A convex body K of diameter $\mathrm{diam}(K)$ is given in Euclidean d-dimensional space, where d is a constant. Given an error parameter $\varepsilon > 0$, the objective is to determine a polytope of minimum combinatorial complexity whose Hausdorff distance from K is at most $\varepsilon \cdot \mathrm{diam}(K)$. By combinatorial complexity we mean the total number of faces of all dimensions of the polytope. A well-known result by Dudley implies that $O(1/\varepsilon^{(d-1)/2})$ facets suffice, and a dual result by Bronshteyn and Ivanov similarly bounds the number of vertices, but neither result bounds the total combinatorial complexity. We show that there exists an approximating polytope whose total combinatorial complexity is $\widetilde{O}(1/\varepsilon^{(d-1)/2})$, where $\widetilde{O}$ conceals a polylogarithmic factor in $1/\varepsilon$. This is a significant improvement upon the best known bound, which is roughly $O(1/\varepsilon^{d-2})$.

Our result is based on a novel combination of both new and old ideas. First, we employ Macbeath regions, a classical structure from the theory of convexity. The construction of our approximating polytope employs a new stratified placement of these regions. Second, in order to analyze the combinatorial complexity of the approximating polytope, we present a tight analysis of a width-based variant of Bárány and Larman's *economical cap covering*, which may be of independent interest. Finally, we use a deterministic variation of the witness-collector technique (developed recently by Devillers et al.) in the context of our stratified construction.

1998 ACM Subject Classification F.2.2 Geometrical problems and computations

Keywords and phrases Polytope approximation, Convex polytopes, Macbeath regions

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.11

1 Introduction

Approximating general convex bodies by convex polytopes is a fundamental geometric problem. It has been extensively studied in the literature under various formulations. (See Bronstein [13] for a recent survey.) Consider a convex body (that is, a closed, convex set of bounded diameter) K in Euclidean d-dimensional space. At issue is determining the simplest polytope P that approximates K.

* Research supported by the Research Grants Council of Hong Kong, China under project number 610012.
† Research supported by NSF grant CCF-1117259.

© Sunil Arya, Guilherme D. da Fonseca, and David M. Mount;
licensed under Creative Commons License CC-BY

There are various ways to define the notions of "simplest" and "approximates." Our notion of approximation will be based on the *Hausdorff metric*, that is, the maximum distance between a point in the boundary of P or K and the boundary of the other body. Normally, approximation error is defined relative to K's diameter. It will simplify matters to assume that K has been uniformly scaled to unit diameter. For a given error $\varepsilon > 0$, we say that a polytope P is an *ε-approximating polytope* to K if the Hausdorff distance between K and P is at most ε. The simplicity of an approximating polytope P will be measured in terms of its *combinatorial complexity*, that is, the total number of k-faces, for $0 \leq k \leq d-1$. For the purposes of stating asymptotic bounds, we assume that the dimension d is a constant.

The bounds given in the literature for convex approximation are of two common types [13]. In both cases, the bounds hold for all $\varepsilon \leq \varepsilon_0$, for some $\varepsilon_0 > 0$. In *nonuniform bounds*, the value of ε_0 depends on K (for example, on K's maximum curvature). Such bounds are often stated as holding "in the limit" as ε approaches zero, or equivalently as the combinatorial complexity of the approximating polytope approaches infinity. Examples include bounds by Gruber [18], Clarkson [14], and others [10, 23]. Our interest is in *uniform bounds*, where the value of ε_0 is independent of K. Examples include the results of Dudley [16] and Bronshteyn and Ivanov [12]. Such bounds hold without any assumptions on K.

Dudley showed that, for $\varepsilon \leq 1$, any convex body K of unit diameter can be ε-approximated by a convex polytope P with $O(1/\varepsilon^{(d-1)/2})$ facets. This bound is known to be optimal in the worst case and is achieved when K is a Euclidean ball (see, e.g., [13]). Alternatively, Bronshteyn and Ivanov showed the same bound holds for the number of vertices, which is also the best possible. No convex polytope approximation is known that attains both bounds simultaneously.[1]

Establishing good uniform bounds on the combinatorial complexity of convex polytope approximations is a major open problem. The Upper-Bound Theorem [22] implies that a polytope with n vertices (resp., facets) has total combinatorial complexity $O(n^{\lfloor d/2 \rfloor})$. Applying this to the results of either Dudley or Bronshteyn and Ivanov directly yields a bound of $O(1/\varepsilon^{(d^2-d)/4})$ on the combinatorial complexity of an ε-approximating polytope. Better uniform bounds without d^2 in the exponent are known, however. Consider a uniform grid Ψ of points with spacing $\Theta(\varepsilon)$, and let P denote the convex hull of $\Psi \cap K$. It is easy to see that P is an ε-approximating polytope for K. The combinatorial complexity of any lattice polytope[2] is $O(V^{(d-1)/(d+1)})$, where V is the volume of the polytope [2, 8]. This implies that P has combinatorial complexity $O(1/\varepsilon^{d(d-1)/(d+1)}) \approx O(1/\varepsilon^{d-2})$. While this is significantly better than the bound provided by the Upper-Bound Theorem, it is still much larger than the lower bound of $\Omega(1/\varepsilon^{(d-1)/2})$.

We show that this gap can be dramatically reduced. In particular, we establish an upper bound on the combinatorial complexity of convex approximation that is optimal to within a polylogarithmic factor in $1/\varepsilon$.

▶ **Theorem 1.1.** *Let $K \subset \mathbb{R}^d$ be a convex body of unit diameter, where d is a fixed constant. For all sufficiently small positive ε (independent of K) there exists an ε-approximating convex polytope P to K of combinatorial complexity $O(1/\widehat{\varepsilon}^{(d-1)/2})$, where $\widehat{\varepsilon} = \varepsilon/\log(1/\varepsilon)$.*

This is within a factor of $O(\log^{(d-1)/2}(1/\varepsilon))$ of the aforementioned lower bound. Our approach employs a classical structure from the theory of convexity, called *Macbeath regions* [21].

---

[1] Jeff Erickson noted that both bounds can be attained simultaneously but at the cost of sacrificing convexity [14].

[2] A *lattice polytope* is the convex hull of any set of points with integer coordinates.

Macbeath regions have found numerous uses in the theory of convex sets and the geometry of numbers (see Bárány [7] for an excellent survey). They have also been applied to a small but growing number of results in the field of computational geometry (see, e.g., [11, 5, 4, 3]). Our construction of the approximating polytope uses a new stratified placement of these regions. In order to analyze the combinatorial complexity of the approximating polytope, in Section 3 we present a tight analysis of a width-based variant of Bárány and Larman's economical cap covering. This result may be of independent interest. Finally, we use a deterministic variation of the witness-collector technique (developed recently by Devillers et al. [15]) in the context of our stratified construction.

The paper is organized as follows. In Section 2, we define concepts related to Macbeath regions and present some of their key properties. In Section 3, we prove the width-based economical cap covering lemma. The stratified placement of the Macbeath regions and the bound on the combinatorial complexity of approximating polytopes follow in Section 4.

2 Geometric Preliminaries

Recall that K is a convex body of unit diameter in $\mathbb{R}^d$. Let ∂K denote its boundary. Let O denote the origin of $\mathbb{R}^d$, and for $x \in \mathbb{R}^d$ and $r \geq 0$, let $B^r(x)$ denote the Euclidean ball of radius r centered at x. It will be convenient to first map K to a convenient form. We say that a convex body K is in *canonical form* if $B^{1/2d}(O) \subseteq K \subseteq B^{1/2}(O)$. A body in canonical form is $(1/d)$-fat and has diameter $\Theta(1)$. We will refer to point O as the *center* of K.

The following lemma shows that, up to constant factors, the problem of approximating an arbitrary convex body can be reduced to approximating a convex body in canonical form. The proof (which will be given in the full version) follows from a combination of John's Theorem [20] and Lemma 3.1 of Agarwal et al. [1].

▶ **Lemma 2.1.** *Let K be a convex body of unit diameter in $\mathbb{R}^d$. There exists a non-singular affine transformation T such that $T(K)$ is in canonical form and if P is any (ε/d)-approximating polytope to $T(K)$, then $T^{-1}(P)$ is an ε-approximating polytope to K.*

We assume henceforth that K is given in canonical form and that ε has been appropriately scaled. This scaling only affects the constant factors in our asymptotic bounds.

A *cap* C is defined to be the nonempty intersection of the convex body K with a halfspace H (see Fig. 1(a)). Let h denote the hyperplane bounding H. We define the *base* of C to be $h \cap K$. The *apex* of C is any point in the cap such that the supporting hyperplane of K at this point is parallel to h. The *width* of C is the distance between h and this supporting hyperplane. Given any cap C of width w and a real $\lambda \geq 0$, we define its λ-*expansion*, denoted C^λ, to be the cap of K cut by a hyperplane parallel to and at distance λw from this supporting hyperplane. (Note that $C^\lambda = K$, if λw exceeds the width of K along the defining direction.) An easy consequence of convexity is that, for $\lambda \geq 1$, C^λ is a subset of the region obtained by scaling C by a factor of λ about its apex. It follows that, for $\lambda \geq 1$, $\mathrm{vol}(C^\lambda) \leq \lambda^d \cdot \mathrm{vol}(C)$. For a given $\varepsilon > 0$, let $K(\varepsilon) \subset K$ denote the points of K within distance at most ε from ∂K (equivalently, the union of all ε-width caps).

Given a point $x \in K$ and real parameter $\lambda \geq 0$, the *Macbeath region* $M^\lambda(x)$ (also called an *M-region*) is defined as:

$$M^\lambda(x) \;=\; x + \lambda((K - x) \cap (x - K)).$$

It is easy to see that $M^1(x)$ is the intersection of K and the reflection of K around x (see Fig. 1(b)), and so $M^1(x)$ is centrally symmetric about x. $M^\lambda(x)$ is a scaled copy of $M^1(x)$

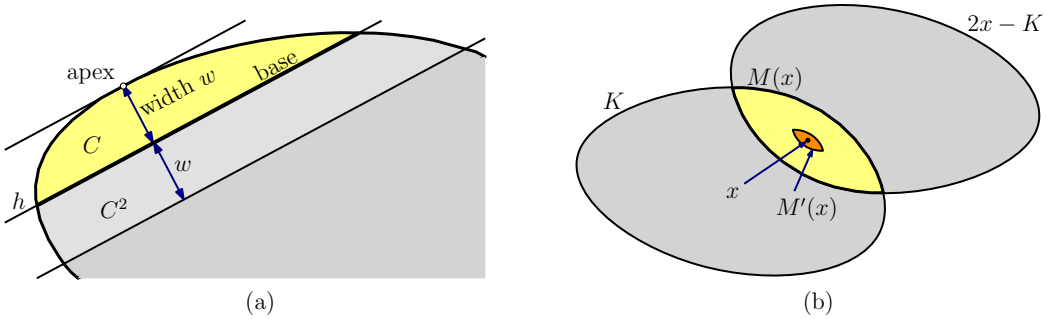


Figure 1 (a) Cap concepts and (b) Macbeath regions.

by the factor λ about x. We refer to x as the *center* of $M^\lambda(x)$ and to λ as its *scaling factor*. As a convenience, we define $M(x) = M^1(x)$ and $M'(x) = M^{1/5}(x)$.

We begin with two lemmas that encapsulate relevant properties of Macbeath regions. Both were proved originally by Ewald, Larman, and Rogers [17], but our statements follow the forms given by Brönnimann, Chazelle, and Pach [11]. (Lemmas 2.2 and 2.3 below are restatements of Lemmas 2.5 and 2.6 from [11], respectively.)

▶ **Lemma 2.2.** *Let K be a convex body. If $x, y \in K$ such that $M'(x) \cap M'(y) \neq \emptyset$, then $M'(y) \subseteq M(x)$.*

▶ **Lemma 2.3.** *Let $K \subset \mathbb{R}^d$ be a convex body in canonical form, and let $\Delta_0 = 1/(6d)$ be a constant. Let C be a cap of K of width at most Δ_0. Let x denote the centroid of the base of this cap. Then $C \subseteq M^{3d}(x)$.*

The following lemma is an immediate consequence of the definition of Macbeath region.

▶ **Lemma 2.4.** *Let K be a convex body and $\lambda > 0$. If x is a point in a cap C of K, then $M^\lambda(x) \cap K \subseteq C^{1+\lambda}$. Furthermore, if $\lambda \leq 1$, then $M^\lambda(x) \subseteq C^{1+\lambda}$.*

The next lemma is useful in situations when we know that a Macbeath region partially overlaps a cap of K. It allows us to conclude that a constant factor expansion of the cap will fully contain the Macbeath region.

▶ **Lemma 2.5.** *Let K be a convex body. Let C be a cap of K and x be a point in K such that $C \cap M'(x) \neq \emptyset$. Then $M'(x) \subseteq C^2$.*

Proof. Let y be any point in $C \cap M'(x)$. Since $M'(x) \cap M'(y) \neq \emptyset$ obviously holds, we can apply Lemma 2.2 to conclude that $M'(x) \subseteq M(y)$. By Lemma 2.4 (with $\lambda = 1$), $M(y) \subseteq C^2$. It follows that $M'(x) \subseteq C^2$. ◀

Next, we give two straightforward lemmas [3] dealing with scaling of centrally symmetric convex bodies. As Macbeath regions are centrally symmetric, these lemmas will be useful to us in conjunction with their standard properties. A proof of Lemma 2.6 appears in Bárány [6]. For any centrally symmetric convex body A, define A^λ to be the body obtained by scaling A by a factor of λ about its center.

[3] We have omitted a number of technical proofs from this version of the paper. An expanded version is available on `http://www.arxiv.org`.

▶ **Lemma 2.6.** *Let $\lambda \geq 1$. Let A and B be centrally symmetric convex bodies such that $A \subseteq B$. Then $A^\lambda \subseteq B^\lambda$.*

▶ **Lemma 2.7.** *Let $\lambda \geq 1$. Let A be a centrally symmetric convex body. Let A' be the body obtained by scaling A by a factor of λ about any point in A. Then $A' \subseteq A^{2\lambda-1}$.*

The following lemma is an easy consequence of Lemmas 2.3 and 2.7.

▶ **Lemma 2.8.** *Let $\lambda \geq 1$ and let K, C, and x be as defined in Lemma 2.3. Then $C^\lambda \subseteq M^{3d(2\lambda-1)}(x)$.*

Proof. By Lemma 2.3, $C \subseteq M^{3d}(x)$. Recall that C^λ is contained within the region obtained by scaling C by a factor of λ about its apex. Applying Lemma 2.7 (applied to $M^{3d}(x)$ and the apex point), it follows that $C^\lambda \subseteq M^{3d(2\lambda-1)}(x)$. ◀

It is well known that if two (1/5)-shrunken Macbeath regions have a nonempty intersection, then a constant factor expansion of one contains the other [17, 11]. We show that this holds for the associated caps.

▶ **Lemma 2.9.** *Let Δ_0 be the constant of Lemma 2.3 and let $\lambda \geq 1$ be any real. There exists a constant $\beta \geq 1$ such that the following holds. Let $K \subset \mathbb{R}^d$ be a convex body in canonical form. Let C_1 and C_2 be any two caps of K of width at most Δ_0. Let x_1 and x_2 denote the centroids of the bases of the caps C_1 and C_2, respectively. If $M'(x_1) \cap M'(x_2) \neq \emptyset$, then $C_1^\lambda \subseteq C_2^{\beta\lambda}$.*

Proof. By Lemma 2.8, $C_1^\lambda \subseteq M^\alpha(x_1)$, where $\alpha = 3d(2\lambda - 1)$. Since $M'(x_1) \cap M'(x_2) \neq \emptyset$, by Lemma 2.2, $M'(x_1) \subseteq M(x_2)$. By definition, $M'(x_1) = M^{1/5}(x_1)$ and so $M^\alpha(x_1) = (M'(x_1))^{5\alpha}$. Since $M'(x_1)$ and $M(x_2)$ are centrally symmetric bodies and $M'(x_1) \subseteq M(x_2)$, by Lemma 2.6, it follows that $(M'(x_1))^{5\alpha} \subseteq M^{5\alpha}(x_2)$. Putting it together, we obtain $C_1^\lambda \subseteq M^\alpha(x_1) = (M'(x_1))^{5\alpha} \subseteq M^{5\alpha}(x_2)$.

By Lemma 2.4, $M^{5\alpha}(x_2) \cap K \subseteq C_2^{1+5\alpha}$. Since $C_1^\lambda \subseteq M^{5\alpha}(x_2)$ and $C_1^\lambda \subseteq K$, we have $C_1^\lambda \subseteq M^{5\alpha}(x_2) \cap K \subseteq C_2^{1+5\alpha}$. Recalling that $\alpha = 3d(2\lambda - 1)$, we have $C_1^\lambda \subseteq C_2^{30d\lambda}$. This proves the lemma for constant $\beta = 30d$. ◀

3 Economical Cap Covering

In this section we present a tight analysis of a width-based variant of Bárány and Larman's economical cap covering [9]. The lemma applies to any convex body K that is fat and has constant diameter. The proof of this lemma follows from the ideas in [17, 9, 6]. Our principal contribution is an optimal bound of $O(1/\varepsilon^{(d-1)/2})$ on the number of bodies needed.

▶ **Lemma 3.1** (Width-based economical cap covering lemma)**.** *Let $\varepsilon > 0$ be a sufficiently small parameter. Let $K \subset \mathbb{R}^d$ be a convex body in canonical form. There exists a collection $\mathcal{R}$ of $k = O(1/\varepsilon^{(d-1)/2})$ disjoint centrally symmetric convex bodies $R_1, \ldots, R_k$ (see Fig. 2(a)) and associated caps $C_1, \ldots, C_k$ such that the following hold (for some constants β and λ, which depend only on d):*

1. *For each i, C_i is a cap of width $\beta\varepsilon$, and $R_i \subseteq C_i \subseteq R_i^\lambda$.*
2. *Let C be any cap of width ε. Then there is an i such that $R_i \subseteq C$ and $C_i^{1/\beta^2} \subseteq C \subseteq C_i$ (see Fig. 2(b)).*

The R_i's in this lemma are Macbeath regions with scaling factor 1/5. Since any cap of width ε is contained in some cap C_i, it follows that the C_i's together cover $K(\varepsilon)$. Further,

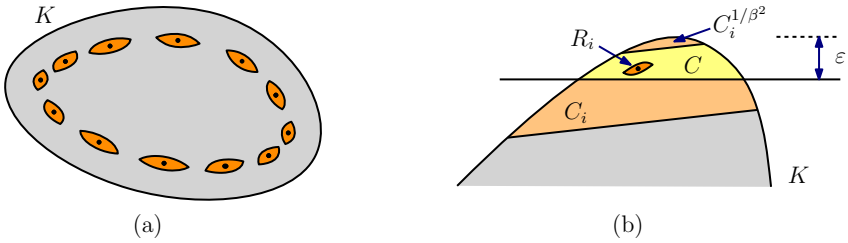


Figure 2 Illustrating Lemma 3.1.

from Property 1, we can see that the sum of the volume of the C_i's is no more than a constant times the volume of $K(\varepsilon)$. It is in this sense that the C_i's constitute an *economical* cap covering.

It is worth mentioning that Property 2 is stronger than similar properties given previously in the literature in the following sense. For any cap of width ε, we show not merely that it is contained within some cap C_i of the cover, but it is effectively "sandwiched" between two caps with parallel bases, each of width $\Theta(\varepsilon)$.

A key technical contribution of our paper is the following lemma. It will help us bound the number of bodies needed in the width-based cap covering lemma.

▶ **Lemma 3.2.** *Let $K \subset \mathbb{R}^d$ be a convex body in canonical form. Let $0 < \delta \leq \Delta_0$, where Δ_0 is the constant of Lemma 2.3. Let $\mathcal{C}$ be a set of caps, whose widths lie between δ and 2δ, such that the Macbeath regions $M'(x)$ centered at the centroids x of the bases of these caps are disjoint. Then $|\mathcal{C}| = O(1/\delta^{(d-1)/2})$.*

Our proof of Lemma 3.2 will require the following geometric observation, which is a straightforward extension of Dudley's convex approximation construction (see Lemma 4.4 of [16]). It is similar to other results based on Dudley's construction (including Lemma 3.6 of [1] and Lemma 23.12 of [19]).

▶ **Lemma 3.3.** *Let K be a convex body that lies within a unit sphere centered at the origin, and let $0 < \delta \leq 1$. Let x' and y' be two points of S. Let x and y be the points of ∂K that are closest to x' and y', respectively. Let h denote the supporting hyperplane at x orthogonal to the segment xx'. Let C denote the cap cut from K by a hyperplane parallel to and at distance δ from h. If $y \notin C$, then $\|x' - y'\| \geq \sqrt{\delta}$.*

We are now ready to present the proof of Lemma 3.2.

Proof. (of Lemma 3.2) Let A be the set of disjoint Macbeath regions $M'(x)$ described in the lemma. For each region $M'(x)$, let $C(x)$ denote the cap whose base centroid point generates $M'(x)$. We begin by pruning A to obtain a subset B, which to within constant factors has the same cardinality as A. We construct B incrementally as follows. Initially B is the empty set. In each step, from among the Macbeath regions that still remain in A, we choose a Macbeath region $M'(x)$ that has the smallest volume, and insert it into B. We then prune all the Macbeath regions from A that intersect the cap $C^4(x)$. We continue in this manner until A is exhausted.

We claim that in each step, we prune a constant number of Macbeath regions from A. Let $M'(x)$ denote the Macbeath region inserted into B in this step. If $M'(y)$ is a Macbeath region that is pruned in this step, then $M'(y)$ intersects the cap $C^4(x)$. It then follows from Lemma 2.5 that $M'(y) \subseteq C^8(x)$. Note that $\text{vol}(C^8(x)) \leq 8^d \text{vol}(C(x)) = O(\text{vol}(C(x)))$. Also, by Lemma 2.3, $C(x) \subseteq M^{3d}(x)$. It follows that $\text{vol}(M(x)) \geq \text{vol}(C(x))/(3d)^d = \Omega(\text{vol}(C(x)))$.

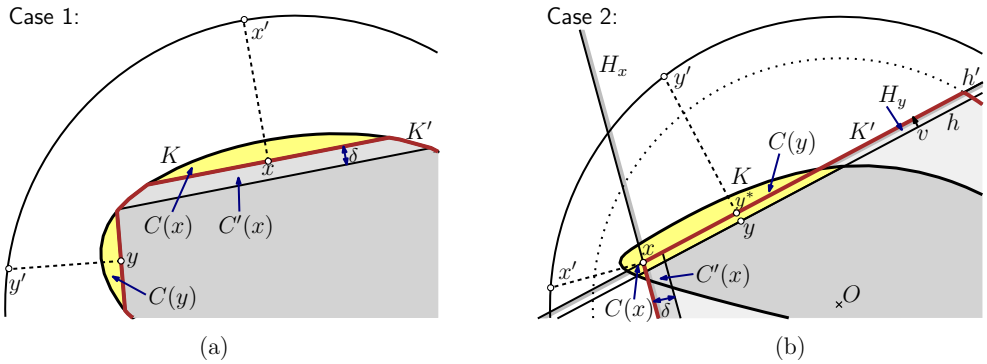


Figure 3 Cases arising in the proof of Lemma 3.2. (Figures not to scale.)

Recall that each Macbeath region pruned has volume greater than or equal to the volume of $M'(x)$. It follows that the volume of each Macbeath region pruned is $\Omega(\text{vol}(M(x))) = \Omega(\text{vol}(C(x)))$. Since the pruned Macbeath regions are disjoint and contained in a region of volume $O(\text{vol}(C(x)))$, a straightforward packing argument implies that the number of Macbeath regions pruned is $O(1)$.

The claim immediately implies that $|A| = O(|B|)$. In the remainder of the proof, we will show that $|B| = O(1/\delta^{(d-1)/2})$, which will complete the proof.

Let X denote the set of centers of the Macbeath regions of B, that is, $X = \{x : M'(x) \in B\}$. We map each point $x \in X$ to a point x' on the Dudley sphere such that xx' is normal to the base of the cap $C(x)$. We claim that the distance between any pair of the projected points x' on the Dudley sphere is at least $\sqrt{\delta}$. Note that this claim would imply the desired bound on $|B|$ and complete the proof.

To see this claim, consider any two Macbeath regions $M'(x)$ and $M'(y)$ in the set B. Without loss of generality, suppose that $M'(y)$ is inserted into B after $M'(x)$. By our construction, it follows that y is not contained in $C^4(x)$ (because otherwise $M'(y)$ would intersect $C^4(x)$ and would have been pruned after inserting $M'(x)$ into B). We now consider two cases, depending on whether or not x is contained in $C(y)$.

Case 1: $x \notin C(y)$. Consider the convex body K' that is the closure of $K \setminus (C(x) \cup C(y))$ (outlined in red in Fig. 3(a)). Note that x and y are on the boundary of the convex body K' and these are the points of $\partial K'$ that are closest to x' and y', respectively. Next, consider the cap of K' whose apex is x and width is δ. Call this cap $C'(x)$. Since the width of $C(x)$ is at least δ, and $y \notin C^4(x)$, it is easy to see that $y \notin C'(x)$. Applying Lemma 3.3 to the convex body K' and the points x', y', x, and y, it follows that $\|x'y'\| \geq \sqrt{\delta}$.

Case 2: $x \in C(y)$. Let h denote the hyperplane that forms the base of $C(y)$ (see Fig. 3(b)). Let h' denote the hyperplane parallel to h that passes through x. Let v denote the vector normal to h, whose magnitude is the distance between h and h'. Note that $h' = h + v$. Since $C(y)$ is a cap of width at most 2δ, the magnitude of the translation vector v is at most 2δ. Let $y^* = y + v$. Let H_y denote the halfspace bounded by h' that contains the origin. Let H_x denote the halfspace that contains the origin and whose boundary is the hyperplane forming the base of $C(x)$. Define the convex body K' as the intersection of H_x and H_y and a ball of unit radius centered at the origin. Note that x and y^* lie on the boundary of K' (since $\|Ox\| < 1$ and $\|Oy^*\| < 1$; $\|Ox\| < 1$ holds trivially since $x \in K$ and $K \subseteq B^{1/2}(O)$, and $\|Oy^*\| \leq \|Oy\| + \|yy^*\| \leq 1/2 + 2\delta \leq 1/2 + 2\Delta_0 < 1$).

Further, the points x and y^* are the points of $\partial K'$ that are closest to x' and y', respectively. Next, consider the cap of K' whose apex is x and width is δ and whose base is parallel to the base of $C(x)$. Call this cap $C'(x)$. Recall that $y \notin C^4(x)$, the width of $C(x)$ is at least δ, and the distance between y and y^* is at most 2δ. It follows that y^* is at distance bigger than $3\delta - 2\delta = \delta$ from the hyperplane passing through the base of $C(x)$. Since the distance between the hyperplanes passing through the bases of $C(x)$ and $C'(x)$, respectively, is δ, it follows that $y^* \notin C'(x)$. Applying Lemma 3.3 to the convex body K' and the points x', y', x, and y^*, it follows that the distance between x' and y' is at least $\sqrt{\delta}$. This establishes the above claim and completes the proof. ◀

The remainder of this section is devoted to proving Lemma 3.1. Assume that $\varepsilon \leq \Delta_0$, where Δ_0 is the constant of Lemma 2.3. Let $\beta = 30d$ be the constant of Lemma 2.9. Let $\mathcal{C}$ be a maximal set of caps, each of width ε/β, such that the $(1/5)$-scaled Macbeath regions centered at the centroids of the bases of these caps are disjoint. Let $A_1, \ldots, A_k$ denote the caps of $\mathcal{C}$. Let x_i denote the centroid of the base of cap A_i. With each cap A_i, we associate a convex body $R_i = M'(x_i)$ and a cap $C_i = A_i^{\beta^2}$. We will show that the convex bodies R_i and caps C_i satisfy the properties given in the lemma.

By Lemma 3.2, $|\mathcal{C}| = O(1/\varepsilon^{(d-1)/2})$, which implies the desired upper bound on k. Since C_i is a β^2-expansion of A_i, its width is $\beta\varepsilon$. To prove Property 1, it remains to show that $M'(x_i) \subseteq C_i \subseteq (M'(x_i))^\lambda$. By Lemma 2.4, $M'(x_i) \subseteq A_i^{6/5}$. Since $A_i^{6/5} \subseteq A_i^{\beta^2} = C_i$, we obtain $M'(x_i) \subseteq C_i$. Also, applying Lemma 2.8, we obtain

$$C_i = A_i^{\beta^2} \subseteq M^{3d(2\beta^2-1)}(x_i) = (M'(x_i))^{15d(2\beta^2-1)} \subseteq (M'(x_i))^\lambda,$$

where $\lambda = 30d\beta^2$. Thus, $M'(x_i) \subseteq C_i \subseteq (M'(x_i))^\lambda$.

To show Property 2, let C be any cap of width ε. Let x denote the centroid of the base of $C^{1/\beta}$. By maximality of $\mathcal{C}$, there must be a Macbeath region $M'(x_i)$ that has a nonempty intersection with $M'(x)$ (note x_i may be the same as point x). Applying Lemma 2.2, it follows that $M'(x_i) \subseteq M(x)$. By Lemma 2.4, $M(x) \subseteq C^{2/\beta}$. Putting it together, we obtain $M'(x_i) \subseteq M(x) \subseteq C^{2/\beta} \subseteq C$, which establishes the first part of Property 2.

It remains to show that $C_i^{1/\beta^2} \subseteq C \subseteq C_i$. Since $M'(x_i) \cap M'(x) \neq \emptyset$, we can apply Lemma 2.9 to caps A_i and $C^{1/\beta}$ (for $\lambda = 1$) to obtain $A_i \subseteq (C^{1/\beta})^\beta$. Applying Lemma 2.9 again to caps $C^{1/\beta}$ and A_i (for $\lambda = \beta$), we obtain $(C^{1/\beta})^\beta \subseteq A_i^{\beta^2}$. Thus $A_i \subseteq C \subseteq A_i^{\beta^2}$. Recalling that $C_i = A_i^{\beta^2}$, we obtain $C_i^{1/\beta^2} \subseteq C \subseteq C_i$, as desired.

4 Polytope Approximation

In this section, we will show how to obtain an ε-approximating convex polytope P of low combinatorial complexity. Let K be a convex body in canonical form. Our strategy is as follows. First, we build a set $\mathcal{R}$ of disjoint centrally symmetric convex bodies lying within K and close to its boundary. These bodies will possess certain key properties to be specified later. For each $R \in \mathcal{R}$, we select a point arbitrarily from this body, and let S denote this set of points. The approximation P is defined as the convex hull of S. In Lemma 4.9, we will prove that P is an ε-approximation of K and, in Lemma 4.10, we will apply a deterministic variation of the witness-collector approach [15] to show that P has low combinatorial complexity.

Before delving into the details, we provide a high-level overview of the witness-collector method, adapted to our context. Let $\mathcal{H}$ denote the set of all halfspaces in $\mathbb{R}^d$. We define a

set $\mathcal{W}$ of regions called *witnesses* and a set $\mathcal{C}$ of regions called *collectors*, which satisfy the following properties:

1. Each witness of $\mathcal{W}$ contains a point of S in its interior.
2. Any halfspace $H \in \mathcal{H}$ either contains a witness $W \in \mathcal{W}$ or $H \cap S$ is contained in a collector $C \in \mathcal{C}$.
3. Each collector $C \in \mathcal{C}$ contains a constant number of points of S.

The key idea of the witness-collector method is encapsulated in the following lemma.

▶ **Lemma 4.1.** *Given a set of witnesses and collectors satisfying the above properties, the combinatorial complexity of the convex hull P of S is $O(|\mathcal{C}|)$.*

Proof. We map each face f of P to any maximal subset $S_f \subseteq S$ of affinely independent points on f. Note that this is a one-to-one mapping and $|S_f| \leq d$. In order to bound the combinatorial complexity of P it suffices to bound the number of such subsets S_f.

For a given face f, let H be any halfspace such that $H \cap P = f$. Clearly H does not contain any witness since otherwise, by Property 1, it would contain a point of S in its interior. By Property 2, $H \cap S$ is contained in some collector $C \in \mathcal{C}$. Thus $S_f \subseteq C$. Since $|S_f| \leq d$, it follows that the number of such subsets S_f that are contained in any collector C is at most $\sum_{1 \leq j \leq d} \binom{|C|}{j} = O(|C|^d) = O(1)$, where in the last step we have used the fact that $|C| = O(1)$ (Property 3). Summing over all the collectors, it follows that the total number of sets S_f, and hence the combinatorial complexity of P, is $O(|\mathcal{C}|)$. ◀

A natural choice for the witnesses and collectors would be the convex bodies R_i and the caps C_i, respectively, from Lemma 3.1. Unfortunately, these bodies do not work for our purposes. The main difficulty is that Property 3 could fail, since a cap C_i could intersect a non-constant number of bodies of $\mathcal{R}$, and hence contain a non-constant number of points of S. (To see this, suppose that K is a cylinder in 3-dimensional space. A cap of width $\Theta(\varepsilon)$ that is parallel to the circular flat face of K intersects $\Omega(1/\sqrt{\varepsilon})$ bodies, which will be distributed around the circular boundary of this face.) In this section, we show that it is possible to construct a set of witnesses and collectors that satisfy all the requirements by scaling and translating the convex bodies from Lemma 3.1 into a stratified placement according to their volumes. The properties we obtain are specified below in Lemma 4.4.

The following technical lemma gives upper and lower bounds on the volume of a cap of width α.

▶ **Lemma 4.2.** *Let $K \subset \mathbb{R}^d$ be a convex body in canonical form and let $\alpha < 1$ be a positive real. Then the volume of any cap C of width α is $O(\alpha)$ and $\Omega(\alpha^d)$.*

The following lemma states that containment of caps is preserved if the halfspaces defining both caps are consistently scaled about a point that is common to both caps.

▶ **Lemma 4.3.** *Let K be a convex body and let $\lambda \geq 1$. Let C_1 and C_2 be two caps of K such that $C_1 \subseteq C_2$. Let H_1 and H_2 be the defining halfspaces of C_1 and C_2, respectively. Let H_1' and H_2' be the halfspaces obtained by scaling H_1 and H_2, respectively, by a factor of λ about p, where p is any point in $K \cap C_1$. Let C_1' and C_2' be the caps $K \cap H_1'$ and $K \cap H_2'$, respectively. Then $C_1' \subseteq C_2'$.*

Proof. Given λ and p, consider the affine transformation $f(q) = \lambda(q - p) + p$, which scales space by a factor of λ about p. Thus, $H_1' = f(H_1)$ and $H_2' = f(H_2)$, Since $p \in K$ and $\lambda \geq 1$, it follows directly from convexity that $K \subseteq f(K)$. Given any halfspace H such

that $p \in K \cap H$, it follows that $K \cap f(H) = K \cap f(K \cap H)$. Since, $C_1 \subseteq C_2$, we have $f(K \cap H_1) \subseteq f(K \cap H_2)$, and thus,

$$C_1' \;=\; K \cap f(H_1) \;=\; K \cap f(K \cap H_1) \;\subseteq\; K \cap f(K \cap H_2) \;=\; K \cap f(H_2) \;=\; C_2',$$

as desired. ◀

Our choice of witnesses and collectors will be based on the following lemma. Specifically, the convex bodies $R_1, \ldots, R_k$, will play the role of the witnesses and the regions $C_1, \ldots, C_k$, will play the role of the collectors. The lemma strengthens Lemma 3.1, achieving the critical property that any collector C_i intersects only a constant number of convex bodies of $\mathcal{R}$. As each witness set R_i will contain one point, this ensures that a collector contains only a constant number of input points (Property 3 of the witness-collector system). This strengthening is achieved at the expense of only an extra polylogarithmic factor in the number of collectors needed, compared with Lemma 3.1. Also, the collectors are no longer simple caps, but have a more complex shape as described in the proof (this, however, has no adverse effect in our application).

▶ **Lemma 4.4.** *Let $\varepsilon > 0$ be a sufficiently small parameter, and $\widehat{\varepsilon} = \varepsilon/\log(1/\varepsilon)$. Let $K \subset \mathbb{R}^d$ be a convex body in canonical form. There exists a collection $\mathcal{R}$ of $k = O(1/\widehat{\varepsilon}^{(d-1)/2})$ disjoint centrally symmetric convex bodies $R_1, \ldots, R_k$ and associated regions $C_1, \ldots, C_k$ such that the following hold:*

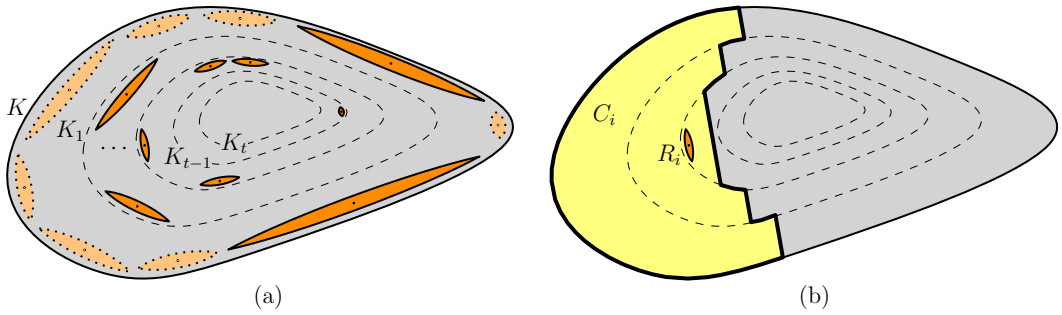
1. *Let C be any cap of width ε. Then there is an i such that $R_i \subseteq C$.*
2. *Let C be any cap. Then there is an i such that either* (i) *$R_i \subseteq C$ or* (ii) *$C \subseteq C_i$.*
3. *For each i, the region C_i intersects at most a constant number of bodies of $\mathcal{R}$.*

As mentioned earlier, our proof of this lemma is based on a stratified placement of the convex bodies from Lemma 3.1, which are distributed among $O(\log(1/\varepsilon))$ layers that lie close to the boundary of K. Let $\alpha = c_1\,\varepsilon/\log(1/\varepsilon)$, where c_1 is a suitable constant to be specified later. We begin by applying Lemma 3.1 to K using $\varepsilon = \alpha$. This yields a collection $\mathcal{R}'$ of $k = O(1/\alpha^{(d-1)/2})$ disjoint centrally symmetric convex bodies $\{R_1', \ldots, R_k'\}$ and associated caps $\mathcal{C}' = \{C_1', \ldots, C_k'\}$. Our definition of the convex bodies R_i and regions C_i required in Lemma 4.4 will be based on R_i' and C_i', respectively. In particular, the convex body R_i will be obtained by translating a scaled copy of R_i' into an appropriate layer, based on the volume of R_i'.

Before describing the construction of the layers, it will be convenient to group the bodies in $\mathcal{R}'$ based on their volumes. We claim that the volume of any convex body R_i' lies between $c_2\alpha^d$ and $c_3\alpha$ for suitable constants c_2 and c_3. By Property 1 of Lemma 3.1, $R_i' \subseteq C_i' \subseteq (R_i')^\lambda$ and C_i' has width $\beta\alpha$, for constants β and λ depending only on d. By Lemma 4.2, the volume of C_i' is $O(\alpha)$ and $\Omega(\alpha^d)$. Since $\mathrm{vol}(R_i') = \Theta(\mathrm{vol}(C_i'))$, the desired claim follows.

We partition the set $\mathcal{R}'$ of convex bodies into t groups, where each group contains bodies whose volumes differ by a factor of at most 2. More precisely, for $0 \le j \le t-1$, group j consists of bodies in $\mathcal{R}'$ whose volume lies between $c_3\alpha/2^j$ and $c_3\alpha/2^{j+1}$. The lower and upper bound on the volume of bodies in $\mathcal{R}'$ implies that the number of groups t can be expressed as $\lfloor c_4 \log(1/\alpha) \rfloor$ for a suitable constant c_4 (depending on c_2 and c_3).

Next we describe how the layers are constructed. We will construct t layers corresponding to the t groups of $\mathcal{R}'$. Let $\gamma = 1 - 4d\beta\alpha$. For $0 \le j \le t$, let T_j denote the linear transformation that represents a uniform scaling by a factor of γ^j about the origin, and let $K_j = T_j(K)$ (see Fig. 4(a)). Note that $K_0 = T_0(K) = K$. For $0 \le j \le t-1$, define layer j, denoted L_j, to be the difference $K_j \setminus K_{j+1}$. Whenever we refer parallel supporting hyperplanes for two bodies K_i and K_j, we assume that both hyperplanes lie on the same side of the origin.



(a) (b)

Figure 4 (a) Stratified placement of the bodies R_i and (b) the region C_i corresponding to a body R_i. (Figures not to scale.)

The following lemma describes some straightforward properties of these layers and the scaling transformations. In particular, the lemma shows that the t layers lie close to the boundary of K (within distance ε) and each layer has a "thickness" of $\Theta(\alpha)$.

▶ **Lemma 4.5.** *Let $\varepsilon > 0$ be a sufficiently small parameter. For sufficiently small constant c_1 in the definition of α (depending on c_4, β, and d), the layered decomposition and the scaling transformations described above satisfy the following properties:*

- **(a)** *For $0 \leq j \leq t-1$, the distance between parallel supporting hyperplanes of K_j and K_{j+1} is at most $2d\beta\alpha$.*
- **(b)** *For $0 \leq j \leq t-1$, the distance between parallel supporting hyperplanes of K_j and K_{j+1} is at least $\beta\alpha$.*
- **(c)** *The distance between parallel supporting hyperplanes of K and K_t is at most ε.*
- **(d)** *For $0 \leq j \leq t$, the scaling factor for T_j is at least 1/2 and at most 1.*
- **(e)** *For $0 \leq j \leq t$, T_j preserves volumes up to a constant factor.*
- **(f)** *For $0 \leq j \leq t$, and any point $p \in K$, the distance between p and $T_j(p)$ is at most $2jd\beta\alpha$.*

We are now ready to define the regions R_i and C_i required in Lemma 4.4. Suppose that R'_i is in group j and let $C'_i = K \cap H'_i$, where H'_i is a halfspace. We define $R_i = T_j(R'_i)$. In order to define C_i, we first define caps $C_{i,r}$ of K_r as $C_{i,r} = K_r \cap T_j(H'_i)$ for $0 \leq r \leq j$. We then define $C_i = \bigcup_{r=0}^{j} C_{i,r}^{\sigma} \cap L_r$, where $\sigma = 4d\beta^2$. (See Fig. 4(b).)

In Lemma 4.6, we show that the regions R_i are contained in layer j if R'_i is in group j. In Lemma 4.7, we establish Properties 1 and 2 of Lemma 4.4. Finally, in Lemma 4.8, we establish Property 3 of Lemma 4.4.

▶ **Lemma 4.6.** *Let $R_i \in \mathcal{R}$. If R'_i is in group j, then $C_{i,j} = T_j(C'_i)$ and $R_i \subseteq C_{i,j} \subseteq L_j$.*

Proof. Let H'_i denote the halfspace as defined above, that is, $C'_i = K \cap H'_i$. By definition, $C_{i,j} = K_j \cap T_j(H'_i) = T_j(K \cap H'_i) = T_j(C'_i)$. By Property 1 of Lemma 3.1, $R'_i \subseteq C'_i$ and C'_i is a cap of K of width $\beta\alpha$. By Lemma 4.52, the distance between any parallel supporting hyperplanes of K and K_1, respectively, is at least $\beta\alpha$. It follows that $R'_i \subseteq C'_i \subseteq L_0 = K \setminus K_1$. Applying the transformation T_j to all these sets yields $R_i \subseteq C_{i,j} \subseteq L_j = K_j \setminus K_{j+1}$. ◀

▶ **Lemma 4.7.** *Let C be any cap of K. Then there is an i such that either* (i) *$R_i \subseteq C$ or* (ii) *$C \subseteq C_i$. Furthermore, if the width of C is ε, then* (i) *holds.*

Proof. Let $C' \subseteq C$ be the cap of width α, whose base is parallel to the base of C. Let H and H' denote the defining halfspaces of C and C', respectively. By Property 2 of Lemma 3.1, there is an i such that $R'_i \subseteq C'$. Suppose that R'_i is in group j. We consider two cases,

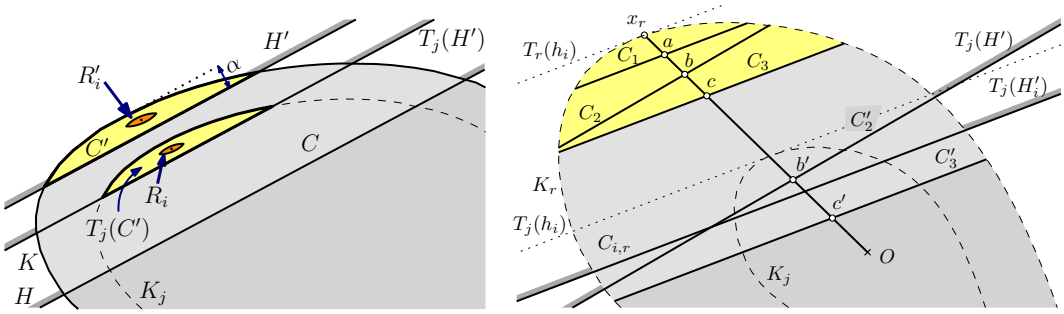


Figure 5 Proof of Lemma 4.7 (a) Case 1 and (b) Case 2. (Figures not to scale.)

depending on whether $T_j(H') \subseteq H$ or $H \subset T_j(H')$. To complete the proof of the lemma, we will show that in the former case, $R_i \subseteq C$ and, in the latter case, $C \subseteq C_i$. Additionally, we will show that if C has width ε, then the former case holds (implying that $R_i \subseteq C$).

Case 1: $T_j(H') \subseteq H$. Arguing as in the proof of Lemma 4.6 (but with C' in place of C'_i), we have $R_i \subseteq T_j(C') = K_j \cap T_j(H') \subseteq L_j$ (see Fig. 5(a)). Observe that $K_j \cap T_j(H') \subseteq K \cap H = C$. Therefore $R_i \subseteq C$.

Also, by Lemma 4.53, the distance between any parallel supporting hyperplanes of K and K_t is at most ε. Since $K_j \cap T_j(H') \subseteq L_j$, it follows that the width of cap $K \cap T_j(H')$ is at most ε. Therefore, if C has width ε, then $T_j(H') \subseteq H$ and Case 1 holds.

Case 2: $H \subset T_j(H')$. Recall that we need to show that $C \subseteq C_i$. Clearly, it suffices to show that $K \cap T_j(H') \subseteq C_i$ since $C = K \cap H \subset K \cap T_j(H')$. In turn, the definition of C_i implies that it suffices to show that for $0 \leq r \leq j$, $T_j(H') \cap K_r \subseteq C^{\sigma}_{i,r}$.

By Property 2 of Lemma 3.1, there is an i such that $(C'_i)^{\phi} \subseteq C' \subseteq C'_i$, where $\phi = 1/\beta^2$. By Property 1 of Lemma 3.1, the widths of the caps $(C'_i)^{\phi}$ and C'_i are α/β and $\beta\alpha$, respectively. Recall that H'_i denotes the defining halfspace for the cap C'_i. Also, let x denote the apex of C'_i, and let h_i denote the supporting hyperplane to K passing through x and parallel to C'_i's base.

Let C_1, C_2, and C_3 denote the caps of K_r obtained by applying the transformation T_r to the caps $(C'_i)^{\phi}$, C', and C'_i, respectively (see Fig. 5(b)). We have $C_1 \subseteq C_2 \subseteq C_3$. Let a, b and c denote the point of intersection of the bases of the caps C_1, C_2 and C_3, respectively, with the line segment Ox. Let b' denote the point of intersection of the base of the cap $K \cap T_j(H')$ with the segment Ox. Let x_r denote the point $T_r(x)$. Consider scaling caps C_2 and C_3 as described in Lemma 4.3, about the point x_r with scaling factor $\rho = \|b'x_r\|/\|bx_r\|$. Let C'_2 and C'_3 denote the caps of K_r obtained from C_2 and C_3, respectively, through this transformation. By Lemma 4.3, $C'_2 \subseteq C'_3$. Our choice of the scaling factor implies that C'_2 is the cap $T_j(H') \cap K_r$. We claim that $C'_3 \subseteq C^{\sigma}_{i,r}$. Note that this claim would imply that $T_j(H') \cap K_r \subseteq C^{\sigma}_{i,r}$, and complete the proof.

To prove the above claim, we first show that $\rho = O(j - r + 1)$. Observe that $\rho = (\|b'b\| + \|bx_r\|)/\|bx_r\| = \|b'b\|/\|bx_r\| + 1$. We have

$$\|bx_r\| \;\geq\; \|ax_r\| \;\geq\; \text{width}(C_1) \;\geq\; \frac{\text{width}((C'_i)^{\phi})}{2} \;\geq\; \frac{\alpha}{2\beta},$$

where in the third inequality, we have used Lemma 4.54 and the fact that $C_1 = T_r((C'_i)^{\phi})$.

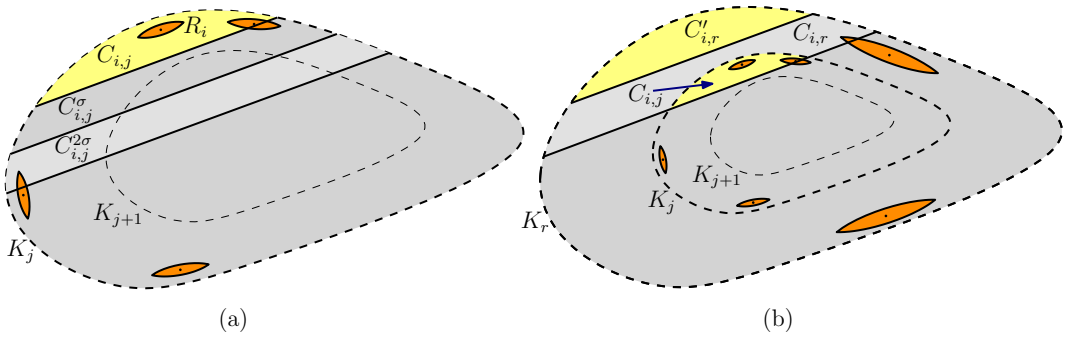


Figure 6 Proof of Lemma 4.8. (Figures not to scale.)

Also, since $T_{j-r}(b) = b'$, it follows from Lemma 4.56 that $\|b'b\|$ is at most $2(j-r)d\beta\alpha$. Substituting the derived bounds on $\|b'b\|$ and $\|bx_r\|$, we obtain $\rho \leq 4d\beta^2(j-r)+1$.

Recall that C'_3 and $C_{i,r}$ are caps of K_r defined by parallel halfspaces. To prove that $C'_3 \subseteq C^{\sigma}_{i,r}$, it therefore suffices to show that $\text{width}(C'_3)/\text{width}(C_{i,r}) \leq \sigma$. We have

$$\text{width}(C'_3) \;=\; \rho \cdot \text{width}(C_3) \;\leq\; \rho \cdot \text{width}(C'_i) \;= \rho\beta\alpha,$$

where in the second step, we have used Lemma 4.54 and the fact that $C_3 = T_r(C'_i)$. Also, it is easy to see that the width of $C_{i,r}$ is the sum of the width of the cap $T_j(C'_i)$ and the distance between the hyperplanes $T_r(h_i)$ and $T_j(h_i)$. Since $\text{width}(C'_i) = \beta\alpha$, by Lemma 4.54, the width of the cap $T_j(C'_i)$ is at least $\beta\alpha/2$. Also, by Lemma 4.52, the distance between the hyperplanes $T_r(h_i)$ and $T_j(h_i)$ is at least $(j-r)\beta\alpha$. It follows that the width of $C_{i,r}$ is at least $\beta\alpha/2 + (j-r)\beta\alpha = (j-r+1/2)\beta\alpha$. Thus,

$$\frac{\text{width}(C'_3)}{\text{width}(C_{i,r})} \;\leq\; \frac{\rho\beta\alpha}{(j-r+1/2)\beta\alpha} \;=\; \frac{\rho}{j-r+1/2} \leq \frac{4d\beta^2(j-r)+1}{j-r+1/2} \;\leq\; 4d\beta^2 \;=\; \sigma,$$

as desired. ◀

▶ **Lemma 4.8.** *For each i, the region C_i intersects $O(1)$ bodies of $\mathcal{R}$.*

Proof. Suppose that R'_i is in group j. Recall that $R_i = T_j(R'_i)$, $C'_i = K \cap H'_i$ and $C_i = \bigcup_{r=0}^{j}(C^{\sigma}_{i,r} \cap L_r)$. We begin by bounding the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,j} \cap L_j$. (See Fig. 6(a).) By Lemma 4.6, $C_{i,j} = T_j(C'_i)$ and $R_i \subseteq C_{i,j} \subseteq L_j$. By Property 1 of Lemma 3.1, we have $C'_i \subseteq (R'_i)^{\lambda}$, which implies that $\text{vol}(R'_i) = \Omega(\text{vol}(C'_i))$. Recall that all the bodies of $\mathcal{R}'$ in group j have the same volumes to within a factor of 2, and so they all have volumes $\Omega(\text{vol}(C'_i))$. By Lemma 4.55, the scaling transformations used in our construction preserve volumes to within a constant factor. Also, recall that the bodies of $\mathcal{R}$ in layer j are scaled copies of the bodies of $\mathcal{R}'$ in group j. It follows that the bodies of $\mathcal{R}$ in layer j all have volumes $\Omega(\text{vol}(C_{i,j}))$.

Next, we assert that any body of $\mathcal{R}$ that overlaps $C^{\sigma}_{i,j} \cap L_j$ is contained within the cap $C^{2\sigma}_{i,j}$. To prove this, recall from the proof of Lemma 3.1 that the bodies of $\mathcal{R}'$ are $(1/5)$-scaled disjoint Macbeath regions with respect to K. It follows that the bodies of $\mathcal{R}$ in layer j are $(1/5)$-scaled disjoint Macbeath regions with respect to K_j. By Lemma 2.5, it now follows that any body of $\mathcal{R}$ that overlaps $C^{\sigma}_{i,j} \cap L_j$ is contained within the cap $C^{2\sigma}_{i,j}$. Since $\text{vol}(C^{2\sigma}_{i,j}) = O(\text{vol}(C_{i,j}))$, and all bodies of $\mathcal{R}$ in layer j have volumes $\Omega(\text{vol}(C_{i,j}))$, it follows by a simple packing argument that the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,j} \cap L_j$ is $O(1)$.

Next we bound the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,r} \cap L_r$, where $0 \leq r < j$. (See Fig. 6(b).) Recall that $C_{i,r} = K_r \cap T_j(H'_i)$. Roughly speaking, we will show that the volume of $C_{i,r}$ exceeds the volume of $C_{i,j}$ by a factor that is at most polynomial in $j - r$, while the volume of the bodies in layer r exceeds the volume of the bodies in layer j by a factor that is exponential in $j - r$. This will allow us to show that the number of bodies of $\mathcal{R}$ that overlap C_i is bounded by a constant. We now present the details.

Define $C'_{i,r} = T_r(C'_i)$. Recall that $C_{i,j} = T_j(C'_i)$. By Lemma 4.55, T_j and T_r preserve volumes up to constant factors, and so $\text{vol}(C'_{i,r}) = \Theta(\text{vol}(C_{i,j}))$. Since the width of C'_i is $\beta\alpha$, by Lemma 4.54, it follows that the width of $C'_{i,r}$ is at least $\beta\alpha/2$. Also, the width of $C_{i,r}$ is upper bounded by the distance between parallel supporting hyperplanes of K_r and K_{j+1} which by Lemma 4.51 is at most $2d\beta\alpha(j - r + 1)$. It follows that the width of $C_{i,r}$ is $O(j - r + 1)$ times the width of $C'_{i,r}$. Recalling that, for $\lambda \geq 1$, the volume of a λ-expansion of a cap is at most λ^d times the volume of the cap, it follows that $\text{vol}(C_{i,r}) = O((j - r + 1)^d) \cdot \text{vol}(C'_{i,r}) = O((j - r + 1)^d) \cdot \text{vol}(C_{i,j})$.

Next, recall that the volume of the bodies of $\mathcal{R}'$ in group r exceeds the volume of the bodies of $\mathcal{R}'$ in group j by a factor of $\Omega(2^{j-r+1})$. It follows from Lemma 4.55 and our construction that the volume of the bodies of $\mathcal{R}$ in layer r exceeds the volume of the bodies of $\mathcal{R}$ in layer j by a factor of $\Omega(2^{j-r+1})$. For the same reasons as discussed above, any body of $\mathcal{R}$ that overlaps $C^{\sigma}_{i,r} \cap L_r$ is contained within $C^{2\sigma}_{i,r}$, and $\text{vol}(C^{2\sigma}_{i,r}) = O(\text{vol}(C_{i,r}))$. Putting this together with the upper bound on $\text{vol}(C_{i,r})$ shown above, we have $\text{vol}(C^{2\sigma}_{i,r}) = O((j - r + 1)^d) \cdot \text{vol}(C_{i,j})$. By a simple packing argument, it follows that the ratio of the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,r} \cap L_r$ to the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,j} \cap L_j$ is $O((j - r + 1)^d / 2^{j-r+1})$. Recall that the number of bodies of $\mathcal{R}$ that overlap $C^{\sigma}_{i,j} \cap L_j$ is $O(1)$. It follows that the number of bodies of $\mathcal{R}$ that overlap $C_i = \bigcup_{r=0}^{j} (C^{\sigma}_{i,r} \cap L_r)$ is on the order of $\sum_{0 \leq r \leq j} (j - r + 1)^d / 2^{j-r+1} = O(1)$, as desired. ◀

Let S be a set of points containing one point inside each body of $\mathcal{R}$ defined in Lemma 4.4 and no other points.

▶ **Lemma 4.9.** *The polytope $P = \text{conv}(S)$ is an ε-approximation of K.*

Proof. A set of points S *stabs* every cap of width ε if every such cap contains at least one point of S. It is well known that if a set of points $S \subset K$ stabs all caps of width ε of K, then $\text{conv}(S)$ is an ε-approximation of K [12]. Let C be a cap of width ε. By Lemma 4.4, Property 1, there is a convex body $R_i \subseteq C$. Since S contains a point that is in R_i, we have that the cap C is stabbed. ◀

To bound the combinatorial complexity of $\text{conv}(S)$, and hence conclude the proof of Theorem 1.1, we use the witness-collector approach [15].

▶ **Lemma 4.10.** *The number of faces of $P = \text{conv}(S)$ is $O(1/\widehat{\varepsilon}^{(d-1)/2})$.*

Proof. Define the witness set $\mathcal{W} = R_1, \ldots, R_k$ and the collector set $\mathcal{C} = C_1, \ldots, C_k$, where the R_i's and C_i's are as defined in Lemma 4.4. As there is a point of S in each body R_i, Property 1 of the witness-collector method is satisfied. To prove Property 2, let H be any halfspace. If H does not intersect K, then Property 2 of the witness-collector method holds trivially. Otherwise let $C = K \cap H$. By Property 2 of Lemma 4.4, there is an i such that either $R_i \subseteq C$ or $C \subseteq C_i$. It follows that H contains witness R_i or $H \cap S$ is contained in collector C_i. Thus Property 2 of the witness-collector method is satisfied. Finally, Property 3 of Lemma 4.4 implies Property 3 of the witness-collector method. Thus, we can apply Lemma 4.1 to conclude that the number of faces of P is $O(|\mathcal{C}|) = O(k)$, which proves the lemma. ◀

References

1. P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. Assoc. Comput. Mach.*, 51:606–635, 2004.
2. G. E. Andrews. A lower bound for the volumes of strictly convex bodies with many boundary points. *Trans. Amer. Math. Soc.*, 106:270–279, 1963.
3. S. Arya, G. D. da Fonseca, and D. M. Mount. Optimal area-sensitive bounds for polytope approximation. In *Proc. 28th Annu. Sympos. Comput. Geom.*, pages 363–372, 2012.
4. S. Arya, T. Malamatos, and D. M. Mount. The effect of corners on the complexity of approximate range searching. *Discrete Comput. Geom.*, 41:398–443, 2009.
5. S. Arya, D. M. Mount, and J. Xia. Tight lower bounds for halfspace range searching. *Discrete Comput. Geom.*, 47:711–730, 2012. doi:10.1007/s00454-012-9412-x.
6. I. Bárány. Intrinsic volumes and f-vectors of random polytopes. *Math. Ann.*, 285:671–699, 1989.
7. I. Bárány. The technique of M-regions and cap-coverings: A survey. *Rend. Circ. Mat. Palermo*, 65:21–38, 2000.
8. I. Bárány. Extremal problems for convex lattice polytopes: A survey. *Contemp. Math.*, 453:87–103, 2008.
9. I. Bárány and D. G. Larman. Convex bodies, economic cap coverings, random polytopes. *Mathematika*, 35:274–291, 1988.
10. K. Böröczky, Jr. Approximation of general smooth convex bodies. *Adv. Math.*, 153:325–341, 2000.
11. H. Brönnimann, B. Chazelle, and J. Pach. How hard is halfspace range searching. *Discrete Comput. Geom.*, 10:143–155, 1993.
12. E. M. Bronshteyn and L. D. Ivanov. The approximation of convex sets by polyhedra. *Siberian Math. J.*, 16:852–853, 1976.
13. E. M. Bronstein. Approximation of convex sets by polytopes. *J. Math. Sci.*, 153(6):727–762, 2008.
14. K. L. Clarkson. Building triangulations using ε-nets. In *Proc. 38th Annu. ACM Sympos. Theory Comput.*, pages 326–335, 2006.
15. O. Devillers, M. Glisse, and X. Goaoc. Complexity analysis of random geometric structures made simpler. In *Proc. 29th Annu. Sympos. Comput. Geom.*, pages 167–176, 2013.
16. R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.
17. G. Ewald, D. G. Larman, and C. A. Rogers. The directions of the line segments and of the r-dimensional balls on the boundary of a convex body in Euclidean space. *Mathematika*, 17:1–20, 1970.
18. P. M. Gruber. Asymptotic estimates for best and stepwise approximation of convex bodies I. *Forum Math.*, 5:521–537, 1993.
19. S. Har-Peled. *Geometric approximation algorithms*. Number 173 in Mathematical surveys and monographs. American Mathematical Society, 2011.
20. F. John. Extremum problems with inequalities as subsidiary conditions. In *Studies and Essays Presented to R. Courant on his 60th Birthday*, pages 187–204. Interscience Publishers, Inc., New York, 1948.
21. A. M. Macbeath. A theorem on non-homogeneous lattices. *Ann. of Math.*, 56:269–293, 1952.
22. P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.
23. R. Schneider. Polyhedral approximation of smooth convex bodies. *J. Math. Anal. Appl.*, 128:470–474, 1987.

Efficient Algorithms to Decide Tightness*

Bhaskar Bagchi[1], Benjamin A. Burton[3], Basudeb Datta[2], Nitin Singh[4], and Jonathan Spreer[3]

1 **Theoretical Statistics and Mathematics Unit, Indian Statistical Institute, Bangalore, India**
bbagchi@isibang.ac.in
2 **Department of Mathematics, Indian Institute of Science, Bangalore, India**
dattab@math.iisc.ernet.in.
3 **School of Mathematics and Physics, The University of Queensland, Brisbane, Australia**
bab@maths.uq.edu.au; j.spreer@uq.edu.au.
4 **IBM India Research Lab, Bangalore, India**
nitisin1@in.ibm.com.

Abstract

Tightness is a generalisation of the notion of convexity: a space is tight if and only if it is "as convex as possible", given its topological constraints. For a simplicial complex, deciding tightness has a straightforward exponential time algorithm, but more efficient methods to decide tightness are only known in the trivial setting of triangulated surfaces.

In this article, we present a new polynomial time procedure to decide tightness for triangulations of 3-manifolds – a problem which previously was thought to be hard. In addition, for the more difficult problem of deciding tightness of 4-dimensional combinatorial manifolds, we describe an algorithm that is fixed parameter tractable in the treewidth of the 1-skeletons of the vertex links. Finally, we show that simpler treewidth parameters are not viable: for all non-trivial inputs, we show that the treewidths of both the 1-skeleton and the dual graph must grow too quickly for a standard treewidth-based algorithm to remain tractable.

1998 ACM Subject Classification G.2.1 [Combinatorics] Combinatorial algorithms, F.2.2 [Non-numerical Algorithms and Problems] Computations on discrete structures, Geometrical problems and computations

Keywords and phrases discrete geometry and topology, polynomial time algorithms, fixed parameter tractability, tight triangulations, simplicial complexes

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.12

1 Introduction

The notion of *convexity* is very powerful in mathematics. Many theorems in many different mathematical fields only hold in the case of a convex base space. However, in geometry and topology, the concept of convexity has significant limitations: most topological spaces simply do not admit a convex representation. That is, most topological features of a space, such as handles or "holes" in the space, are an obstruction to convexity.

* This work is supported by the Department of Industry and Science, Australia under the Australia-India Strategic Research Fund (project AISRF06660). The third author is also supported by the UGC Centre for Advanced Studies.

© Bhaskar Bagchi, Benjamin A. Burton, Basudeb Datta, Nitin Singh, and Jonathan Spreer;
licensed under Creative Commons License CC-BY

Nonetheless, there is a distinct intuition that even for topologically non-trivial spaces, some representations look "more convex" than others. For example, for a solid torus, a doughnut shape is intuitively "more convex" than a coffee mug with one handle.

The idea of *tightness* captures this intuition in a mathematically precise way that applies to a much larger class of topological spaces than just balls and spheres. We give a precise definition in Section 3, but roughly speaking, a particular embedding of a topological space into some Euclidean space is said to be *tight* if it is "as convex as possible" given its topological constraints. In particular, a topological ball or sphere is tight if and only if it is convex.

Originally, tight embeddings were studied by Alexandrov in 1938 as objects that minimise total absolute curvature [1]. Later work by Milnor [34], Chern and Lashof [15] and Kuiper [30] linked the concept to topology, by relating tightness to the sum of the Betti numbers β_i of a d-dimensional topological space (these essentially count i-dimensional "holes"; see Section 2.2 for details). The framework was then applied to polyhedral surfaces by Banchoff [5], and later fully developed in the combinatorial setting by Kühnel [27]. Finally, work by Effenberger [20], and the first and third authors [3] made the concept accessible to computations.

There is a straightforward relaxation of tightness linking it to another powerful concept in geometry and topology: Morse theory (see Section 2.3 for a brief introduction). In a sense, *all* Morse functions on a tight embedding of a topological space must be perfect; that is, they must satisfy the Morse inequalities with equality. [1] Deciding whether a given embedding is tight is closely related to the much-studied problem of finding a perfect Morse function: the former asks if every Morse function is perfect, and the latter asks if there is at least one.

This article deals with tightness in the discrete setting, where our topological spaces are represented as simplicial complexes. Here there are only a finite number of "essentially distinct" ways of (i) embedding the complex into Euclidean space, as used for tightness; or (ii) defining a simplexwise linear "height" function on it, as used in Morse theory. For instance, in the latter setting, the critical points of a simplexwise linear height function are completely determined by the order of the heights of the vertices of the complex.

In this discrete setting, we can replace the notion of a tight embedding with a purely combinatorial notion of a *tight combinatorial manifold*. These are rare but very special objects. For instance, tight combinatorial manifolds are conjectured to be strongly minimal (i.e., to contain the minimum number of faces of every dimension) amongst all triangulations of the same manifold [29, Conjecture 1.3]. More generally, they make deep connections between the *combinatorial* condition of tightness and its *geometric and topological* properties of a manifold, which are still far from being fully understood. Few such connections are known, which emphasises the importance of tightness in discrete and computational topology.

In this discrete setting, both deciding tightness of an embedding and finding a perfect Morse function become (decidable) algorithmic problems. The question remains as to how hard these problems are. It is known that, for discrete Morse functions as defined by Forman [21], finding perfect Morse functions is NP-hard in general [25, 33, 36], but also fixed parameter tractable in the treewidth of the dual graph of the triangulation [10] (see Sections 2.1 and 2.4 for definitions of the dual graph and treewidth respectively). For the related piecewise-linear variant of Morse theory as formulated by Kuiper and Banchoff [5, 19, 27] the hardness remains unknown; however, both theories are closely related since they both discretise smooth Morse theory in similar (but technically different) ways.

[1] This statement only holds in general when applied within the right version of Morse-type theory; namely, the theory of *piecewise linear functions* and the *lower star filtration* [19]. This is sometimes also referred to as the theory of *regular simplex-wise linear functions* [27].

In this paper, we consider the hardness of deciding tightness in the discrete setting. In Section 5 of this article, we give a polynomial time solution in the case of 3-manifolds:

▶ **Theorem 1.** *Let M be a combinatorial 3-manifold with n vertices. There is an algorithm to decide whether or not M is tight with running time polynomial in n. Furthermore, the dominating term in the running time of the algorithm is the time required to compute the first Betti number $\beta_1(M, \mathbb{F}_2)$.*

This is a surprising result, given the close relation between deciding tightness and finding perfect Morse functions which, for Forman's discrete Morse theory, is known to be NP-hard (see above). Furthermore, this polynomial time solution links to a number of other problems in 3-dimensional computational topology where polynomial time procedures are unknown but conjectured to exist. For instance, finding a perfect Morse function on a 3-manifold solves specific instances of the 3-sphere and the unknot recognition problems, both of which are conjectured but not known to be polynomial time solvable [22, 23, 32, 35].

We next move to the more difficult setting of four dimensions. Here we do not obtain a polynomial time algorithm, but we do show that deciding tightness is fixed parameter tractable (see Section 2.4 for an overview of fixed parameter tractability):

▶ **Theorem 2.** *Let M be a combinatorial d-manifold, $d \leq 4$. Then deciding tightness for any field is fixed parameter tractable in the treewidth of the 1-skeletons of the vertex links of M.*

This essentially means that deciding tightness is polynomial time for inputs where the *parameter* – i.e., the treewidth of the 1-skeletons of the vertex links – is universally bounded.

This parameter is complex to describe, but our final result shows that this is necessarily so. In Section 7 we consider the simpler parameters of (i) the treewidth of the 1-skeleton of M; and (ii) the treewidth of the dual 1-skeleton of M, also known as the *dual graph* of M. We show that, for all non-trivial inputs, both of these parameters must grow with the input size, and therefore cannot be universally bounded. In other words, for these simpler parameters, the notion of fixed parameter tractability cannot help.

The results in this paper are not only relevant for computational geometry – they also hold significance for the study of tightness itself. One of the major difficulties in studying tightness is in obtaining explicit examples of tight combinatorial manifolds. Some infinite families are known for simple manifolds [27, 17, 8]; beyond these, only sporadic examples are known. However, these sporadic examples feature several of the most fascinating triangulations in the field, including minimal triangulations of the complex projective plane [28], the K3 surface [14], and a triangulation of an 8-manifold conjectured to be PL-homeomorphic to the quaternionic plane [6] (see [29] for a comprehensive overview of many more examples).

Most of these examples were proven to be tight using theoretical arguments specific to each case. Fast algorithms, such as those presented here in Theorems 1 and 2, will open the door to finding and exploring new tight combinatorial manifolds using powerful computer-assisted techniques.

The proofs in this paper are relatively simple – they use significant theoretical groundwork from the literature to do the heavy lifting, as outlined through Sections 2–4. The main contribution of this paper is to bring together significant results from combinatorial topology, computational complexity and Morse theory to build tractable and fixed parameter tractable algorithms for problems that have until now been beyond the realm of computation.

2 Preliminaries

2.1 Combinatorial manifolds

Let C be an abstract simplicial complex, i.e., a simplicial complex without a particular embedding. For any vertex $v \in C$, the *star of v in C* is the set of all faces of C containing v, as well as all of their subfaces, and is denoted by $\mathrm{st}_C(v)$. The boundary of $\mathrm{st}_C(v)$, that is, all faces of $\mathrm{st}_C(v)$ which do not contain v, is called the *link of v in C*, written $\mathrm{lk}_C(v)$. Given an abstract simplicial complex C, its underlying set $|C|$ is called the *geometric carrier of C*. An abstract simplicial complex is said to be *pure of dimension d* if all of its maximal faces (that is, faces which are not contained in any other face as a proper surface) are of dimension d.

A *combinatorial d-manifold* is an abstract pure simplicial complex M of dimension d in which all vertex links are combinatorial $(d-1)$-dimensional standard PL-spheres. A combinatorial $(d-1)$-dimensional standard PL sphere is a combinatorial $(d-1)$-manifold whose underlying space, i.e., its *geometric carrier*, is a PL standard sphere.

The *f-vector* of M is the $(d+1)$-tuple $f(M) = (f_0, f_1, \dots f_d)$ where f_i denotes the number of i-dimensional faces of M. The set of vertices of M is denoted by $V(M)$, and the d-dimensional faces of M are referred to as *facets*.

We call M *k-neighbourly* if $f_{k-1} = \binom{f_0}{k}$, that is, if M contains all possible $(k-1)$-dimensional faces. Given a combinatorial manifold M with vertex set $V(M)$ and $W \subset V(M)$, the *sub-complex of M induced by W* is the simplicial complex $M[W] = \{\sigma \in M \,|\, V(\sigma) \subset W\}$, i.e., the simplicial complex of all faces of M whose vertices are contained in W.

A pure simplicial complex of dimension d is said to be a *weak pseudomanifold* if every $(d-1)$-dimensional face is contained in at most two facets. Naturally, any combinatorial manifold is a weak pseudomanifold. However, a weak pseudomanifold is not always a combinatorial manifold – it might allow singularities around faces of co-dimension ≥ 2 (e.g., the apex of a double cone). Given a weak pseudomanifold M, the *dual graph* of M is the graph whose vertices represent the facets of M, and whose edges represent gluings between the facets along common $(d-1)$-faces of M; this is denoted $\Gamma(M)$. Weak pseudomanifolds are the most general class of simplicial complexes for which a dual graph can be defined.

For brevity, we call a combinatorial manifold a *triangulation* of the underlying space.

2.2 Homology and $\mathbb{F}$-orientability

Given a d-dimensional topological space M and a field $\mathbb{F}$, the *homology groups* of M are a series of $\mathbb{F}$-vector spaces $H_\star(M, \mathbb{F}) = (H_0(M, \mathbb{F}),\ H_1(M, \mathbb{F}),\ \dots,\ H_d(M, \mathbb{F}))$ associated with M. Roughly speaking, the ith homology group counts i-dimensional "holes" in M.

Homology can be defined on abstract simplicial complexes as follows. Let K be a d-dimensional abstract simplicial complex with an ordering on its vertices $V(K)$, and let $\mathbb{F}$ be a field. The *group of i-chains of K* $(0 \leq i \leq d)$, denoted $C_i(K, \mathbb{F})$, is the group of formal sums of ordered i-dimensional faces of K with coefficients in $\mathbb{F}$. The *boundary operator* is a linear operator $\partial_i : C_i(K, \mathbb{F}) \to C_{i-1}(K, \mathbb{F})$ that maps each i-face to its $(i-1)$-dimensional boundary: $\partial_i(v_0, \cdots, v_i) = \sum_{j=0}^{i} (-1)^j (v_0, \cdots, \widehat{v_j}, \cdots, v_i)$, where $(v_0, \dots, v_i)$ denotes the ordered i-face with vertices $v_0, \dots, v_i$, and $\widehat{v_j}$ means v_j is deleted from the tuple. Denote by $Z_i(K, \mathbb{F})$ and $B_{i-1}(K, \mathbb{F})$ the kernel and the image of ∂_i respectively. Observing $\partial_i \circ \partial_{i+1} = 0$, we define the ith (simplicial) homology group $H_i(K, \mathbb{F})$ of K to be the $\mathbb{F}$-vector space $H_i(K, \mathbb{F}) = Z_i(K, \mathbb{F})/B_i(K, \mathbb{F})$. The rank of $H_i(M, \mathbb{F})$ is called the *ith Betti number* of M, denoted $\beta_i(M, \mathbb{F})$. The field $\mathbb{F}$ is called the *field of coefficients*. This construction is a topological invariant, in that two distinct simplicial complexes triangulating the same topological space will have isomorphic homology groups.

As an example, the circle C and the solid torus T both have exactly one "1-dimensional hole". Hence, they both have first homology group $H_1(C,\mathbb{F}) = H_1(T,\mathbb{F}) = \mathbb{F}$ (for any $\mathbb{F}$), and $\beta_1(T,\mathbb{F}) = 1$. The surface of a torus, however, has two "1-dimensional holes" and thus first homology group $H_1(T,\mathbb{F}) = \mathbb{F}^2$. The 2-dimensional sphere S has trivial first homology group $H_1(S,\mathbb{F}) = 0$. For more details about homology theory see [24].

A triangulation of a d-manifold is *orientable* if all of its d-faces Δ, and thus their boundaries $\partial_d\Delta$, can be endowed with a sign such that d-faces are glued together along their boundary $(d-1)$-faces with opposite signs in a globally consistent way. Otherwise the triangulation is called *non-orientable*. For instance, triangulations of the 2-sphere are orientable, but triangulations of the Möbius strip are not. Orientability is likewise a topological invariant.

Orientability can be generalised in terms of homology. A triangulation of a manifold is $\mathbb{F}$-*orientable* if it forms a d-dimensional void; that is, if $H_d(M,\mathbb{F}) = \mathbb{F}$ (instead of 0). All manifolds are orientable with respect to the field with two elements $\mathbb{F}_2$.

The homology groups of orientable manifolds satisfy a special relation known as *Poincaré duality*: if $\mathbb{F}$ is a field and M is an $\mathbb{F}$-orientable d-manifold, then $\beta_i(M,\mathbb{F}) = \beta_{d-i}(M,\mathbb{F})$. We make use of this fact in our algorithms to decide tightness for 3- and 4-manifolds.

2.3 Piecewise linear Morse theory

Smooth Morse theory studies functions $f : M \to \mathbb{F}$ from a manifold M to the real numbers. Here we introduce a piecewise linear analogue to this theory as defined in [19, 27].

Let M be a d-dimensional combinatorial manifold. A function $f : |M| \to \mathbb{R}$ is called *piecewise linear*, or *regular simplexwise linear (rsl)*, if $f(v) \neq f(w)$ for any two vertices $v \neq w$ of M and f is linear when restricted to any simplex of M.

Fix a field $\mathbb{F}$. A vertex $x \in V(M)$ is said to be *critical with respect to* $\mathbb{F}$ for an rsl-function $f : |M| \to \mathbb{R}$ if $H_\star(\mathrm{lk}_M(x)^-,\mathbb{F}) \neq (0,\dots,0)$. Here $\mathrm{lk}_M(x)^-$ denotes the *lower link* of x with respect to f, defined as $\mathrm{lk}_M(x)^- = \mathrm{lk}_M(x)[\{y \in V(\mathrm{lk}_M(x)) \,|\, f(y) \leq f(x)\}]$. Essentially, a point x is critical whenever an i-dimensional hole is contained in $M[\{y \in V(M) \,|\, f(y) \leq f(x)\}]$, but not in $M[\{y \in V(M) \,|\, f(y) < f(x)\}]$. In the language of filtrations, the hole appears at value $f(x)$ in the filtration defined by f. Such a hole must intersect the lower link $\mathrm{lk}_M(x)^-$ in a hole of dimension $(i-1)$, yielding the homology relation described above.

We call a vertex x *critical of index i and multiplicity m* if $\beta_i(\mathrm{lk}_M(x)^-,\mathbb{F}) = m$. The vector $(\beta_0(\mathrm{lk}_M(x)^-,\mathbb{F}),\dots,\beta_d(\mathrm{lk}_M(x)^-,\mathbb{F}))$ is called *multiplicity vector of x with respect to* $\mathbb{F}$. The sum $\mathbf{m}_i(f,\mathbb{F}) := \sum_{x\in M} \beta_i(\mathrm{lk}_M(x)^-,\mathbb{F})$ is called the *number of critical points of f of index i with respect to* $\mathbb{F}$. The sum of $\mathbf{m}_i(f,\mathbb{F})$ over all indices i is called the *number of critical points of f with respect to* $\mathbb{F}$. Note that in the theory of piecewise linear functions, higher order multiplicities cannot always be avoided.

▶ **Theorem 3** (Morse relations, [31]). *Let M be a combinatorial d-manifold, $f : |M| \to \mathbb{R}$ a piecewise linear function, and $\mathbb{F}$ a field. Then the following statements hold:*

i) $\beta_i(M,\mathbb{F}) \leq \mathbf{m}_i(f,\mathbb{F})$,

ii) $\sum_{i=0}^{d}(-1)^i\mathbf{m}_i(f,\mathbb{F}) = \chi(M) = \sum_{i=0}^{d}(-1)^i\beta_i(M,\mathbb{F})$,

where $\chi(M)$ is an invariant of M called the Euler characteristic.

If there exists a field $\mathbb{F}$ such that $\mathbf{m}_i(f,\mathbb{F}) = \beta_i(M,\mathbb{F})$ for all $0 \leq i \leq d$, then f is called *perfect*. Geometrically speaking this means that M, seen from the direction described by f, is "as convex as possible".

2.4 Parameterised complexity and treewidth

The framework of *parameterised complexity*, as introduced by Downey and Fellows [18], provides a refined complexity analysis for hard problems. Given a typically NP-hard problem p with A as its set of possible inputs, we choose some *parameter* $k : A \to \mathbb{N}$, which grades the inputs according to their corresponding parameter values. This *parameterised version* of p is then said to be *fixed parameter tractable (FPT)* with respect to parameter k if p can be solved in time $O(f(k) \cdot n^{O(1)})$, where f is an arbitrary (computable) function independent of the input size n and k is the parameter value of the input. Note that the exponent of n must be independent of k. In essence, if p is FPT in parameter k, then k (and not the problem size) encapsulates the hardness of p.

A priori, a parameter can be many things, such as the maximum vertex degree of a graph, the sum of the Betti numbers of a triangulation, or the output size. However, the significance of an FPT result strongly depends on specific properties of the parameter – the parameter should be small for interesting classes of problem instances and ideally efficient to compute.

In the setting of computational topology, the *treewidths* of various graphs associated with triangulations turn out to be such good choices of parameter [10, 12, 9]. Informally, the treewidth of a graph measures how "tree-like" a graph is. The precise definition is as follows.

▶ **Definition 4** (Treewidth)**.** A *tree decomposition* of a graph G is a tree $T = (B, E)$ whose vertices $\{B_i \mid i \in I\}$ are called *bags*. Each bag B_i is a subset of vertices of G, and we require:

- every vertex of G is contained in at least one bag (*vertex coverage*);
- for each edge of G, at least one bag must contains both its endpoints (*edge coverage*);
- the induced subgraph of T spanned by all bags sharing a given vertex of G must form a (connected) subtree of T (*subtree property*).

The *width* of a tree decomposition is defined as $\max |B_i| - 1$, and the *treewidth* of G is the minimum width over all tree decompositions.

When describing FPT algorithms which operate on tree decompositions, the following construction has proven to be extremely convenient.

▶ **Definition 5** (Nice tree decomposition)**.** A tree decomposition $T = (\{B_i \mid i \in I\}, E)$ is called a *nice tree decomposition* if T can be expressed as rooted tree for which:

1. Every bag of the tree T has at most two children.
2. The bag B_r at the root of the tree (called the *root bag*) has $|B_r| = 1$.
3. If bag B_j has no children, then $|B_j| = 1$ (in this case B_j is called a *leaf bag*).
4. If a bag B_i has two children B_j and B_k, then the sets B_i, B_j and B_k are identical (in this case B_i is called a *join bag*).
5. If a bag B_i has one child B_j, then either:
   a. $|B_i| = |B_j| + 1$ and $B_j \subset B_i$ (in this case B_i is called an *introduce bag*); or
   b. $|B_j| = |B_i| + 1$ and $B_i \subset B_j$ (in this case B_i is called a *forget bag*).

Nice tree decompositions are small and easy to construct by virtue of the following.

▶ **Lemma 6** ([26])**.** *Given a tree decomposition of a graph G that has width k and $O(n)$ bags, where n is the number of vertices of G, we can find a nice tree decomposition of G that also has width k and $O(n)$ bags in time $O(n)$.*

We make use of nice tree decompositions in Section 6.

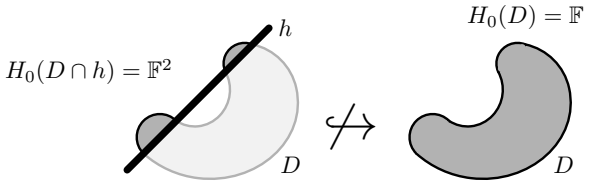


Figure 1 Non-convex embedding of the disk D in the plane. There exist a half space h such that $h \cap D$ has homological features (two connected components) which D has not.

3 Tightness

In its most general form, *tightness* is defined for compact connected subsets of Euclidean space. Here and in the following, $H_\star(M, \mathbb{F}) = (H_0(M, \mathbb{F}), H_1(M, \mathbb{F}), \ldots, H_d(M, \mathbb{F}))$ denote the simplicial homology groups of a d-manifold M with coefficients in the field $\mathbb{F}$.

Intuitively, a manifold M embedded into some Euclidean space E^d is tight if intersecting it with a half space $h \subset E^d$ does not introduce any topological (more precisely, homological) features in $h \cap M$ which are topologically (homologically) trivial in $M \subset E^d$. For instance, think of a non-convex embedding of the 2-dimensional disk D into E^2, as illustrated in Figure 1. We can always cut D with a half space $h \subset E^d$ such that $h \cap D$ is disconnected, or, in homology terms, $H_0(h \cap D, \mathbb{F}) = \mathbb{F}^2$ (note that multiple connected components are "0-dimensional holes"). But D itself is connected and thus $H_0(D, \mathbb{F}) = \mathbb{F}$. Hence a topological feature of $h \cap D$ disappears in D, and so D is not tight.

For a different type of non-tight embedding, imagine an embedding of the 3-dimensional ball B into E^3 with a dent. Because of the dent, we can find a half space $h \subset E^3$ such that $h \cap B$ is a solid torus. Thus, $h \cap B$ is still connected, but we have $H_1(h \cap B, \mathbb{F}) = \mathbb{F}$ whereas $H_1(B, \mathbb{F}) = 0$ and B is not tight. We formalise this intuition through the following definition.

▶ **Definition 7** (k-tightness and tightness [27])**.** A compact connected subset $M \subset E^d$ is called *k-tight with respect to a field $\mathbb{F}$* if, for every open or closed half space $h \subset E^d$, the induced homomorphism $H_k(h \cap M, \mathbb{F}) \to H_k(M, \mathbb{F})$ is injective. If $M \subset E^d$ is k-tight with respect to $\mathbb{F}$ for all k, $0 \leq k \leq d$, then it is called *$\mathbb{F}$-tight*.

If a connected subset $M \subset E^d$ is referred to as *tight* without specifying a field, then it is usually understood that there exists a field $\mathbb{F}$ such that M is $\mathbb{F}$-tight. See Figure 2 on the left for a tight embedding of the torus into Euclidean 3-space.

If M is given as an abstract simplicial complex, tightness can be formulated as a combinatorial condition. In this setting, all half spaces from the geometric definition are replaced by subsets of vertices $W \subset V(M)$ of M, thought of as lying inside the half space. In particular, there are only $2^{|V(M)|}$ distinct subsets and thus only $2^{|V(M)|}$ "essentially distinct" half spaces to consider. The restriction $M \cap h$ from the geometric definition translates to combinatorics via the concept of induced subcomplexes $M[W]$.

▶ **Definition 8** (k-tightness and tightness [3, 27])**.** Let C be an abstract simplicial complex and let $\mathbb{F}$ be a field. We say that C is *tight with respect to* $\mathbb{F}$ if, for all subsets $W \subset V(C)$ and all $0 \leq k \leq d$, the induced homomorphism $H_k(C[W], \mathbb{F}) \to H_k(C, \mathbb{F})$ is injective.

See Figure 2 in the center and on the right hand side for the unique tight triangulations of the 2-sphere (the boundary of the tetrahedron) and the torus (the so-called Möbius torus, see [16] for an embedding of this complex into Euclidean 3-space with straight lines). Take a moment to verify that no induced subcomplex introduces holes which are filled in the full complex, or disconnected components which become connected in the full complex.

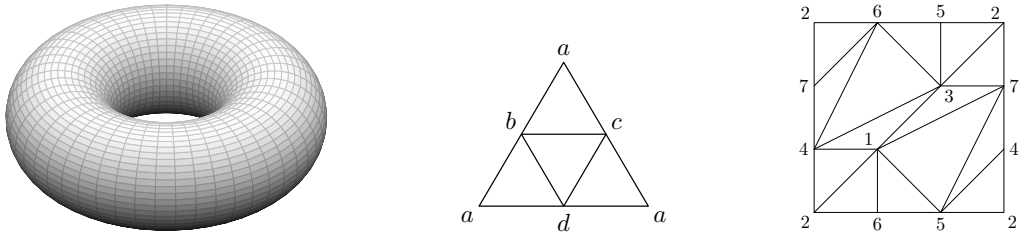


Figure 2 Left: tight embedding of the torus into E^3. Center: unique tight triangulation of the 2-sphere, the boundary of a tetrahedron. Right: unique tight triangulation of the torus with 7 vertices. Vertices with equal labels are identified.

Note that the above definition does not depend on a specific embedding. However, Definition 7 can be linked to Definition 8 by considering the standard embedding of a simplicial complex C into the $(|V(C)|-1)$-simplex (every vertex of C is sent to a unit vector in $E^{|V(C)|}$). Also, notice that in Definition 8, and as mentioned above, tightness is a definition depending on a finite number of conditions, namely $2^{|V(C)|}$, giving rise to an exponential time algorithm to decide tightness. Unless otherwise stated we work with Definition 8.

There are many criteria in the literature on when a simplicial complex is tight. See [29] for a more thorough survey of the field and [3, 4] for a summary of recent developments.

One of the most fundamental criteria says that an orientable triangulated surface is tight if and only if it is 2-*neighbourly* [27, Section 2A] (i.e., its set of edges forms a clique, see Section 2.1).

It is not difficult to see that the condition of 2-neighbourliness is necessary for tightness: Assume a connected simplicial complex C has two vertices u and v not connected by an edge, then the induced subcomplex $C[\{u, v\}]$ is not connected, and thus C cannot be 0-tight. Note that for dimensions greater than two, and for 2-dimensional complexes different from combinatorial surfaces, the converse of this statement no longer holds: there are 2-neighbourly complexes which are not tight.

In particular, for $d > 2$, no easy-to-check characterisation of tightness for general combinatorial d-manifolds is known (see [4] for a full characterisation of tightness of combinatorial 3-manifolds with respect to fields of odd characteristic).

Instead, the above condition for combinatorial surfaces generalises to combinatorial 3- and 4-manifolds in the following way.

▶ **Theorem 9** (Bagchi, Datta). *An $\mathbb{F}$-orientable combinatorial manifold of dimension ≤ 4 is $\mathbb{F}$-tight if and only if it is 0-tight and 1-tight with respect to $\mathbb{F}$.*

This directly follows from Theorem 2.6 (c) in [3]. The $\mathbb{F}$-orientability is necessary to apply Poincaré duality to the Betti numbers.

The two main results of this paper (presented in Sections 5 and 6) make use of this fact, yielding (i) a polynomial time procedure to decide tightness for combinatorial 3-manifolds, and (ii) a fixed parameter tractable algorithm for combinatorial 4-manifolds.

The σ- and μ-vectors

We briefly review the definition of the σ- and μ-vectors as introduced in [3] by the first and third authors. These *combinatorial invariants* build the foundation of a more combinatorial study of tightness of simplicial complexes. In essence they count the number of critical points of all piecewise linear functions on a complex simultaneously in order to detect a function

with too many critical points (i.e., a non-perfect Morse function) which then can be used to conclude that the complex is not tight.

▶ **Definition 10** (Bagchi, Datta [3]). Let C be a simplicial complex of dimension d. The σ-vector $(\sigma_0, \sigma_1, \ldots, \sigma_d)$ of C with respect to a field $\mathbb{F}$ is defined as

$$\sigma_i = \sigma_i(C;\mathbb{F}) := \sum_{A \subseteq V(C)} \frac{\tilde{\beta}_i(C[A],\mathbb{F})}{\binom{f_0(C)}{|A|}}, \quad 0 \leq i \leq d,$$

where $\tilde{\beta}_i$ denotes the reduced ith Betti number ($\beta_i = \tilde{\beta}_i$ for $i > 0$, and $\beta_0 = \tilde{\beta}_0 + 1$). For $i > \dim(C)$, we formally set $\sigma_i(X;\mathbb{F}) = 0$.

In other words, $\sigma_i(C,\mathbb{F})$ adds the number of i-dimensional homological features, $0 \leq i \leq d$, in induced subcomplexes $C[A]$, weighted by the number of subsets $A \subset V(C)$ of size $|A|$. Definition 10 can then be used to define the μ-vector.

▶ **Definition 11** (Bagchi [2] and Bagchi, Datta [3]). Let C be a 2-neighbourly simplicial complex of dimension d, $n = |V(C)|$. We define

$$\mu_0 = \mu_0(C;\mathbb{F}) := 1 \tag{1}$$
$$\mu_i = \mu_i(C;\mathbb{F}) := \delta_{i1} + \frac{1}{n} \sum_{v \in V(C)} \sigma_{i-1}(\mathrm{lk}_v(C)), \ 1 \leq i \leq d,$$

where δ_{ij} is the Kronecker delta.

Note that $\mu_1(C;\mathbb{F})$ only depends on the 0-homology of subcomplexes of C and 0-homology is independent of the field $\mathbb{F}$. Hence, we sometimes write $\mu_1(X)$ instead of $\mu_1(X;\mathbb{F})$.

Intuitively speaking, μ_i averages over the number of homological features in the vertex links of dimension $i-1$. Thus, in some sense μ_i counts the average number of critical points of index i of a piecewise linear function on C (see the definition of critical points for piecewise linear functions in Section 2.3). The following result is thus essentially a consequence of the Morse relations, see Theorem 3.

▶ **Lemma 12** (Bagchi, Datta [3]). *Let M be an $\mathbb{F}$-orientable, 2-neighbourly, combinatorial closed d-manifold, $d \leq 4$. Then $\beta_1(M;\mathbb{F}) \leq \mu_1(M)$, and equality holds if and only if M is $\mathbb{F}$-tight.*

This is a special case of a much more general result, but suffices for the purpose of this work. To read more about many recent advances in studying tightness of simplicial complexes using the framework of combinatorial invariants, see [4].

4 Vertex links of tight 3-manifolds

Before we give a description of our polynomial time algorithm in Section 5, we first need to have a closer look at the vertex links of tight combinatorial 3-manifolds and a way to accelerate the computation of their σ-vectors.

▶ **Theorem 13** (Bagchi, Datta, Spreer [4]). *Let M be a tight triangulation of a closed combinatorial 3-manifold. Then each vertex link of M is a combinatorial 2-sphere obtained from a collection of copies of the boundary of the tetrahedron S^2_4 and the boundary of the icosahedron I^2_{12} glued together by iteratively cutting out triangles and identifying their boundaries.*

Furthermore, we have the following decomposition theorem for the σ-vector.

▶ **Theorem 14** (Bagchi, Datta, Spreer [4]). *Let C_1 and C_2 be induced subcomplexes of a simplicial complex C and $\mathbb{F}$ be a field. Suppose $C = C_1 \cup C_2$ and $K = C_1 \cap C_2$. If K is k-neighbourly, $k \geq 2$, then*

$$\sigma_i(C;\mathbb{F}) = (f_0(C)+1)\left(\frac{\sigma_i(C_1;\mathbb{F})}{f_0(C_1)+1} + \frac{\sigma_i(C_2;\mathbb{F})}{f_0(C_2)+1} - \frac{\sigma_i(K;\mathbb{F})}{f_0(K)+1}\right)$$

for $0 \leq i \leq k-2$.

Moreover, we call a simplicial complex C a *primitive simplicial complex* if it does *not* admit a splitting $C_1 \cup C_2 = C$ such that $K = C_1 \cap C_2$ is k-neighbourly, $k \geq 2$.

In particular, this theorem applies to the σ_0-value of a combinatorial 2-sphere S which can be split into two discs D_1 and D_2 along a common triangle K (which is 2-neighbourly), i.e. $S = D_1 \cup_K D_2$. In this case we write $S = S_1 \# S_2$, where S_i is the 2-sphere obtained from D_i, $1 \leq i \leq 2$, by pasting a triangle along the boundary. Note that whether or not we paste the last triangle into D_i does not change the σ_0-value of the construction. Now, Theorem 14, together with the explicit computation of the σ_0-value of the tetrahedron and the icosahedron, gives rise to the following statement.

▶ **Corollary 15** (Bagchi, Datta, Spreer [4]). *Let $k, \ell \geq 0$ and $(k,\ell) \neq (0,0)$. Then*

$$\sigma_0(\, kI_{12}^2 \,\#\, \ell S_4^2\,) = (9k+\ell+3)\left(\frac{617}{1716}k + \frac{1}{20}\ell - \frac{1}{4}\right).$$

5 Deciding tightness of 3-manifold triangulations in polynomial time

In this section we give a proof of Theorem 1, that is, we describe a polynomial time procedure to decide tightness for combinatorial 3-manifolds. The dominant part of the running time of the procedure accounts for computing the first Betti number of the combinatorial 3-manifold which runs in $O(n^6) = O(t^3)$ time, where n is the number of vertices and t is the number of tetrahedra of the triangulation.

The algorithm accepts any 3-dimensional simplicial complex M on n vertices, given as a list of abstract tetrahedra, i.e., subsets of size four of $\{1,\ldots,n\}$, together with a field $\mathbb{F}$ of characteristic $\chi(\mathbb{F})$ as input. It checks if M is an $\mathbb{F}$-orientable combinatorial manifold and, in the case it is, decides whether or not M is tight with respect to $\mathbb{F}$.

We use the fact that, by Lemma 12, a 2-neighbourly, $\mathbb{F}$-orientable combinatorial 3-manifold M is $\mathbb{F}$-tight if and only if $\beta_1(M;\mathbb{F}) = \mu_1(M)$.

First note that there is an $O(n^2\log(n))$ procedure to determine whether an n-vertex 3-dimensional simplicial complex is an $\mathbb{F}$-orientable 2-neighbourly combinatorial manifold:

- Check that M has $\binom{n}{2}$ edges and $(\binom{n}{2} - n)$ tetrahedra, which can be done in $O(n^2)$ time.
- Check that each of the triangles occurs exactly in two tetrahedra and store this gluing information (this can be done in almost quadratic time $O(n^2\log n)$).
- Compute all n vertex links of M. Since M is 2-neighbourly by the above, each vertex link must have $n-1$ vertices. Moreover, if M is a combinatorial manifold, each of the vertex links triangulates a 2-sphere. Since an $(n-1)$-vertex 2-sphere must have $2n-6$ triangles, we can either compute these vertex links in $O(n)$ time, or else conclude that M is not a combinatorial 3-manifold because some vertex link exceeds this size.

- Check that each vertex link is a 2-sphere. Since we know from the above that every triangle in M is contained in exactly two tetrahedra, and hence each edge in a link is contained in two triangles, this can be done by computing the Euler characteristic of each link, again a linear time procedure for each vertex link.
- If the characteristic of $\mathbb{F}$ is distinct from 2, use the gluing information from above to compute an orientation on M in quadratic time. Otherwise M is always $\mathbb{F}$-orientable.

Now, if M fails to be a combinatorial 3-manifold we stop. If M is a combinatorial 3-manifold, but either not 2-neighbourly or not $\mathbb{F}$-orientable we conclude that M is not tight (note that a non-orientable manifold can never be tight, see [3, Proposition 2.9 (b)]).

In case M is 2-neighbourly and $\mathbb{F}$-orientable we now have to test whether $\beta_1(M;\mathbb{F}) = \mu_1(M)$. More precisely, we must compute $\sigma_0(\mathrm{lk}_M(v))$ for all vertices $v \in V(M)$, a procedure which naïvely requires the analysis of $O(2^n)$ induced subcomplexes of $\mathrm{lk}_M(v)$.

However, using Corollary 15, computing the σ_0-value of the vertex link of a tight combinatorial 3-manifold simplifies to the following algorithm to be carried out for each vertex link S (note that because of the 2-neighbourliness, every vertex link needs to contain $n-1$ vertices, $3n-9$ edges, and $2n-6$ triangles):

- Enumerate all induced 3-cycles of S and split S along these cycles into separate connected components. This can be done in $O(n^3)$ time by first storing a list of adjacent edges for each vertex and a list of adjacent triangles for each edge, and then running over all vertex subsets of size three checking if the induced subcomplex is an empty 3-cycle.
- For each connected component, check if the 1-skeleton of the component is isomorphic to the graph of either S^2_4 or I^2_{12}. If not, return that M is not tight. Otherwise, sum up the number of connected components of each type. Note that each component can be processed in constant time, but that there is a linear number of components.
- Use Corollary 15 to compute the σ_0-value of the link.

Add up all σ_0-values and divide by n to obtain $\mu_1(M)$. Now, we know from Lemma 12 that M is tight if and only if $\mu_1(M) = \beta_1(M,\mathbb{F})$. Thus, if $\mu_1(M)$ is not an integer, M cannot be tight. This step overall requires a running time of $O(n^3)$.

Finally, suppose $\mu_1(M)$ is an integer. In this case, we must compute $\beta_1(M,\mathbb{F})$. For all $\mathbb{F}$, this information is encoded in $H_1(M,\mathbb{Z})$ which can be computed in $O(n^6)$ by determining the Smith normal form of the boundary matrix. Hence, this last step dominates the running time of the algorithm.

Altogether, checking for tightness can be done in the same time complexity as computing homology, a task which is considered to be easy in computational topology.

Furthermore, recent theoretical results in [4] show that, if the characteristic of $\mathbb{F}$ is odd, then M must be what is called a 2-neighbourly, *stacked* combinatorial manifold. Such an object can be identified in $O(n^2 \log(n))$ time (i.e., almost linear in the number of tetrahedra).

6 A fixed parameter tractable algorithm for dimension four

In the previous section, we prove that deciding tightness of 3-manifolds can be done efficiently. However, the algorithm for dimension three relies heavily on special properties of the vertex links. No such characterisation of the vertex links is known in higher dimensions.

However, both Lemma 12 and Theorem 14 can still be applied in the 4-dimensional setting. Hence, any computation of the σ_0-value of the vertex link of some combinatorial 4-manifold immediately reduces to computing the σ_0-value of the primitive components of that vertex link. For the remaining primitive pieces we have the following.

▶ **Theorem 16.** *Let C be a simplicial complex C whose 1-skeleton has treewidth $\leq k$. Then there exists an algorithm to compute $\sigma_0(C)$ in $O(f(k)\ n^5)$ time, where n is the number of vertices of C.*

Proof. We give an overview of the structure of this algorithm.

Note that for a simplicial complex C, $\sigma_0(C)$ only depends on the 1-skeleton C_1 of C. Thus, let $T = (B, E)$ be a nice tree decomposition of C_1. We write $B = \{B_1, B_2, \ldots, B_r\}$, where the bags are ordered in a bottom-up fashion which is compatible with a dynamic programming approach. In other words: the tree T is rooted; whenever B_i is a parent of B_j we have $i > j$; and our algorithm processes the bags in the order $B_1, \ldots, B_r$.

For each bag B_i, we consider the induced subgraph $C_1[B_i]$ spanned by all vertices in the bag B_i. Furthermore, we denote the induced subgraph of the 1-skeleton spanned by all vertices contained in bags in or underneath B_i by $C^-_{1,i}$.

Given a bag B_i: for each subset of vertices $S \subset B_i$, for each partition π of elements of S, and for all integers c, m with $c \leq m \leq n$, we count the number of induced subcomplexes of $C^-_{1,i}$ with m vertices and c connected components whose vertex set intersects bag B_i in precisely the set S, and whose connected components partition this set S according to π. Note that the count c includes connected components which are already "forgotten" (i.e., which do not meet bag B_i at all). Here, n denotes the number of vertices of C.

These lists can be trivially set up in constant time for each one-vertex leaf bag.

For each introduce bag, the list elements must be updated by either including the added vertex to the induced subcomplex or not. Note that in each step, the edges inside $C_1[B_i]$ place restrictions on which partitions of subsets $S \subset B_i$ can correspond to induced subcomplexes in $C^-_{1,i}$. The overall running time of such an introduce operation is dominated by the length of the list, which is at most quadratic in n multiplied by a function in k (for each subset and partition of vertices in B_i, up to $O(n^2)$ distinct list items can exist corresponding to different values of c, m).

For each forget bag, we remove the forgotten vertex from each list item, thereby possibly aggregating list items with equal values of S, π, c and m. This operation again has a running time dominated by the length of the lists.

Finally, whenever we join two bags, we pairwise combine list elements whenever the underlying induced subcomplexes are well-defined (i.e., whenever the subsets in the bag coincide and the partitions are compatible). This requires $O(n^4)$ time in total (the product of the two child list lengths).

After processing the root bag we are left with $O(n^2)$ list entries, labelled by the empty set, the empty partition, and the various possible values of c, m. Given the values of these list items it is now straightforward to compute $\sigma_0(C)$, as in Definition 10. Given that there are overall $O(n)$ bags to process, we have an overall running time of $O(f(k)\ n^5)$. ◀

The FPT algorithm to decide tightness for d-dimensional combinatorial manifolds M, $d \leq 4$, now consists of computing $\mu_1(M)$ by applying Theorem 16 to each vertex link, and comparing $\mu_1(M)$ to $\beta_1(M, \mathbb{F})$. Since the computation of $\mu_1(M)$ is independent of $\mathbb{F}$ and $\beta_1(M, \mathbb{F}) \leq \mu_1(M)$ for all $\mathbb{F}$, we can choose $\mathbb{F}$ to maximise β_1 and, by Theorem 9, M is tight if and only if

$$\max_{\mathbb{F}}\ \beta_1(M, \mathbb{F}) = \mu_1(M).$$

Following the proof of Theorem 16, the overall running time of this procedure is $O(\text{poly}(n) + n^6 f(k))$ for some function f.

Note that the 1-skeleton of M must always be the complete graph (or M is trivially not tight). It follows that each vertex link of M must contain $n-1$ vertices – but possibly only a linear number of edges. In this case the treewidth of the 1-skeleton of the links of M maybe very low and the above algorithm efficient. However, the 1-skeleton of the vertex link of a tight combinatorial 4-manifold M can be the complete graph, which has maximal treewidth. In this case, the algorithm presented in this section fails to give feasible running times. But in this case M must be a 3-neighbourly simply connected combinatorial 4-manifold which is tight by [27, Theorem 4.9]. Hence, worst case running times for the algorithm above are expected for combinatorial 4-manifolds of high topological complexity and with low, but strictly positive, first Betti numbers.

7 The treewidth of the dual graph of tight triangulations

It seems plausible that similar techniques as applied in Section 6 can be used to decide tightness in arbitrary dimensions. In fact, it can be shown that deciding j-tightness with respect to the field $\mathbb{F}_2$ is fixed parameter tractable in the treewidth of the dual graph of the combinatorial d-manifold. However, such an algorithm does not provide any real information about the hardness of the problem of tightness due to the following result (see [9] by the first author and Downey for a proof of a theorem implying this more special result).

▶ **Theorem 17.** *Let M be a connected combinatorial d-manifold, d fixed, with t facets and n vertices. Then either M is trivially not tight, or the treewidth k of the dual graph $\Gamma(M)$ of M satisfies*

$$k \in \Omega(t^{1/d} d^{-1}).$$

Proof. Let $\Gamma(M)$ be of treewidth k and let $T = (B, E)$ be a tree-decomposition of $\Gamma(M)$ of width k. For each bag $B_i \in B$, define B'_i to be the set of all vertices of M which are contained in some facet in B_i.

Claim: The tree $T' = (B', E)$ is a tree-decomposition of the primal graph of M.

Every vertex and every edge of M is contained in some facet of M, thus T' trivially satisfies the properties of *node coverage* and *arc coverage* of a tree-decomposition.

To verify the *subtree property*, let v be a vertex of M, let $B_v \subset B'$ be the set of bags containing v, and let $X, Y \in B_v$ be any two bags in this subset. By construction, both X and Y correspond to bags in B containing facets Δ_X and Δ_Y in the star of v in M. Since M is a combinatorial manifold, the star of v is a ball and hence there exists a sequence

$$\Delta_X = \Delta_0, \Delta_1, \ldots, \Delta_\ell = \Delta_Y$$

of facets in the star of v such that consecutive entries share a common $(d-1)$-face. Since $T = (B, E)$ is a tree-decomposition of $\Gamma(M)$ the set of bags containing a node corresponding to Δ_i, $0 \leq i \leq \ell$, is a subtree of T. Moreover, any two subtrees of consecutive entries Δ_i, Δ_{i+1} must intersect because of the edge coverage property of T. Hence, there is a path in T' from X to Y of bags containing v (i.e., bags in B_v). Since $X, Y \in B_v$ were arbitrary it follows that $T'[B_v]$ is connected and thus a subtree. This proves the claim.

Thus T' is a tree-decomposition of the 1-skeleton of M of width $(d+1)(k+1)-1$ and the treewidth of the 1-skeleton of M is at most this number.

To complete the proof, let us first assume that two vertices v and w of M are not connected by an edge. Then the induced subcomplex $M[\{v, w\}]$ is not connected but M is. It follows that M is not 0-tight and thus trivially not tight.

Now assume that any two vertices of M are connected by an edge. Hence the 1-skeleton of M is the complete graph on the set of vertices which has treewidth $n-1$, and we have $(d+1)(k+1) \geq n$. Since M is a combinatorial manifold it follows from the upper bound theorem that $t \in O(n^d)$, and thus $k \in \Omega(t^{1/d}d^{-1})$. ◀

The treewidth of the dual graph of a triangulation has recently been established as a standard tool to obtain fixed parameter tractable algorithms for problems in computational geometry and topology, see for example [10, 12, 9, 11]. Theorem 17 seems to significantly restrict the power of this approach, in particular in the simplicial setting where many typical input triangulations are 2-neighbourly, i.e., contain the complete graph in their 1-skeleton.

However, few problems in topology require us not to modify the input triangulation or even to stay in the simplicial category. In addition, there exist efficient algorithms to transform triangulations with a large number of vertices and edges into (PL-)homeomorphic triangulations with only a constant number of vertices. More precisely, there exist a polynomial time algorithm [7] due to the second author which turns any generalised triangulation of a (closed) 3-manifold M into a partial connected sum decomposition of M such that each summand is a triangulation with only one vertex. For higher dimensions there exist effective heuristics to produce crystallisations – which have a constant number of vertices – or even 1-vertex triangulations. For instance in [13] the second and fifth authors compute millions of distinct 1-vertex, 1-edge triangulations of the so-called $K3$-surface represented by a simplicial complex containing the complete graph on 16 vertices.

References

1 Aleksandr D. Alexandrov. On a class of closed surfaces. *Recueil Math. (Moscow)*, 4:69–72, 1938.

2 Bhaskar Bagchi. The mu vector, Morse inequalities and a generalized lower bound theorem for locally tame combinatorial manifolds. *European J. Combin.*, 51:69–83, 2016.

3 Bhaskar Bagchi and Basudeb Datta. On stellated spheres and a tightness criterion for combinatorial manifolds. *European J. Combin.*, 36:294–313, 2014.

4 Bhaskar Bagchi, Basudeb Datta, and Jonathan Spreer. Tight triangulations of closed 3-manifolds. *European J. Combin.*, 54:103–120, 2016.

5 Thomas F. Banchoff. Critical points and curvature for embedded polyhedral surfaces. *Amer. Math. Monthly*, 77:475–485, 1970.

6 Ulrich Brehm and Wolfgang Kühnel. 15-vertex triangulations of an 8-manifold. *Math. Ann.*, 294(1):167–193, 1992.

7 Benjamin A. Burton. A new approach to crushing 3-manifold triangulations. *Discrete Comput. Geom.*, 52(1):116–139, 2014.

8 Benjamin A. Burton, Basudeb Datta, Nitin Singh, and Jonathan Spreer. A construction principle for tight and minimal triangulations of manifolds, 2015. 27 pages, 2 figures. URL: `http://arxiv.org/abs/1511.04500`.

9 Benjamin A. Burton and Rodney G. Downey. Courcelle's theorem for triangulations. `arXiv:1403.2926 [math.GT]`, 24 pages , 7 figures, 2014.

10 Benjamin A. Burton, Thomas Lewiner, João Paixão, and Jonathan Spreer. Parameterized complexity of discrete morse theory. In *Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry*, pages 127–136. ACM SIAM, 2013.

11 Benjamin A. Burton, Clément Maria, and Jonathan Spreer. Algorithms and complexity for Turaev-Viro invariants. In *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part 1*, pages 281–293. Springer, 2015. `doi:10.1007/978-3-662-47672-7_23`.

12 Benjamin A. Burton and Jonathan Spreer. The complexity of detecting taut angle structures on triangulations. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 168–183. ACM SIAM, 2013. URL: `http://arxiv.org/abs/1207.0904`.

13 Benjamin A. Burton and Jonathan Spreer. Computationally proving triangulated 4-manifolds to be diffeomorphic, 2013. 29th ACM Symposium on Computational Geometry, Young Researchers Forum, Collections of abstracts, 2013, pages 15–16.

14 Mario Casella and Wolfgang Kühnel. A triangulated $K3$ surface with the minimum number of vertices. *Topology*, 40(4):753–772, 2001.

15 Shiing-Shen Chern and Richard K. Lashof. On the total curvature of immersed manifolds. *Amer. J. Math.*, 79:306–318, 1957.

16 Ákos Császár. A polyhedron without diagonals. *Acta Univ. Szeged. Sect. Sci. Math.*, 13:140–142, 1949.

17 Basudeb Datta and Nitin Singh. An infinite family of tight triangulations of manifolds. *J. Combin. Theory Ser. A*, 120(8):2148–2163, 2013.

18 Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*, volume 3. Springer, 1999.

19 Herbert Edelsbrunner and John L. Harer. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010.

20 Felix Effenberger. Stacked polytopes and tight triangulations of manifolds. *J. Combin. Theory Ser. A*, 118(6):1843–1862, 2011.

21 Robin Forman. Morse theory for cell complexes. *Adv. Math.*, 134(1):90–145, 1998.

22 Joel Hass and Greg Kuperberg. The complexity of recognizing the 3-sphere. *Oberwolfach Reports*, 24:1425–1426, 2012.

23 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *J. ACM*, 46(2):185–211, 1999.

24 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

25 Michael Joswig and Marc E. Pfetsch. Computing optimal Morse matchings. *SIAM J. Discrete Math.*, 20(1):11–25, 2006.

26 Ton Kloks. *Treewidth: Computations and Approximations*, volume 842. Springer, 1994.

27 Wolfgang Kühnel. *Tight polyhedral submanifolds and tight triangulations*, volume 1612 of *Lecture Notes in Math.* Springer-Verlag, Berlin, 1995.

28 Wolfgang Kühnel and Thomas F. Banchoff. The 9-vertex complex projective plane. *Math. Intelligencer*, 5(3):11–22, 1983.

29 Wolfgang Kühnel and Frank H. Lutz. A census of tight triangulations. *Period. Math. Hungar.*, 39(1-3):161–183, 1999. Discrete geometry and rigidity (Budapest, 1999).

30 Nicolaas H. Kuiper. Immersions with minimal total absolute curvature. In *Colloque Géom. Diff. Globale (Bruxelles, 1958)*, pages 75–88. Centre Belge Rech. Math., Louvain, 1959.

31 Nicolaas H. Kuiper. Morse relations for curvature and tightness. In *Proceedings of Liverpool Singularities Symposium, II*, volume 209 of *Lecture Notes in Math.*, pages 77–89, 1971.

32 Greg Kuperberg. Knottedness is in **np**, modulo GRH. *Adv. Math.*, 256:493–506, 2014.

33 Thomas Lewiner, Lopes Hélio, and Geovan Tavares. Toward optimality in discrete Morse theory. *Experiment. Math.*, 12(3):271–285, 2003.

34 John Milnor. On the relationship between the Betti numbers of a hypersurface and an integral of its Gaussian curvature (1950). In *Collected papers. Vol. 1, Geometry*, pages 15–26. Publish or Perish Inc., Houston, TX, 1994.

35 Saul Schleimer. Sphere recognition lies in NP. In *Low-dimensional and symplectic topology*, volume 82 of *Proc. Sympos. Pure Math.*, pages 183–213. Amer. Math. Soc., 2011.

36 Martin Tancer. Recognition of collapsible complexes is np-complete. `arXiv:1211.6254 [cs.CG]`, 21 pages, 13 figures, 2012.

Anchored Rectangle and Square Packings*

Kevin Balas[1], Adrian Dumitrescu[2], and Csaba D. Tóth[3]

1 **Department of Mathematics, California State University Northridge, Los Angeles, USA; and Mathematics Department, Los Angeles Mission College, Sylmar, USA**
balask@lamission.edu

2 **Department of Computer Science, University of Wisconsin – Milwaukee, USA**
dumitres@uwm.edu

3 **Department of Mathematics, California State University Northridge, Los Angeles, USA; and Department of Computer Science, Tufts University, Medford, USA**
cdtoth@acm.org

Abstract

For points $p_1, \ldots, p_n$ in the unit square $[0,1]^2$, an *anchored rectangle packing* consists of interior-disjoint axis-aligned empty rectangles $r_1, \ldots, r_n \subseteq [0,1]^2$ such that point p_i is a corner of the rectangle r_i for $i = 1, \ldots, n$ (r_i is *anchored* at p_i). We show that for every set of n points in $[0,1]^2$, there is an anchored rectangle packing of area at least $7/12 - O(1/n)$, and for every $n \in \mathbb{N}$, there are point sets for which the area of every anchored rectangle packing is at most $2/3$. The maximum area of an anchored *square* packing is always at least $5/32$ and sometimes at most $7/27$.

The above constructive lower bounds immediately yield constant-factor approximations, of $7/12 - \varepsilon$ for rectangles and $5/32$ for squares, for computing anchored packings of maximum area in $O(n \log n)$ time. We prove that a simple greedy strategy achieves a $9/47$-approximation for anchored square packings, and $1/3$ for lower-left anchored square packings. Reductions to maximum weight independent set (MWIS) yield a QPTAS and a PTAS for anchored rectangle and square packings in $n^{O(1/\varepsilon)}$ and $\exp(\mathrm{poly}(\log(n/\varepsilon)))$ time, respectively.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Rectangle packing, anchored rectangle, greedy algorithm, charging scheme, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.13

1 Introduction

Let $P = \{p_1, \ldots, p_n\}$ be a finite set of points in an axis-aligned bounding rectangle U. An *anchored rectangle packing* for P is a set of axis-aligned empty rectangles $r_1, \ldots, r_n$ that lie in U, are interior-disjoint, and p_i is one of the four corners of r_i for $i = 1, \ldots, n$; rectangle r_i is said to be *anchored* at p_i. For a given point set $P \subset U$, we wish to find the maximum total area $A(P)$ of an anchored rectangle packing of P. Since the ratio between areas is an affine invariant, we may assume that $U = [0,1]^2$. However, if we are interested in the maximum area of an *anchored square packing*, we *must* assume that $U = [0,1]^2$ (or that the aspect ratio of U is bounded from below by a constant; otherwise, with an *arbitrary* rectangle U, the guaranteed area is zero).

* This work was partially supported by the NSF awards CCF-1422311 and CCF-1423615.

© Kevin Balas, Adrian Dumitrescu, and Csaba D. Tóth;
licensed under Creative Commons License CC-BY

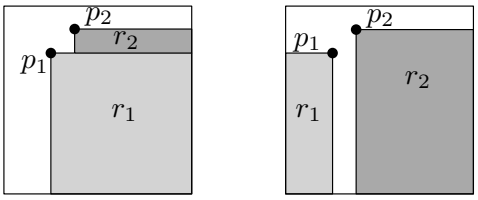


Figure 1 For $P = \{p_1, p_2\}$, with $p_1 = (\frac{1}{4}, \frac{3}{4})$ and $p_2 = (\frac{3}{8}, \frac{7}{8})$, a greedy algorithm selects rectangles of area $\frac{3}{4} \cdot \frac{3}{4} + \frac{1}{8} \cdot \frac{5}{8} = \frac{41}{64}$ (left), less than the area $\frac{1}{4} \cdot \frac{3}{4} + \frac{5}{8} \cdot \frac{7}{8} = \frac{47}{64}$ of the packing on the right.

Table 1 Table of results for the four variants studied in this paper. The last two columns refer to lower-left anchored rectangles and lower-left anchored squares, respectively.

Anchored packing with	rectangles	squares	LL-rect.	LL-sq.
Guaranteed max. area	$\frac{7}{12} - O(\frac{1}{n}) \le A(n) \le \frac{2}{3}$	$\frac{5}{32} \le A_{\text{sq}}(n) \le \frac{7}{27}$	0	0
Greedy approx. ratio	$7/12 - \varepsilon$	9/47	0.091 [20]	1/3
Approximation scheme	QPTAS	PTAS	QPTAS	PTAS

Finding the maximum area of an anchored rectangle packing of n given points is suspected but not known to be NP-hard. Balas and Tóth [8] observed that the number of distinct rectangle packings that attain the maximum area, $A(P)$, can be exponential in n. From the opposite direction, the same authors [8] proved an exponential upper bound on the number of maximum area configurations, namely $2^n C_n = \Theta(8^n/n^{3/2})$, where $C_n = \frac{1}{n+1}\binom{2n}{n} = \Theta(4^n/n^{3/2})$ is the nth Catalan number. Note that a greedy strategy may fail to find $A(P)$; see Fig. 1.

Variants and generalizations. We consider three additional variants of the problem. An *anchored square packing* is an anchored rectangle packing in which all rectangles are squares; a *lower-left anchored rectangle packing* is a rectangle packing where each point $p_i \in r_i$ is the lower-left corner of r_i; and a *lower-left anchored square packing* has both properties.

We suspect that all variants, with rectangles or with squares, are NP-hard. Here, we put forward several approximation algorithms, while it is understood that the news regarding NP-hardness can occur at any time or perhaps take some time to establish.

Contributions. Our results are summarized in Table 1. Due to space limitations, some proofs are omitted; the reader is referred to [7] for details.

- **(i)** We first deduce upper and lower bounds on the maximum area of an anchored rectangle packing of n points in $[0,1]^2$. For $n \in \mathbb{N}$, let $A(n) = \inf_{|P|=n} A(P)$. We prove that $\frac{7}{12} - O(1/n) \le A(n) \le \frac{2}{3}$ for all $n \in \mathbb{N}$ (Sections 2 and 3).
- **(ii)** Let $A_{\text{sq}}(P)$ be the maximum area of an anchored square packing for a point set P, and $A_{\text{sq}}(n) = \inf_{|P|=n} A_{\text{sq}}(P)$. We prove that $\frac{5}{32} \le A_{\text{sq}}(n) \le \frac{7}{27}$ for all n (Sections 2 and 4).
- **(iii)** The above constructive lower bounds immediately yield constant-factor approximations for computing anchored packings of maximum area ($7/12 - \varepsilon$ for rectangles and 5/32 for squares) in $O(n \log n)$ time (Sections 3 and 4). In Section 5 we show that a (natural) greedy algorithm for anchored square packings achieves a better approximation ratio, namely $9/47 = 1/5.22\ldots$, in $O(n^2)$ time. By refining some of the tools developed for this bound, in Section 6 we prove a tight bound of 1/3 for the approximation ratio of a greedy algorithm for lower-left anchored square packings.
- **(iv)** We obtain a polynomial-time approximation scheme (PTAS) for the maximum area

anchored square packing problem, and a quasi-polynomial-time approximation scheme (QPTAS) for the maximum area anchored rectangle packing problem, via a reduction to the maximum weight independent set (MWIS) problem for axis-aligned squares [16] and rectangles [2], respectively. Given n points, an $(1-\varepsilon)$-approximation for the anchored square packing of maximum area can be computed in time $n^{O(1/\varepsilon)}$; and for the anchored rectangle packing of maximum area, in time $\exp(\text{poly}(\log(n/\varepsilon)))$. Both results extend to the lower-left anchored variants; see [7, Section 7].

Motivation and related work. Packing axis-aligned rectangles in a rectangular container, albeit without anchors, is the unifying theme of several classic optimization problems. The 2D knapsack problem, strip packing, and 2D bin packing involve arranging a set of given rectangles in the most economic fashion [2, 9]. The maximum area independent set (MAIS) problem for rectangles (squares, or disks, etc.), is that of selecting a maximum area packing from a given set [3]; see classic papers such as [5, 28, 29, 30, 31] and also more recent ones [10, 11, 20] for quantitative bounds and constant approximations. These optimization problems are NP-hard, and there is a rich literature on approximation algorithms. Given an axis-parallel rectangle U in the plane containing n points, the problem of computing a maximum-area empty axis-parallel sub-rectangle contained in U is one of the oldest problems studied in computational geometry [4, 17]; the higher dimensional variant has been also studied [19]. In contrast, our problem setup is fundamentally different: the rectangles (one for each anchor) have variable sizes, but their location is constrained by the anchors.

Map labeling problems in geographic information systems (GIS) [24, 25, 27] call for choosing interior-disjoint rectangles that are incident to a given set of points in the plane. GIS applications often impose constraints on the label boxes, such as aspect ratio, minimum and maximum size, or priority weights. Most optimization problems of such variants are known to be NP-hard [21, 22, 23, 26]. In this paper, we focus on maximizing the total area of an anchored rectangle packing.

In a restricted setting where each point p_i is the lower-left corner of the rectangle r_i and $(0,0) \in P$, Allen Freedman [32, 33] conjectured almost 50 years ago that there is a lower-left anchored rectangle packing of area at least $1/2$. The current best lower bound on the area under these conditions is (about) 0.091, as established in [20]. The analogous problem of estimating the total area for lower-left anchored square packings is much easier. If P consists of the n points $(i/n, i/n)$, $i = 0, 1, \ldots, n-1$, then the total area of the n anchored squares is at most $1/n$, and so it tends to zero as n tends to infinity. A looser anchor restriction, often appearing in map labeling problems with square labels, requires the anchors to be contained in the boundaries of the squares, however the squares need to be congruent; see e.g., [34].

In the context of *covering* (as opposed to *packing*), the problem of covering a given polygon by disks with given centers such that the sum of areas of the disks is minimized has been considered in [1, 14]. In particular, covering $[0,1]^2$ with ℓ_∞-disks of given centers and minimal area as in [12, 13] is dual to the anchored square packings studied here.

Notation. Given an n-element point set P contained in $U = [0,1]^2$, denote by $\text{OPT} = \text{OPT}(P)$ a packing (of rectangles or squares, as the case may be) of maximum total area. An algorithm for a packing problem has approximation ratio α if the packing it computes, Π, satisfies $\text{area}(\Pi) \geq \alpha\, \text{area}(\text{OPT})$, for some $\alpha \leq 1$. A set of points is *in general position* if no two points have the same x- or y-coordinate. The boundary of a planar body B is denoted by ∂B, and its interior by $\text{int}(B)$.

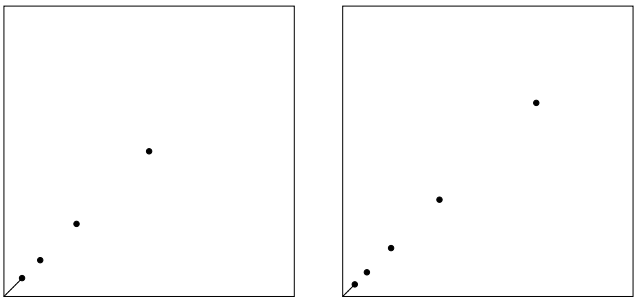


Figure 2 Left: 2/3 upper bound construction for anchored rectangles. Right: 7/27 upper bound construction for anchored squares.

2 Upper Bounds

▶ **Proposition 1.** *For every $n \in \mathbb{N}$, there exists a point set P_n such that every anchored rectangle packing for P_n has area at most $\frac{2}{3}$. Consequently, $A(n) \leq \frac{2}{3}$.*

Proof. Consider the point set $P = \{p_1, \ldots, p_n\}$, where $p_i = (x_i, y_i) = (2^{-i}, 2^{-i})$, for $i = 1, \ldots, n$; see Fig. 2 (left). Let $R = \{r_1, \ldots, r_n\}$ be an anchored rectangle packing for P. Since $p_1 = (\frac{1}{2}, \frac{1}{2})$, any rectangle anchored at p_1 has height at most $\frac{1}{2}$, width at most $\frac{1}{2}$, and hence area at most $\frac{1}{4}$.

For $i = 2, \ldots, n$, the x-coordinate of p_i, x_i, is halfway between 0 and x_{i-1}, and y_i is halfway between 0 and y_{i-1}. Consequently, if p_i is the lower-right, lower-left or upper-left corner of r_i, then $\text{area}(r_i) \leq (\frac{1}{2^i})(1 - \frac{1}{2^i}) = \frac{1}{2^i} - \frac{1}{4^i}$. If, p_i is the upper-right corner of r_i, then $\text{area}(r_i) \leq \frac{1}{4^i}$. Therefore, in all cases, we have $\text{area}(r_i) \leq \frac{1}{2^i} - \frac{1}{4^i}$. The total area of an anchored rectangle packing is bounded from above as follows:

$$A(P) \leq \sum_{i=1}^{n} \left(\frac{1}{2^i} - \frac{1}{4^i} \right) = \left(1 - \frac{1}{2^n} \right) - \frac{1}{3} \left(1 - \frac{1}{4^n} \right) = \frac{2}{3} - \frac{1}{2^n} + \frac{1}{3 \cdot 4^n} \leq \frac{2}{3}. \qquad \blacktriangleleft$$

▶ **Proposition 2.** *For every $n \in \mathbb{N}$, there exists a point set P_n such that every anchored square packing for P_n has area at most $\frac{7}{27}$. Consequently, $A_{\text{sq}}(n) \leq \frac{7}{27}$.*

Proof. Consider the point set $P = \{p_1, \ldots, p_n\}$, where $p_i = (\frac{4}{3} \cdot 2^{-i}, \frac{4}{3} \cdot 2^{-i})$, for $i = 1, \ldots, n$; see Fig. 2 (right). Let $S = \{s_1, \ldots, s_n\}$ be an anchored square packing for P. Since $p_1 = (\frac{2}{3}, \frac{2}{3})$ and $p_2 = (\frac{1}{3}, \frac{1}{3})$, any square anchored at p_1 or at p_2 has side-length at most $\frac{1}{3}$, hence area at most $\frac{1}{9}$. For $i = 3, \ldots, n$, the x-coordinate of p_i, x_i, is halfway between 0 and x_{i-1}, and y_i is halfway between 0 and y_{i-1}. Hence any square anchored at p_i has side-length at most $x_i = y_i = \frac{4}{3 \cdot 2^i}$, hence area at most $\frac{16}{9 \cdot 4^i}$. The total area of an anchored square packing is bounded from above as follows:

$$A_{\text{sq}}(P) \leq \frac{2}{9} + \frac{1}{9} \sum_{j=1}^{n-1} \frac{1}{4^j} < \frac{2}{9} + \frac{1}{9} \sum_{j=1}^{\infty} \frac{1}{4^j} = \frac{2}{9} + \frac{1}{9} \cdot \frac{1}{3} = \frac{7}{27}. \qquad \blacktriangleleft$$

▶ Remark. Stronger upper bounds hold for small n, e.g., $n \in \{1, 2\}$. Specifically, $A(1) = A_{\text{sq}}(1) = 1/4$ attained for the center $(\frac{1}{2}, \frac{1}{2}) \in [0, 1]^2$, and $A(2) = 4/9$ and $A_{\text{sq}}(2) = 2/9$ attained for $P = \{(\frac{1}{3}, \frac{1}{3}), (\frac{2}{3}, \frac{2}{3})\}$.

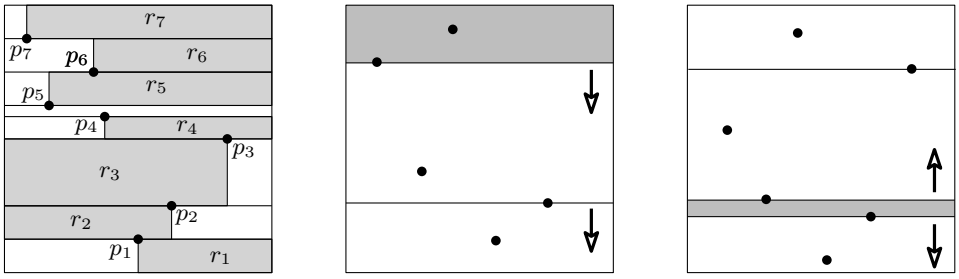


Figure 3 Left: Horizontal strips with anchored rectangles for 7 points. Middle: an example of the partition for odd n (here $n = 5$). Right: an example of the partition for even n (here $n = 6$). The strip that is discarded is shaded in the drawing.

3 Lower Bound for Anchored Rectangle Packings

In this section, we prove that for every set P of n points in $[0,1]^2$, we have $A(P) \geq \frac{7n-2}{12(n+1)}$. Our proof is constructive; we give a divide & conquer algorithm that partitions U into horizontal strips and finds n anchored rectangles of total area bounded from below as required. We start with a weaker lower bound, of about 1/2, and then sharpen the argument to establish the main result of this section, a lower bound of about 7/12.

▶ **Proposition 3.** *For every set of n points in the unit square $[0,1]^2$, an anchored rectangle packing of area at least $\frac{n}{2(n+1)}$ can be computed in $O(n \log n)$ time.*

Proof. Let $P = \{p_1, \ldots, p_n\}$ be a set of points in the unit square $[0,1]^2$ sorted by their y-coordinates. Draw a horizontal line through each point in P; see Fig. 3 (left). These lines divide $[0,1]^2$ into $n+1$ horizontal strips. A strip can have zero width if two points have the same x-coordinate. We leave a narrowest strip empty and assign the remaining strips to the n points such that each rectangle above (resp., below) the chosen narrowest strip is assigned to a point of P on its bottom (resp., top) edge. For each point divide the corresponding strip into two rectangles with a vertical line through the point. Assign the larger of the two rectangles to the point.

The area of narrowest strip is at most $\frac{1}{n+1}$. The rectangle in each of the remaining n strips covers at least $\frac{1}{2}$ of the strip. This yields a total area of at least $\frac{n}{2(n+1)}$. ◀

A key observation allowing a stronger lower bound is that for *two* points in a horizontal strip, one can always pack two anchored rectangles in the strip that cover strictly more than half the area of the strip. Specifically, we have the following easy-looking statement with 2 points in a rectangle (however, we do not have an easy proof!); details are in [7].

▶ **Lemma 4.** *Let $P = \{p_1, p_2\}$ be two points in an axis-parallel rectangle R such that p_1 lies on the bottom side of R. Then there exist two empty rectangles in R anchored at the two points of total area at least $\frac{7}{12}\,\mathrm{area}(R)$, and this bound is the best possible.*

In order to partition the unit square into strips that contain two points, one on the boundary, we need to use parity arguments. Let P be a set of n points in $[0,1]^2$ with y-coordinates $0 \leq y_1 \leq y_2 \leq \cdots \leq y_n \leq 1$. Set $y_0 = 0$ and $y_{n+1} = 1$. For $i = 1, \ldots, n+1$, put $h_i = y_i - y_{i-1}$, namely h_i is the ith vertical gap. Obviously, we have

$$h_i \geq 0 \text{ for all } i = 1, \ldots, n+1, \text{ and } \sum_{i=1}^{n+1} h_i = 1. \tag{1}$$

Parity considerations are handled by the following lemma.

▶ **Lemma 5.**

(i) *If n is odd, at least one of the following $(n+1)/2$ inequalities is satisfied:*

$$h_i + h_{i+1} \leq \frac{2}{n+1}, \text{ for (odd) } i = 1, 3, \ldots, n-2, n. \tag{2}$$

(ii) *If n is even, at least one of the following $n+2$ inequalities is satisfied:*

$$h_1 \leq \frac{2}{n+2}, \qquad h_{n+1} \leq \frac{2}{n+2}, \qquad h_i + h_{i+1} \leq \frac{2}{n+2}, \text{ for } i = 1, 2, \ldots, n. \tag{3}$$

Proof. Assume first that n is odd. Put $a = \frac{2}{n+1}$ and assume that none of the inequalities in (2) is satisfied. Summation would yield $\sum_{i=1}^{n+1} h_i > \frac{n+1}{2}\, a = 1$, an obvious contradiction.

Assume now that n is even. Put $a = \frac{2}{n+2}$ and assume that none of the inequalities in (3) is satisfied. Summation would yield $2\sum_{i=1}^{n+1} h_i > (n+2)a = 2$, again a contradiction. ◀

We can now prove the main result of this section.

▶ **Theorem 6.** *For every set of n points in the unit square $[0,1]^2$, an anchored rectangle packing of area at least $\frac{7(n-1)}{12(n+1)}$ when n is odd and $\frac{7n}{12(n+2)}$ when n is even can be computed in $O(n \log n)$ time.*

Proof. Let $P = \{p_1, \ldots, p_n\}$ be a set of points in the unit square $[0,1]^2$ sorted by their y-coordinates with the notation introduced above. By Lemma 5, we find a horizontal strip corresponding to one of the inequalities that is satisfied.

Assume first that n is odd. Draw a horizontal line through each point in $p_j \in P$, for j even, as shown in Fig. 3. These lines divide $[0,1]^2$ into $\frac{n+1}{2}$ rectangles (horizontal strips). Suppose now that the satisfied inequality is $h_i + h_{i+1} \leq \frac{2}{n+2}$ for some odd i. Then we leave a rectangle between $y = y_{i-1}$ and $y = y_{i+1}$ empty, i.e., r_i is a rectangle of area 0. For the remaining rectangles, we assign two consecutive points of P such that each strip above $y = y_{i+1}$ (resp., below $y = y_{i-1}$) is assigned a point of P on its bottom (resp., top) edge. Within each rectangle R, we can choose two anchored rectangles of total area at least $\frac{7}{12}\,\text{area}(R)$ by Lemma 4. By Lemma 5(i), the area of the narrowest strip is at most $\frac{2}{n+1}$. Consequently, the area of the anchored rectangles is at least $\frac{7}{12}(1 - \frac{2}{n+1}) = \frac{7(n-1)}{12(n+1)}$.

Assume now that n is even. If the selected horizontal strip corresponds to the inequality $h_1 \leq \frac{2}{n+2}$, then divide the unit square along the lines $y = y_i$, where i is odd. We leave the strip of height h_1 empty, and assign pairs of points to all remaining strips such that one of the two points lies on the top edge of the strip. We proceed analogously if the inequality $h_{n+1} \leq \frac{2}{n+2}$ is satisfied. Suppose now that the satisfied inequality is $h_i + h_{i+1} \leq \frac{2}{n+2}$. If i is odd, we leave the strip of height $h_i \leq \frac{2}{n+2}$ (between $y = y_{i-1}$ and $y = y_i$) empty; if i is even, we leave the strip of height $h_{i+1} \leq \frac{2}{n+2}$ (between $y = y_i$ and $y = y_{i+1}$) empty. Above and below the empty strip, we can form a total of $n/2$ strips, each containing two points of P, with one of the two points lying on the bottom or the top edge of the strip. By Lemma 5(i), the area of the empty strip is at most $\frac{2}{n+2}$. Consequently, the area of the anchored rectangles is at least $\frac{7}{12}(1 - \frac{2}{n+2}) = \frac{7n}{12(n+2)}$, as claimed. ◀

4 Lower Bound for Anchored Square Packings

Given a set $P \subset U = [0,1]^2$ of n points, we show there is an anchored square packing of large total area. The proof we present is constructive; we give a recursive partitioning algorithm

(as an inductive argument) based on a quadtree subdivision of U that finds n anchored squares of total area at least $5/32$. We need the following easy fact:

▶ **Observation 7.** *Let $u, v \subseteq U$ be two congruent axis-aligned interior-disjoint squares sharing a common edge such that $u \cap P \neq \emptyset$ and $\mathrm{int}(v) \cap P = \emptyset$. Then $u \cup v$ contains an anchored empty square whose area is at least $\mathrm{area}(u)/4$.*

Proof. Let a denote the side-length of u (or v). Assume that v lies right of u. Let $p \in P$ be the rightmost point in u. If p lies in the lower half-rectangle of u then the square of side-length $a/2$ whose lower-left anchor is p is empty and has area $a^2/4$. Similarly, if p lies in the higher half-rectangle of u then the square of side-length $a/2$ whose upper-left anchor is p is empty and has area $a^2/4$. ◀

▶ **Theorem 8.** *For every set of n points in $U = [0,1]^2$, where $n \geq 1$, an anchored square packing of total area at least $5/32$ can be computed in $O(n \log n)$ time.*

Proof. We first derive a lower bound of $1/8$ and then sharpen it to $5/32$. We proceed by induction on the number of points n contained in U and assigned to U; during the subdivision process, the rôle of U is taken by any subdivision square. If all points in P lie on U's boundary, ∂U, pick one arbitrarily, say, $(x, 0)$ with $x \leq 1/2$. (All assumptions in the proof are made without loss of generality.) Then the square $[x, x + 1/2] \times [0, 1/2]$ is empty and its area is $1/4 > 5/32$, as required. Otherwise, discard the points in $P \cap \partial U$ and continue on the remaining points.

If $n = 1$, we can assume that $x(p), y(p) \leq 1/2$. Then the square of side-length $1/2$ whose lower-left anchor is p is empty and contained in U, as desired; hence $A_{\mathrm{sq}}(P) \geq 1/4$. If $n = 2$ let x_1, x_2, x_3 be the widths of the 3 vertical strips determined by the two points, where $x_1 + x_2 + x_3 = 1$. We can assume that $0 \leq x_1 \leq x_2 \leq x_3$; then there are two anchored empty squares with total area at least $x_2^2 + x_3^2 \geq 2/9 > 5/32$, as required.

Assume now that $n \geq 3$. Subdivide U into four congruent squares, $U_1, \ldots, U_4$, labeled counterclockwise around the center of U according to the quadrant containing the square. Partition P into four subsets $P_1, \ldots, P_4$ such that $P_i \subset U_i$ for $i = 1, \ldots, 4$, with ties broken arbitrarily. We next derive the lower bound $A_{\mathrm{sq}}(P) \geq 1/8$. We distinguish 4 cases, depending on the number of empty sets P_i.

Case 1: precisely one of $P_1, \ldots, P_4$ is empty. We can assume that $P_1 = \emptyset$. By Observation 7, $U_1 \cup U_2$ contains an empty square anchored at a point in $P_1 \cup P_2$ of area at least $\mathrm{area}(U_1)/4 = 1/16$. By induction, U_3 and U_4 each contain an anchored square packing of area at least $c \cdot \mathrm{area}(U_3) = c \cdot \mathrm{area}(U_4)$. Overall, we have $A_{\mathrm{sq}}(P) \geq 2c/4 + 1/16 \geq c$, which holds for $c \geq 1/8$.

Case 2: precisely two of $P_1, \ldots, P_4$ are empty. We can assume that the pairs $\{P_1, P_2\}$ and $\{P_3, P_4\}$ each consist of one empty and one nonempty set. By Observation 7, $U_1 \cup U_2$ and $U_3 \cup U_4$, respectively, contain a square anchored at a point in $P_1 \cup P_2$ and $P_3 \cup P_4$ of area at least $\mathrm{area}(U_1)/4 = 1/16$. Hence $A_{\mathrm{sq}}(P) \geq 2 \cdot \frac{1}{16} = 1/8$.

Case 3: precisely three of $P_1, \ldots, P_4$ are empty. We can assume that $P_3 \neq \emptyset$. Let $(a, b) \in P$ be a maximal point in the product order (e.g., the sum of coordinates is maximum). Then $s = [a, a + \frac{1}{2}] \times [b, b + \frac{1}{2}]$ is a square anchored at (a, b), $s \subseteq [0,1]^2$ since $(a, b) \in U_3$, and $\mathrm{int}(s) \cap P = \emptyset$. Hence $A_{\mathrm{sq}}(P) \geq \mathrm{area}(s) = 1/4$.

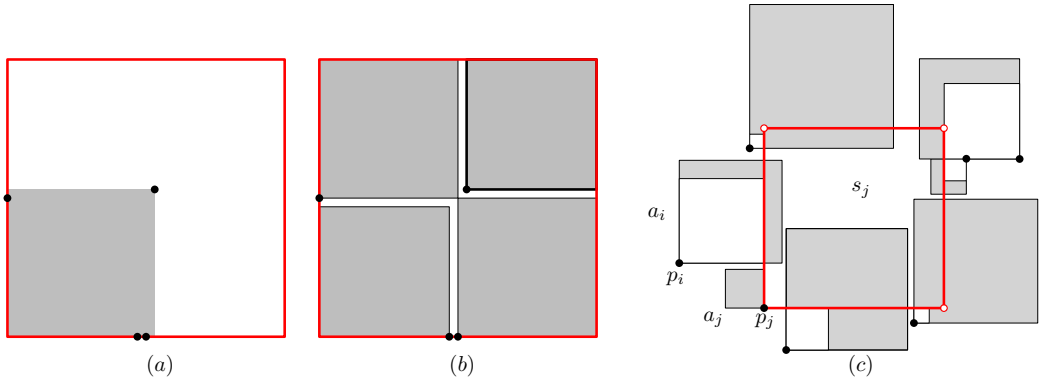


Figure 4 (a–b) A 1/4 upper bound for the approximation ratio of Algorithm 9. (c) Charging scheme for Algorithm 9. Without loss of generality, the figure illustrates the case when s_j is a lower-left anchored square.

Case 4: $P_i \neq \emptyset$ for every $i = 1, \ldots, 4$. Note that $A_{\text{sq}}(P) \geq \sum_{i=1}^{4} A_{\text{sq}}(P_i)$, where the squares anchored at P_i are restricted to U_i. Induction completes the proof in this case.

In all four cases, we have verified that $A_{\text{sq}}(P) \geq 1/8$, as claimed. The inductive proof can be turned into a recursive algorithm based on a quadtree subdivision of the n points, which can be computed in $O(n \log n)$ time [6, 18]. In addition, computing an extreme point (with regard to a specified axis-direction) in any subsquare over all needed such calls can be executed within the same time bound. Note that the bound in Case 3 is at least 5/32 and Case 4 is inductive. Sharpening the analysis of Cases 1 and 2 yields an improved bound 5/32; since $5/32 < 1/4$, the value 5/32 is not a bottleneck for Cases 3 and 4. Details are given in [7]; the running time remains $O(n \log n)$. ◀

5 Constant-Factor Approximations for Anchored Square Packings

In this section we investigate better approximations for square packings. Given a finite point set $P \subset [0,1]^2$, perhaps the most natural greedy strategy for computing an anchored square packing of large area is the following.

▶ **Algorithm 9.** Set $Q = P$ and $S = \emptyset$. While $Q \neq \emptyset$, repeat the following. For each point $q \in Q$, compute a *candidate* square $s(q)$ such that (i) $s(q) \subseteq [0,1]^2$ is *anchored* at q, (ii) $s(q)$ is empty of points from P in its interior, (iii) $s(q)$ is interior-disjoint from all squares in S, and (iv) $s(q)$ has maximum area. Then choose a largest candidate square $s(q)$, and a corresponding point q, and set $Q \leftarrow Q \setminus \{q\}$ and $S \leftarrow S \cup \{s(q)\}$. When $Q = \emptyset$, return the set of squares S.

▶ Remark. Let ρ_9 denote the approximation ratio of Algorithm 9, if it exists. The construction in Fig. 4(a–b) shows that $\rho_9 \leq 1/4$. For a small $\varepsilon > 0$, consider the point set $P = \{p_1, \ldots, p_n\}$, where $p_1 = (1/2 + \varepsilon, 1/2 + \varepsilon)$, $p_2 = (1/2, 0)$, $p_3 = (0, 1/2)$, and the rest of the points lie on the lower side of U in the vicinity of p_2, i.e., $x_i \in (1/2 - \varepsilon/2, 1/2 + \varepsilon/2)$ and $y_i = 0$ for $i = 4, \ldots, n$. The packing generated by Algorithm 9 consists of a single square of area $(1/2 + \varepsilon)^2$, as shown in Fig. 4(a), while the packing in Fig. 4(b) has an area larger than $1 - \varepsilon$. By letting ε be arbitrarily small, we deduce that $\rho_9 \leq 1/4$.

We first show that Algorithm 9 achieves a ratio of 1/6 (Theorem 12) using a charging scheme that compares the greedy packing with an optimal packing. We then refine our analysis and sharpen the approximation ratio to $\frac{9}{47} = 1/5.22\ldots$ (Theorem 17).

Charging scheme for the analysis of Algorithm 9. Label the points in $P = \{p_1, \ldots, p_n\}$ and the squares in $S = \{s_1, \ldots, s_n\}$ in the order in which they are processed by Algorithm 9 with $q = p_i$ and $s_i = s(q)$. Let $G = \sum_{i=1}^{n} \text{area}(s_i)$ be the area of the greedy packing, and let OPT denote an optimal packing with $A = \text{area}(\text{OPT}) = \sum_{i=1}^{n} \text{area}(a_i)$, where a_i is the square anchored at p_i.

We employ a charging scheme, where we distribute the area of every optimal square a_i with $\text{area}(a_i) > 0$ among some greedy squares; and then show that the total area of the optimal squares charged to each greedy square s_j is at most $6\,\text{area}(s_j)$ for all $j = 1, \ldots, n$. (Degenerate optimal squares, i.e., those with $\text{area}(a_i) = 0$ do not need to be charged). For each step $j = 1, \ldots, n$ of Algorithm 9, we shrink some of the squares $a_1, \ldots, a_n$, and charge the area-decrease to the greedy square s_j. By the end (after the nth step), each of the squares $a_1, \ldots, a_n$ will be reduced to a single point.

Specifically in step j, Algorithm 9 chooses a square s_j, and: (1) we shrink square a_j to a single point; and (2) we shrink every square a_i, $i > j$ that intersects s_j in its interior until it no longer does so. This procedure ensures that no square a_i, with $i < j$, intersects s_j in its interior in step j. Refer to Fig. 4(c). Observe three important properties of the above iterative process:

- **(i)** After step j, the squares $s_1, \ldots, s_j, a_1, \ldots, a_n$ have pairwise disjoint interiors.
- **(ii)** After step j, we have $\text{area}(a_j) = 0$ (since a_j was shrunk to a single point).
- **(iii)** At the beginning of step j, if a_i intersects s_j in its interior (and so $i \geq j$), then $\text{area}(a_i) \leq \text{area}(s_j)$ since s_j is feasible for p_i when a_j is selected by Algorithm 9 due to the greedy choice.

▶ **Lemma 10.** *Suppose there exists a constant $\varrho \geq 1$ such that for every $j = 1, \ldots, n$, square s_j receives a charge of at most $\varrho\,\text{area}(s_j)$. Then Algorithm 9 computes an anchored square packing whose area G is at least $1/\varrho$ times the optimal.*

Proof. Overall, each square s_j receives a charge of at most $\varrho\,\text{area}(s_j)$ from the squares in an optimal solution. Consequently, $A = \text{area}(\text{OPT}) = \sum_{i=1}^{n} \text{area}(a_i) \leq \varrho \sum_{j=1}^{n} \text{area}(s_j) = \varrho\, G$, and thus $G \geq A/\varrho$, as claimed. ◀

In the remainder of this section, we bound the charge received by one square s_j, for $j = 1, \ldots, n$. We distinguish two types of squares a_i, $i > j$, whose area is reduced by s_j:

- $\mathcal{Q}_1 = \{a_i : i > j$, the area of a_i is reduced by s_j, and $\text{int}(a_i)$ contains no corner of $s_j\}$,
- $\mathcal{Q}_2 = \{a_i : i > j$, the area of a_i is reduced by s_j, and $\text{int}(a_i)$ contains a corner of $s_j\}$.

It is clear that if the insertion of s_j reduces the area of a_i, $i > j$, then a_i is in either $\mathcal{Q}_1$ or $\mathcal{Q}_2$. Note that the area of a_j is also reduced to 0, but it is in neither $\mathcal{Q}_1$ nor $\mathcal{Q}_2$.

▶ **Lemma 11.** *Each square s_j receives a charge of at most $6\,\text{area}(s_j)$.*

Proof. Consider the squares in $\mathcal{Q}_1$. Assume that a_i intersects the interior of s_j, and it is shrunk to a_i'. The area-decrease $a_i \setminus a_i'$ is an L-shaped region, at least half of which lies inside s_j; see Fig. 4. By property (i), the L-shaped regions are pairwise interior-disjoint; and hence the sum of their areas is at most $2\,\text{area}(s_j)$. Consequently, the area-decrease caused by s_j in squares in $\mathcal{Q}_1$ is at most $2\,\text{area}(s_j)$.

Consider the squares in $\mathcal{Q}_2$. There are at most three squares a_i, $i > j$, that can contain a corner of s_j since the anchor of s_j is not contained in the interior of any square a_i. Since the area of each square in $\mathcal{Q}_2$ is at most $\text{area}(s_j)$ by property (iii), the area decrease is at most $3\,\text{area}(s_j)$, and so is the charge received by s_j from squares.

Finally, $\text{area}(a_j) \leq \text{area}(s_j)$ by property (iii), and this is also charged to s_j. Overall s_j receives a charge of at most $6\,\text{area}(s_j)$. ◀

The combination of Lemmas 10 and 11 readily implies the following.

▶ **Theorem 12.** *Algorithm 9 computes an anchored square packing whose area is at least* $1/6$ *times the optimal.*

Refined analysis of the charging scheme. We next improve the upper bound for the charge received by s_j; we assume for convenience that $s_j = U = [0,1]^2$. For the analysis, we use only a few properties of the optimal solution. Specifically, assume that $a_1, \ldots, a_m$ are interior-disjoint squares such that each a_i: (a) intersects the interior of s_j; (b) has at least a corner in the exterior of s_j; (c) does not contain $(0,0)$ in its interior; and (d) $\text{area}(a_i) \leq \text{area}(s_j)$.

The intersection of any square a_i with ∂U is a polygonal line on the boundary ∂U, consisting of one or two segments. Since the squares a_i form a packing, these intersections are interior-disjoint.

Let $\Delta_1(x)$ denote the maximum area-decrease of a set of squares a_i in $\mathcal{Q}_1$, whose intersections with ∂U have total length x. Similarly, let $\Delta_2(x)$ denote the maximum area-decrease of a set of squares a_i in $\mathcal{Q}_2$, whose intersections with ∂U have total length x. By adding suitable squares to $\mathcal{Q}_1$, we can assume that $4-x$ is the total length of the intersections $a_i \cap \partial U$ over squares in $\mathcal{Q}_2$ (i.e., the squares in $\mathcal{Q}_1 \cup \mathcal{Q}_2$ cover the entire boundary of U). Consequently, the maximum total area-decrease is given by

$$\Delta(x) = \Delta_1(x) + \Delta_2(4-x), \text{ and } \Delta = \sup_{0 \leq x \leq 4} \Delta(x). \tag{4}$$

We now derive upper bounds for $\Delta_1(x)$ and $\Delta_2(x)$ independently, and then combine these bounds to optimize $\Delta(x)$. Since the total perimeter of U is 4, the domain of $\Delta(x)$ is $0 \leq x \leq 4$.

▶ **Lemma 13.** *The following inequalities hold:*

$$\Delta_1(x) \leq 2, \tag{5}$$

$$\Delta_1(x) \leq x, \tag{6}$$

$$\Delta_1(x) \leq 1 + (x-1)^2, \text{ for } 1 \leq x \leq 2, \tag{7}$$

$$\Delta_1(x) \leq 1 + \frac{\lfloor x \rfloor}{4} + \frac{(x - \lfloor x \rfloor)^2}{4}, \text{ for } 0 \leq x \leq 4. \tag{8}$$

Proof. Inequality (5) was explained in the proof of Theorem 12. Inequalities (6) and (7) follow from the fact that the side-length of each square a_i is at most 1 and from the fact that the area-decrease is at most the area (of respective squares); in addition, we use the inequality $\sum x_i^2 \leq (\sum x_i)^2$, for $x_i \geq 0$ and $\sum x_i \leq 1$, and the inequality $x^2 + y^2 \leq 1 + (x+y-1)^2$, for $0 \leq x, y \leq 1$, and $x + y > 1$. Write

$$\Delta_1(x) = \Delta_1^{\text{in}}(x) + \Delta_1^{\text{out}}(x), \tag{9}$$

where $\Delta_1^{\text{in}}(x)$ and $\Delta_1^{\text{out}}(x)$ denote the maximum area-decrease contained in U and the complement of U, respectively, of a set of squares in $\mathcal{Q}_1$ whose intersections with ∂U have total length x, where $0 \leq x \leq 4$. Obviously, $\Delta_1^{\text{in}}(x) \leq \text{area}(U) = 1$. We next show that

$$\Delta_1^{\text{out}}(x) \leq \frac{\lfloor x \rfloor}{4} + \frac{(x - \lfloor x \rfloor)^2}{4},$$

and thereby establish inequality (8).

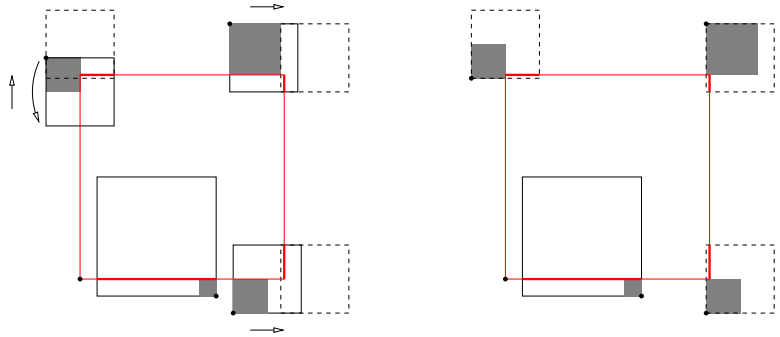


Figure 5 Bounding the area-decrease; moving the squares in $\mathcal{Q}_2$ into canonical position. The parts of ∂U covered by each square (after transformations) are drawn in thick red lines.

Consider a square a_i of side-length $x_i \leq 1$ in $\mathcal{Q}_1$. Let z_i denote the length of the shorter side of the rectangle $a_i \setminus U$. The area-decrease outside U equals $x_i z_i - z_i^2$ and so it is bounded from above by $x_i^2/4$ (equality is attained when $z_i = x_i/2$).

Consequently,

$$\Delta_1^{\text{out}}(x) \leq \sup \sum_{\substack{0 \leq x_i \leq 1 \\ \sum x_i = x}} \frac{x_i^2}{4} = \frac{\lfloor x \rfloor}{4} + \frac{(x - \lfloor x \rfloor)^2}{4},$$

where the last equality follows from a standard weight-shifting argument, and equality is attained when x is subdivided into $\lfloor x \rfloor$ unit length intervals and a remaining shorter interval of length $x - \lfloor x \rfloor$. ◀

Let $k \leq 3$ be the number of squares a_i in $\mathcal{Q}_2$, where $i > j$. We can assume that exactly 3 squares a_i, with $i > j$, are in $\mathcal{Q}_2$, one for each corner except the lower-left anchor corner of U, that is, $k = 3$; otherwise the proof of Lemma 11 already yields an approximation ratio of $1/5$. Clearly, we have $\Delta_2(x) \leq k \leq 3$, for any x.

We first bring the squares in $\mathcal{Q}_2$ into *canonical position*: x monotonically decreases, $\Delta(x)$ does not decrease, and properties (a–d) listed earlier are maintained. Specifically, we transform each square $a_i \in \mathcal{Q}_2$ as follows (refer to Fig. 5):

- Move the anchor of a_i to another corner if necessary so that one of its coordinates is contained in the interval $(0, 1)$;
- translate a_i horizontally or vertically so that $a_i \cap U$ decreases to a skinny rectangle of width ε, for some small $\varepsilon > 0$.

▶ **Lemma 14.** *The following inequality holds:*

$$\Delta_2(x) \leq 2x - \frac{x^2}{3}, \text{ for } 0 \leq x \leq 4. \tag{10}$$

Proof. Assume that the squares in $\mathcal{Q}_2$ are in canonical position. Let y_i denote the side-length of a_i, let x_i denote the length of the longer side of the rectangle $a_i \cap U$ and z_i denote the length of the shorter side of the rectangle $a_i \setminus U$, $i = 1, 2, 3$. Since the squares in $\mathcal{Q}_2$ are in canonical position, we have $x_i + z_i = y_i \leq 1$, for $i = 1, 2, 3$. We also have $\sum_{i=1}^{3} x_i = x - O(\varepsilon)$. Letting $\varepsilon \to 0$, we have $\sum_{i=1}^{3} x_i = x$.

$$\Delta_2(x) = \sup_{x_i+z_i=y_i\le 1} \sum_{i=1}^{3}(y_i^2 - z_i^2) = \sup_{0\le x_1,x_2,x_3\le 1} \sum_{i=1}^{3}(1-(1-x_i^2))$$
$$= \sup_{0\le x_1,x_2,x_3\le 1} \sum_{i=1}^{3}(2x_i - x_i^2) = 2x - \inf_{0\le x_1,x_2,x_3\le 1} \sum_{i=1}^{3} x_i^2 = 2x - \frac{x^2}{3}. \quad \blacktriangleleft$$

Observe that the inequality $\Delta_2(x) \le 3$, for every $0 \le x \le 4$, is implied by (10). Putting together the upper bounds in Lemmas 13 and 14 yields Lemma 15 (refer to [7] for the proof):

▶ **Lemma 15.** *The following inequality holds:*

$$\Delta \le \frac{38}{9}. \tag{11}$$

From the opposite direction, $\Delta \ge 4$ holds even in a geometric setting, i.e., as implied by several constructions with squares.

▶ **Lemma 16.** *Each square s_j receives a charge of at most $\frac{47}{9}\operatorname{area}(s_j)$.*

Proof. By Lemma 15, the area-decrease is at most $38/9\operatorname{area}(s_j)$, and so is the charge received by s_j from squares in $\mathcal{Q}_1$ and from squares in $\mathcal{Q}_2$ with the exception of the case $i = j$. Adding this last charge yields a total charge of at most $\left(1 + \frac{38}{9}\right)\operatorname{area}(s_j) = \frac{47}{9}\operatorname{area}(s_j)$. ◀

The combination of Lemmas 10 and 16 now yields the following.

▶ **Theorem 17.** *Algorithm 9 computes an anchored square packing whose area is at least $9/47$ times the optimal.*

6 Constant-Factor Approximations for Lower-Left Anchored Square Packings

The following greedy algorithm, analogous to Algorithm 9, constructs a lower-left anchored square packing, given a finite point set $P \subset [0,1]^2$.

▶ **Algorithm 18.** Set $Q = P$ and $S = \emptyset$. While $Q \neq \emptyset$, repeat the following. For each point $q \in Q$, compute a *candidate* square $s(q)$ such that (i) $s(q) \subseteq [0,1]^2$ has q as its *lower-left* anchor, (ii) $s(q)$ is empty of points from P in its interior, (iii) $s(q)$ is interior-disjoint from all squares in S, and (iv) $s(q)$ has maximum area. Then choose a largest candidate square $s(q)$, and a corresponding point q, and set $Q \leftarrow Q \setminus \{q\}$ and $S \leftarrow S \cup \{s(q)\}$. When $Q = \emptyset$, return the set of squares S.

▶ Remark. Let ρ_{18} denote the approximation ratio of Algorithm 18. The construction in Fig. 6 shows that $\rho_{18} \le 1/3$. Specifically, for $\varepsilon > 0$, with $\varepsilon^{-1} \in \mathbb{N}$, consider the point set $P = \{(\varepsilon,\varepsilon), (0,\frac{1}{2}), (\frac{1}{2},0)\} \cup \{(\frac{1}{2}+k\varepsilon, \frac{1}{2}+k\varepsilon) : k = 1,\ldots,1/(2\varepsilon)-1\}$. Then the area of the packing in Fig. 6 (right) is $\frac{3}{4} - O(\varepsilon)$, but Algorithm 18 returns the packing shown in Fig. 6 (left) of area $\frac{1}{4} + O(\varepsilon)$.

We next demonstrate that Algorithm 18 achieves approximation ratio $1/3$. According to the above example, this is the best possible for this algorithm.

▶ **Theorem 19.** *Algorithm 18 computes a lower-left anchored square packing whose area is at least $1/3$ times the optimal.*

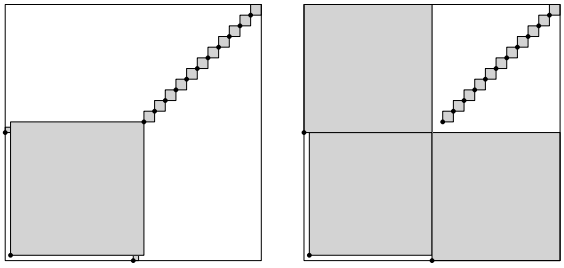


Figure 6 A 1/3 upper bound for the approximation ratio of Algorithm 18.

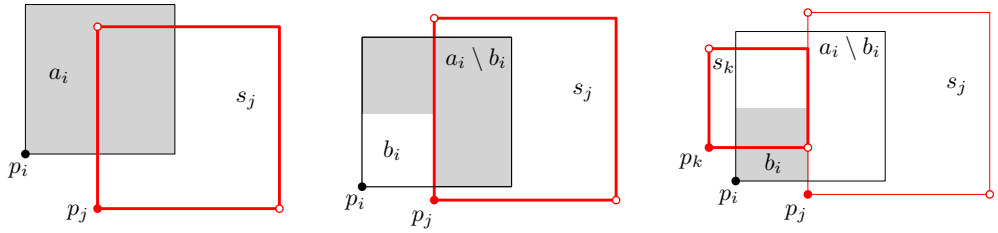


Figure 7 Left: a_i contains the upper-left corner of s_j; and area(a_i) is charged to s_j. Middle and Right: a_i contains no corner of s_j, but it contains the lower-right corner of s_k. Then area($a_i \setminus b_i$) is charged to s_j and area(b_i) is charged to s_k.

Proof. Label the points in $P = \{p_1, \ldots, p_n\}$ and the squares in $S = \{s_1, \ldots, s_n\}$ in the order in which they are processed by Algorithm 18 with $q = p_i$ and $s_i = s(q)$. Let $G = \sum_{i=1}^{n} \text{area}(s_i)$ be the area of the greedy packing, and let OPT denote an optimal packing with $A = \text{area(OPT)} = \sum_{i=1}^{n} \text{area}(a_i)$, where a_i is the square anchored at p_i.

We charge the area of every optimal square a_i to one or two greedy squares s_ℓ; and then show that the total area charged to s_ℓ is at most $3\,\text{area}(s_\ell)$ for all $\ell = 1, \ldots, n$. Consider a square a_i, $1 \le i \le n$, with area(a_i) > 0. Let $j = j(i)$ be the minimum index such that s_j intersects the interior of a_i. Let b_i denote the candidate square associated to p_i in step $j + 1$ of Algorithm 18. Note that $b_i \subset a_i$, thus area(b_i) $<$ area(a_i). If area(b_i) > 0, then let $k = k(i)$ be the minimum index such that s_k intersects the interior of b_i.

We can now describe our *charging scheme*: If a_i contains the upper-left or lower-right corner of s_j, then charge area(a_i) to s_j (Fig. 7, left). Otherwise, charge area($a_i \setminus b_i$) to s_j, and charge area(b_i) to s_k (Fig. 7, middle-right).

We first argue that the charging scheme is well-defined, and the total area of a_i is charged to one or two squares (s_j and possibly s_k). Indeed, if no square s_ℓ, $\ell < i$, intersects the interior of a_i, then $a_i \subseteq s_i$, and $j(i) = i$; and if $a_i \not\subseteq s_j$ and no square s_ℓ, $j < \ell < i$, intersects the interior of b_i, then $b_i \subseteq s_i$ and $k(i) = i$.

Note that if area(a_i) is charged to s_j, then area(a_i) $\le$ area(s_j). Indeed, if area(a_i) $>$ area(s_j), then a_i is entirely free at step j, so Algorithm 18 would choose a square at least as large as a_i instead of s_j, which is a contradiction. Analogously, if area(b_i) is charged to s_k, then area(b_i) $\le$ area(s_k). Moreover, if area(b_i) is charged to s_k, then the upper-left or lower-right corner of s_k is on the boundary of b_i, and so this corner is contained in a_i; refer to Fig. 7 (right).

Fix $\ell \in \{1, \ldots, n\}$. We show that the total area charged to s_ℓ is at most $3\,\text{area}(s_\ell)$. If a square a_i, $i = 1, \ldots, n$, sends a positive charge to s_ℓ, then $\ell = j(i)$ or $\ell = k(i)$. We distinguish two types of squares a_i that send a positive charge to s_ℓ; refer to Fig. 8:

T1 a_i contains the upper-left or lower-right corner of s_ℓ in its interior.

T2 a_i contains neither the upper-left nor the lower-right corner of s_ℓ.

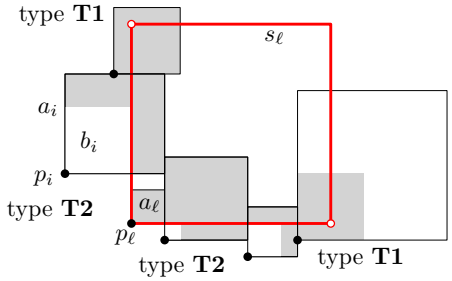


Figure 8 The shaded areas are charged to square s_ℓ.

Since OPT is a packing, at most one optimal square contains each corner of s_ℓ. Consequently, there is at most two squares a_i of type **T1**. Since $\text{area}(a_i) \leq \text{area}(s_\ell)$, the charge received from the squares of type **T1** is at most $2\,\text{area}(s_\ell)$.

By [7, Lemma 9], s_ℓ receives a charge of at most $\text{area}(s_\ell)$ from squares of type **T2**. It follows that each s_ℓ received a charge of at most $3\,\text{area}(s_\ell)$. Consequently,

$$A = \text{area}(\text{OPT}) = \sum_{i=1}^{n} \text{area}(a_i) \leq 3 \sum_{\ell=1}^{n} \text{area}(s_\ell) = 3\,G, \text{ and thus } G \geq A/3. \quad \blacktriangleleft$$

7 Conclusion

We conclude with a few open problems:

1. Is the problem of computing the maximum-area anchored rectangle (respectively, square) packing NP-hard?
2. Is there a polynomial-time approximation scheme for the problem of computing an anchored rectangle packing of maximum area?
3. What lower bound on $A(n)$ can be obtained by extending Lemma 4 concerning rectangles from 2 to 3 points? Is there a short proof of Lemma 4?
4. Does Algorithm 9 for computing an anchored square packing of maximum area achieve a ratio of $1/4$? By Theorem 17 and the construction in Fig. 4, the approximation ratio is between $9/47 = 1/5.22\ldots$ and $1/4$. Improvements beyond the $1/5$ ratio are particularly exciting.
5. Is $A(n) = \frac{2}{3}$? Is $A_{\text{sq}}(n) = \frac{7}{27}$?
6. What upper and lower bounds on $A(n)$ and $A_{\text{sq}}(n)$ can be established in higher dimensions?
7. A natural variant of anchored squares is one where the anchors must be the centers of the squares. What approximation can be obtained in this case?

Acknowledgment. The authors are thankful to René Sitters for constructive feedback on the problems discussed in this paper. In particular, the preliminary approximation ratio $1/6$ in Theorem 12 incorporates one of his ideas.

References

1 A. K. Abu-Affash, P. Carmi, M. J. Katz, and G. Morgenstern. Multi cover of a polygon minimizing the sum of areas. *Int. J. Comput. Geom. & Appl.*, 21(6):685–698, 2011.

2 A. Adamaszek and A. Wiese. Approximation schemes for maximum weight independent set of rectangles. In *Proc. 54th Sympos. Found. of Comp. Sci.*, pp. 400-409, IEEE, 2013.

3 A. Adamaszek and A. Wiese. A quasi-PTAS for the two-dimensional geometric knapsack problem. In *Proc. 26th ACM-SIAM Sympos. Discrete Algor.*, pp. 1491–1505, SIAM, 2015.

4 A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Sympos. Comput. Geom.*, pp. 278–290, ACM, 1987.

5 M. Ajtai. The solution of a problem of T. Rado. *Bulletin de l'Académie Polonaise des Sciences, Série des Sci. Mathématiques, Astronomiques et Physiques*, 21:61–63, 1973.

6 S. Aluru. Quadtrees and octrees. In *Handbook of Data Structures and Applications (D. P. Mehta and S. Sahni, editors)*, ch. 19, Chapman & Hall/CRC, 2005.

7 K. Balas, A. Dumitrescu, and Cs. D. Tóth. Anchored rectangle and square packings. Preprint, `arXiv.org/abs/1603.00060.v1`, 2016.

8 K. Balas and Cs. D. Tóth. On the number of anchored rectangle packings for a planar point set. *Theor. Comput. Sci.*, in press, 2016.

9 N. Bansal and A. Khan. Improved approximation algorithm for two-dimensional bin packing. In *Proc. 25th ACM-SIAM Sympos. Discrete Algor.*, pp. 13–25, SIAM, 2014.

10 S. Bereg, A. Dumitrescu, and M. Jiang. Maximum area independent set in disk intersection graphs. *Internat. J. Comput. Geom. Appl.*, 20:105–118, 2010.

11 S. Bereg, A. Dumitrescu, and M. Jiang. On covering problems of Rado. *Algorithmica*, 57:538–561, 2010.

12 S. Bhowmick, K. Varadarajan, and S.-K. Xue. A constant-factor approximation for multi-covering with disks. In *Proc. 29th Sympos. Comput. Geom.*, pp. 243–248, ACM, 2013.

13 S. Bhowmick, K. Varadarajan, and S.-K. Xue. Addendum to "A constant-factor approximation for multi-covering with disks." manuscript, 2014, available at `http://homepage.cs.uiowa.edu/~kvaradar/papers.html`.

14 V. Bílo, I. Caragiannis, C. Kaklamanis, and P. Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *Proc. ESA*, LNCS 3669, pp. 460–471, Springer, 2005.

15 P. Chalermsook and J. Chuzhoy. Maximum independent set of rectangles. In *Proc. 20th ACM-SIAM Sympos. Discrete Algor.*, pp. 892–901, SIAM, 2009.

16 T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46:178–189, 2003.

17 B. Chazelle, R.L. Drysdale, and D.T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15(1):300–315, 1986.

18 K. L. Clarkson. Fast algorithms for the all nearest neighbors problem. In *Proc. 24th IEEE Sympos. Found. Comput. Sci.*, pp. 226–232, IEEE, 1983.

19 A. Dumitrescu and M. Jiang. On the largest empty axis-parallel box amidst n points. *Algorithmica*, 66(2):225–248, 2013.

20 A. Dumitrescu and Cs. D. Tóth. Packing anchored rectangles. *Combinatorica*, 35(1):39–61, 2015.

21 M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Sympos. Comput. Geom.*, pp. 281–288, ACM, 1991.

22 C. Iturriaga and A. Lubiw. Elastic labels around the perimeter of a map. *J. Algorithms*, 47(1):14–39, 2003.

23 J.-W. Jung and K.-Y. Chwa. Labeling points with given rectangles. *Inf. Process. Lett.*, 89(3):115–121, 2004.

24 K. G. Kakoulis and I. G. Tollis. Labeling algorithms. In *Handbook of Graph Drawing and Visualization (R. Tamassia, ed.)*, ch. 28, CRC Press, 2013.

25 D. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Disc. Math.*, 5:36–47, 1992.

26 A. Koike, S.-I. Nakano, T. Nishizeki, T. Tokuyama, and S. Watanabe. Labeling points with rectangles of various shapes. *Int. J. Comput. Geometry Appl.*, 12(6):511–528, 2002.

27 M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels, *Comput. Geom.*, 13:21–47, 1999.

28 R. Rado. Some covering theorems (I). *Proc. Lond. Math. Soc.*, 51:232–264, 1949.

29 R. Rado. Some covering theorems (II). *Proc. Lond. Math. Soc.*, 53:243–267, 1951.

30 R. Rado. Some covering theorems (III). *Proc. Lond. Math. Soc.*, 42:127–130, 1968.

31 T. Rado. Sur un problème relatif à un théorème de Vitali. *Fund. Math.*, 11:228–229, 1928.

32 W. Tutte. *Recent Progress in Combinatorics: Proc. 3rd Waterloo Conf. Combin.* Academic Press, New York, 1969.

33 P. Winkler. Packing rectangles. In *Mathematical Mind-Benders*, pp. 133–134, A.K. Peters Ltd., Wellesley, MA, 2007.

34 B. Zhu and M. Jiang. A combinatorial theorem on labeling squares with points and its application. *J. Comb. Optim.*, 11(4):411–420, 2006.

On Variants of k-means Clustering*

Sayan Bandyapadhya[1] and Kasturi Varadarajan[2]

1 **Department of Computer Science, University of Iowa, Iowa City, USA**
sayan-bandyapadhyay@uiowa.edu
2 **Department of Computer Science, University of Iowa, Iowa City, USA**
kasturi-varadarajan@uiowa.edu

Abstract

Clustering problems often arise in fields like data mining and machine learning. Clustering usually refers to the task of partitioning a collection of objects into groups with similar elements, with respect to a similarity (or dissimilarity) measure. Among the clustering problems, *k-means* clustering in particular has received much attention from researchers. Despite the fact that k-means is a well studied problem, its status in the plane is still open. In particular, it is unknown whether it admits a PTAS in the plane. The best known approximation bound achievable in polynomial time is $9 + \varepsilon$.

In this paper, we consider the following variant of k-means. Given a set C of points in $\mathcal{R}^d$ and a real $f > 0$, find a finite set F of points in $\mathcal{R}^d$ that minimizes the quantity $f * |F| + \sum_{p \in C} \min_{q \in F} ||p - q||^2$. For any fixed dimension d, we design a PTAS for this problem that is based on local search. We also give a "bi-criterion" local search algorithm for k-means which uses $(1 + \varepsilon)k$ centers and yields a solution whose cost is at most $(1 + \varepsilon)$ times the cost of an optimal k-means solution. The algorithm runs in polynomial time for any fixed dimension.

The contribution of this paper is two-fold. On the one hand, we are able to handle the square of distances in an elegant manner, obtaining a near-optimal approximation bound. This leads us towards a better understanding of the k-means problem. On the other hand, our analysis of local search might also be useful for other geometric problems. This is important considering that little is known about the local search method for geometric approximation.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases k-means, Facility location, Local search, Geometric approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.14

1 Introduction

Clustering is the task of partitioning a set of items (or objects) into groups of similar items with respect to a similarity (or dissimilarity) measure. Due to its fundamental nature clustering has several applications in fields like data mining, machine learning, pattern recognition, and image processing [8, 11, 12, 13, 14, 19, 20, 22, 31]. Often the objects to cluster are mapped to a high-dimensional metric space and the distance between two objects represents the similarity (or dissimilarity) between the objects. Then the goal is to minimize (or maximize) a certain objective function that depends on the distances between the objects. Among the different variants of clustering, the *k-means* problem in particular has received much attention. In k-means clustering, given a set P of n points in $\mathcal{R}^d$ and an integer $k > 0$,

* This material is based upon work supported by the National Science Foundation under Grant CCF-1318996.


© Sayan Bandyapadhyay and Kasturi Varadarajan;
licensed under Creative Commons License CC-BY

the goal is to find a set K of k centers in $\mathcal{R}^d$, such that the quantity

$$cost(K) = \sum_{p \in P} \min_{q \in K} ||p - q||^2$$

is minimized. k-means is known to be $\mathcal{NP}$-hard even in the plane [1, 26]. The most common heuristic for k-means is an algorithm due to Lloyd [25] which is based on an iterative refinement technique. For historical reasons this algorithm is referred to as the k-means algorithm. Though this algorithm converges to a local optimum, the quality of the solution is not guaranteed to be "good" compared to the global optimum. Thus researchers have focused on designing polynomial time approximation algorithms that give some theoretical guarantee for all inputs. In fact, there are many $(1 + \varepsilon)$-factor approximation algorithms whose time complexity depend linearly on n [16, 17, 24, 28]. Unfortunately, the time complexity of these algorithms depends exponentially on k and hence they are not suitable in practice when k is large. For arbitrary k and d, the best known approximation factor is $9 + \varepsilon$ based on a local search technique [21]. On the other hand, Makarychev *et. al* [27] have designed three "bi-criteria" approximation algorithms for k-means that use at most βk $(> k)$ centers and achieve an approximation factor $\alpha(\beta)$ $(< 9 + \varepsilon)$ that depends on β. Moreover, $\alpha(\beta)$ decreases rapidly with β (for instance $\alpha(2) < 2.59, \alpha(3) < 1.4$). These algorithms have only polynomial dependence on the dimension of input points. Recently, Awasthi *et. al* [6] have studied the inapproximabilty of k-means. They have shown that k-means is APX-hard in sufficiently high $(\Omega(\log n))$ dimensions. However, as they have pointed out in their paper, the status of this problem in constant dimensions is still not resolved. See also [5, 30] for some related work.

An insight about the difficulty of k-means can be found by comparing it to *k-median* clustering. k-median is similar to k-means except the goal is to minimize the sum of distances, instead of the sum of squares of distances. Arora *et. al* [3] presented a PTAS for k-median in the plane based on a novel technique due to Arora [2]. Kolliopoulos and Rao [23] improved the time complexity significantly to $O(\rho n \log n \log k)$, where $\rho = exp[O((1 + \log 1/\varepsilon)/\varepsilon)^{d-1}]$ and ε is the constant of the PTAS. From the results on k-median one might conclude that a reason behind the difficulty of k-means is its objective function. One reason that squares of distances are harder to handle compared to Euclidean distances is they do not satisfy the triangle inequality in general. However, the important question in this context is, "Is the objective function of k-means by itself sufficient to make this problem harder?". We are interested in addressing this question in this paper. To set up the stage we consider another famous problem, which is called the facility location problem.

Facility location is similar to the k-median problem, where we are given a set of points (clients) in $\mathcal{R}^d$. The goal is to choose another set of points (facilities) in $\mathcal{R}^d$ which "serve" the clients. Though there is no global constraint on the number of facilities, for each facility, we need to pay a fixed cost. Here the objective function to minimize is the facility costs plus the sum of the distances from each of the clients to its nearest facility. Facility location has got much attention in operations research, approximation algorithms, etc. Arora *et. al* [3] give a PTAS for facility location in the plane using the same technique that they use to solve k-median. Actually, k-median has always been considered harder compare to facility location due to the global constraint on the number of centers as mentioned in [3]. Now going back to our original question for k-means one can infer that the global constraint in k-means might also play a crucial role. Motivated by this, we define the following variant of facility location.

Sum of Squares Facility Location Problem (SOS-FL). Given a set C of points (clients) in $\mathcal{R}^d$ and a real $f > 0$, find a finite set F of points (facilities) in $\mathcal{R}^d$, such that the quantity

$$cost(F) = f * |F| + \sum_{p \in C} \min_{q \in F} ||p - q||^2$$

is minimized. Note that SOS-FL is similar to k-means except the global constraint on the number of facilities (or centers) is absent here. In this paper we study the following interesting question.

Is it possible to get a PTAS for SOS-FL in $\mathcal{R}^d$ for fixed d?

We answer this question in the affirmative. In particular for any $\varepsilon > 0$, we give a $(1+\varepsilon)$-factor approximation for SOS-FL based on a *local search method*. The significance of this result is that it addresses our earlier question regarding k-means. To be precise it lets us infer that it is the joint effect of the global constraint and the objective function that makes k-means complicated to deal with.

Local Search. Local search is a popular technique in practice, and the study of when it yields approximation guarantees has a long history in combinatorial optimization. For geometric problems, results on approximation guarantees for local search have been relatively rare until recently. Thus we know very little about local search for geometric approximation. Arya *et. al* [4] gave a $3 + \frac{2}{p}$ factor approximation for metric k-median based on a local search that swaps p facilities; a different and arguably simplified analysis was given by Gupta and Tangwongsan [15]. The $9+\varepsilon$ factor approximation for k-means [21], mentioned above, builds on the analysis by Arya *et. al* [4]. Mustafa and Ray [29] gave a local search PTAS for the discrete hitting set problem over pseudodisks and r-admissible regions in the plane. Chan and Har-Peled [9] designed a local search method for the independent set problem over fat objects, and for pseudodisks in the plane, which yields a PTAS. Recently, Cohen-Addad and Mathieu [10] showed the effectiveness of local search technique for geometric optimization by designing local search algorithms for many geometric problems including k-median and facility location. For facility location they achieved a PTAS. For k-median their approach yields a $1+\varepsilon$ factor approximation using at most $(1+\varepsilon)k$ centers. Very recently, Bhattiprolu and Har-Peled [7] designed a local search PTAS for a geometric hitting set problem over balls in $\mathcal{R}^d$. Their PTAS also works for an infinite set of balls which can be represented implicitly in a specific way mentioned in their paper. One of the bicriteria algorithms for k-means in [27], mentioned above, is based on a local search method.

1.1 Our Results and Techniques

In this work we consider both SOS-FL and k-means. The main contribution of this work is that we are able to handle the square of distances in an elegant way, which yields near optimal approximation bounds. In particular, it gives a better understanding of the classical k-means problem, whose status in the plane has remained open for a long time. We design polynomial time approximation algorithms based on the local search technique for both of these problems. For any $\varepsilon > 0$, the algorithm for SOS-FL yields a $(1+\varepsilon)$-factor approximation. For k-means, the algorithm uses at most $(1+\varepsilon)k$ centers and yields a solution whose cost is at most $(1+\varepsilon)$ times the cost of an optimal k-means solution.

The algorithm and the analysis for both of the problems are similar. However, in case of k-means there are more subtleties, which arise due to the limitation on the number of

Algorithm 1 Local Search

Require: A set of clients $C \subset \mathcal{R}^d$, an $f > 0$, and a parameter $\varepsilon > 0$.
Ensure: A set of facilities F.
1: $F \leftarrow$ the set of facilities with one facility at each client
2: **while** $\exists$ a set F_1 s.t. $cost(F_1) < (1 - \frac{1}{n})cost(F)$ and $|F_1 \setminus F| + |F \setminus F_1| \leq \frac{c}{\varepsilon^d}$ **do**
3: $\quad F \leftarrow F_1$
4: **return** F

centers. In general, both of the algorithms are based on a local search method that allows swapping in and out of constant number of facilities or centers. Like the approaches in [7, 9, 10, 29] we also use separators to prove the quality of the approximation. To be precise we use the separator from [7], which turns out to be quite suitable for the purpose of handling the square of distances. The separator is used repeatedly to partition the local and global optimal facilities simultaneously into constant sized "parts", like the analysis of the hitting set algorithm in [7]. The rest of the analysis involves assignment of clients corresponding to the local facilities of each "part" to the global facilities corresponding to that "part" only or to some "auxilliary" points. Here one should be careful that a client should not be assigned to a point "far" away from it compared to its nearest local and global facility. The choice of the separator plays a crucial role to give a bound on this cost. From a high level our approach is similar to that of the work of Cohen-Addad and Mathieu [10], which is one source of inspiration for our work. But the details of the analysis are significantly different in places. For example, they use the dissection technique from [23] as their separator and thus the assignment in their case is completely different and more complicated than ours. In this regard we would like to mention, that the dissection technique from [23] or the quadtree based approach of Arora [2] are not flexible enough to handle the square of distances. The local search algorithms for SOS-FL and k-means are described in Section 2 and Section 3, respectively.

2 PTAS for Sum of Squares Facility Location

In this section we describe a simple local search algorithm for SOS-FL. We show that the solution returned by this algorithm is within $(1 + O(\varepsilon))$-factor of the optimal solution for any $\varepsilon > 0$. Recall that in SOS-FL we are given a set C of points in $\mathcal{R}^d$ and a real $f > 0$. Let $|C| = n$. For a point p and a set R of points, let $d(p, R) = \min_{q \in R} ||p - q||$.

2.1 The Local Search Algorithm

The local search algorithm starts with the solution where one facility is placed at each client (see Algorithm 1). Note that the cost of this solution is nf. Denote by OPT the cost of any optimal solution. As $OPT \geq f$, the initial solution has cost at most $n \cdot OPT$. In each iteration the algorithm looks for local improvement. The algorithm returns the current set of facilities if there is no such improvement. Notice that in line 2, we consider swaps with at most $\frac{c}{\varepsilon^d}$ facilities, where c is a constant. Next we show that this algorithm runs in polynomial time.

We note that in each iteration of the while loop the cost of F gets dropped by a factor of at least $(1 - \frac{1}{n})$. Let the number of iterations of the while loop be t. As we start with a

solution of cost at most $n \cdot OPT$, after t iterations we have

$$cost(F) \leq \left(1 - \frac{1}{n}\right)^t \cdot n \cdot OPT.$$

As $cost(F) \geq OPT$, we have

$$OPT \leq \left(1 - \frac{1}{n}\right)^t \cdot n \cdot OPT.$$

Thus the number t of iterations of the while loop is $O(n \log n)$. Now consider a single iteration of the while loop: we need to compute if there exists an F_1 such that $cost(F_1) < (1 - \frac{1}{n}) cost(F)$ and $|F_1 \setminus F| + |F \setminus F_1| \leq \frac{c}{\varepsilon^d}$. Assuming such an F_1 exists, fix one such F_1. Let $A = F \setminus F_1$. Since $|A| \leq \frac{c}{\varepsilon^d}$, we can enumerate over the possibilities for A. Let us therefore assume we have A. We would like to find a set $A' \subset \mathcal{R}^d$ with $\lambda := |A'| \leq \frac{c}{\varepsilon^d} - |A|$ that minimizes $cost((F \setminus A) \cup A')$. Since each point in A' has d coordinates, this is an optimization problem with $\lambda \cdot d$ variables once we fix λ.

This optimization problem can be solved in polynomial time using techniques similar to the ones used by [18]. To explain this further, let the optimal A' be $\{a'_1, a'_2, \ldots, a'_\lambda\}$, let χ be the assignment that assigns each client in C to the nearest facility in $(F \setminus A) \cup A'$. For solving our optimization problem, it suffices to know χ, even though A' is unknown: we can set a'_i to be the centroid of the points $\chi^{-1}(a'_i)$. For $c \in C$, there are $\lambda + 1$ possibilities for $\chi(c)$: the nearest facility in $F \setminus A$, or one of the λ unknown facilities in A'. Naively, this suggests $(\lambda + 1)^n$ possibilites for χ. However, if we view the problem as an optimization problem in $\lambda \cdot d$ dimensions, and use standard ideas about arrangements, we can see that we only need to consider $n^{O(\lambda \cdot d)}$ possibilities for χ, and these can be enumerated in $n^{O(\lambda \cdot d)}$ time. We conclude that Algorithm 1 can be implemented in polynomial time.

▶ Remark. An alternative approach to solving the problem of finding an A' of size λ that minimizes $cost((F \setminus A) \cup A')$ is as follows. Our analysis for local search works even if we solve this problem to within an approximation of $(1 + \frac{1}{2n})$. For this purpose, we can compute a $\frac{1}{2n}$-approximate centroid set $\mathcal{D} \subset \mathcal{R}^d$ of size $n^{O(d)}$ in time $n^{O(d)}$, by adapting an algorithm of Matousek [28]. We can think of $\mathcal{D}$ as a suitable discretization of $\mathcal{R}^d$. With $\mathcal{D}$ in hand, we simply minimize $cost((F \setminus A) \cup A')$ over all subsets $A' \subset \mathcal{D}$ of size λ, in time $n^{O(\lambda \cdot d)}$.

2.2 Analysis of the Local Search Algorithm

We analyze the local search algorithm using a partitioning scheme based on the separator theorem from [7]. The idea is to partition the set of facilities in the local search solution and an optimal solution into parts of size $O(\frac{1}{\varepsilon^d})$. Now for each such small part, we assign the clients corresponding to the local facilities of that part either to the optimal facilities in that part or to the points belonging to a special set. This yields a new solution, whose symmetric difference with the local search solution contains $O(\frac{1}{\varepsilon^d})$ facilities. Thus, using the local optimality criteria, the cost of this new solution is not "small" compared to the local search solution. This gives us one inequality for each part. Then we combine the inequalities corresponding to all the parts to give a bound on the cost of the local search solution. To start with we describe the separator theorem.

2.2.1 Separator Theorem

A ball B is said to be stabbed by a point p if $p \in B$. The following theorem is due to Bhattiprolu and Har-Peled [7] which shows the existence of a "small" point set (separator), that divides a given set of points into two in a "balanced" manner.

Algorithm 2 PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$

$\mu = \frac{\gamma}{\varepsilon^d}$, $i = 1$, $L_1 = \mathcal{L}$, $O_1 = \mathcal{O}$, $\mathcal{Z}_1 = \emptyset$
while $|L_i \cup O_i \cup \mathcal{Z}_i| > \alpha\mu$ **do**
  Let B_i, T_i be the ball and the point set computed by applying the Separator algorithm on the set $L_i \cup O_i \cup \mathcal{Z}_i$ with parameter μ
  $\mathcal{L}_i = L_i \cap B_i, \mathcal{O}_i = O_i \cap B_i$
  $L_{i+1} = L_i \setminus \mathcal{L}_i, O_{i+1} = O_i \setminus \mathcal{O}_i$
  $\mathcal{Z}_{i+1} = (\mathcal{Z}_i \setminus B_i) \cup T_i$
  $i = i + 1$
$I = i$
Let B_I be any ball that contains all the points in $L_I \cup O_I \cup \mathcal{Z}_I$
$T_I = \emptyset$

▶ **Theorem 1** ([7] Separator Theorem). *Let X be a set of points in $\mathcal{R}^d$, and $\mu > 0$ be an integer such that $|X| > \alpha\mu$, where α is a constant. There is an algorithm which can compute, in $O(|X|)$ expected time, a set $\mathcal{T}$ of $O(\mu^{1-\frac{1}{d}})$ points and a sphere $\mathcal{S}$ such that (a) the number of points of X inside $\mathcal{S}$ is $\Theta(\mu)$, and (b) any ball that intersects $\mathcal{S}$ and is stabbed by a point of X is also stabbed by a point of $\mathcal{T}$.*

The next corollary follows from Theorem 1, and will be useful for the analysis of our local search algorithm.

▶ **Corollary 2.** *Let X be a set of points in $\mathcal{R}^d$,and $\mu > 0$ be an integer such that $|X| > \alpha\mu$, where α is a constant. There is an algorithm which can compute, in $O(|X|)$ expected time, a set $\mathcal{T}$ of $O(\mu^{1-\frac{1}{d}})$ points and a ball B such that (a) $|B \cap X| = \Theta(\mu)$, and (b) for any point $p \in \mathcal{R}^d$, $d(p, \mathcal{T}) \leq \max\{d(p, X \setminus B), d(p, B \cap X)\}$.*

Proof. We use the same Algorithm in Theorem 1 to compute the sphere $\mathcal{S}$ and the set $\mathcal{T}$. Let B be the ball that has $\mathcal{S}$ as its boundary. Now consider any point $p \in \mathcal{R}^d$. Let p_1 (resp. p_2) be a point in $B \cap X$ (resp. $X \setminus B$) nearest to p. If $p \in B$, the ball B_1 centered at p and having radius $||p - p_2||$ must intersect the sphere $\mathcal{S}$, as $p_2 \notin B$. As p_2 stabs B_1, by Theorem 1 there is a point in $\mathcal{T}$ that also stabs B_1. Hence $d(p, \mathcal{T}) \leq ||p - p_2|| = d(p, X \setminus B)$. Similarly, if $p \notin B$, the ball B_2 centered at p and having radius $||p - p_1||$ intersects the sphere $\mathcal{S}$, as $p_1 \in B$. As B_2 is stabbed by p_1, by Theorem 1 there is a point in $\mathcal{T}$ that also stabs B_2. Hence $d(p, \mathcal{T}) \leq ||p - p_1|| = d(p, B \cap X)$ and the corollary follows. ◀

The algorithm in Corollary 2 will be referred to as the *Separator algorithm.*

2.2.2 The Partitioning Algorithm

For the sake of analysis fix an optimal solution $\mathcal{O}$. Let $\mathcal{L}$ be the solution computed by the local search algorithm. We design a procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$ (see Algorithm 2) which divides the set $\mathcal{L} \cup \mathcal{O}$ into disjoint subsets of small size using the Separator algorithm. The procedure iteratively removes points from the set until the size of the set becomes less than or equal to $\alpha\mu$, where α is the constant in Corollary 2, and $\mu = \frac{\gamma}{\varepsilon^d}$ for some constant γ. This procedure is similar to the ApproximateSeparator algorithm in [7]. Next, we describe some important properties of this procedure that we will need to bound the cost of the local search solution. But before proceeding further we define some notation.

Let $T = \cup_{i=1}^{I} T_i$ be the union of the point sets computed by the Separator algorithm in PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. Also let $C_l = \{p \in C | d(p, \mathcal{L}) \leq d(p, \mathcal{O})\}$ and $C_o = C \setminus C_l$. Consider

a point set $R \subset \mathcal{R}^d$. We denote the nearest neighbor voronoi diagram of R by $\mathcal{V}_R$. For $p \in R$, let $\mathcal{V}_R(p)$ be the voronoi cell of p in $\mathcal{V}_R$. Also let $C_R(p) = \mathcal{V}_R(p) \cap C$, that is $C_R(p)$ is the set of clients that are contained in the voronoi cell of p in $\mathcal{V}_R$. For $Q \subseteq R$, define $C_R(Q) = \cup_{q \in Q} C_R(q)$.

Now consider a client c. Denote its nearest neighbor in $\mathcal{O}$ (resp. $\mathcal{L}$) by $c(\mathcal{O})$ (resp. $c(\mathcal{L})$). Also let $c_O = ||c - c(\mathcal{O})||^2$ and $c_L = ||c - c(\mathcal{L})||^2$.

▶ **Definition 3.** An *assignment* is a function that maps a set of clients to the set $\mathcal{L} \cup \mathcal{O} \cup T$.

Now with all these definitions we move on towards the analysis. We begin with the following observation.

▶ **Observation 4.** Consider the procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. The following assertions hold.

1. $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$ for a constant β and each $1 \leq i \leq I$
2. $I \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$
3. $|T| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$
4. $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5$

Proof. 1. Note that $|T_i| = O(\mu^{1-\frac{1}{d}}) = O(\frac{1}{\varepsilon^{d-1}})$ and $|\mathcal{L}_i \cup \mathcal{O}_i \cup \mathcal{Z}_i| = O(\mu) = O(\frac{1}{\varepsilon^d})$ for $1 \leq i \leq I$. Thus there is a constant β such that $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$.
2. As in each iteration we add $O(\mu^{1-\frac{1}{d}})$ points and remove $\theta(\mu)$ points, the number of iterations $I = O(\frac{|\mathcal{L}|+|\mathcal{O}|}{\mu}) = O(\varepsilon^d(\frac{|\mathcal{L}|+|\mathcal{O}|}{\gamma}))$. By choosing the constant γ sufficiently large one can ensure that $I \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$.
3. $|T| = \sum_{i=1}^{I} |T_i| = O(\frac{|\mathcal{L}|+|\mathcal{O}|}{\mu} \cdot \mu^{1-\frac{1}{d}}) = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)/\gamma^{\frac{1}{d}}) \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$, by choosing the value of γ sufficiently large.
4. Consider any point $p \in T_i$ for some $1 \leq i \leq I$. If $p \in \mathcal{Z}_j \cap B_j$ for some $j > i$, then p is removed in iteration j and hence cannot appear in any other $\mathcal{Z}_t \cap B_t$ for $j+1 \leq t \leq I$. Thus p can appear in at most two sets of the collection $\{T_1 \cup (\mathcal{Z}_1 \cap B_1), \ldots, T_I \cup (\mathcal{Z}_I \cap B_I)\}$. It follows that $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| \leq 2 \sum_{i=1}^{I} |T_i| = 2|T| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5$. ◀

The next lemma states the existence of a "cheap" assignment for any client c such that its nearest neighbor $c(\mathcal{O})$ in $\mathcal{O}$ is in $\mathcal{O}_i$ and its nearest neighbor $c(\mathcal{L})$ in $\mathcal{L}$ is in $\mathcal{L}_j$ for $i < j$.

▶ **Lemma 5.** *Consider any client c, such that $c(\mathcal{O}) \in \mathcal{O}_i$, $c(\mathcal{L}) \in \mathcal{L}_j$ with $1 \leq i < j \leq I$. Also consider the sets T_j, $\mathcal{Z}_j$, and the ball B_j computed by PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. There exists a point $p \in (\mathcal{Z}_j \cap B_j) \cup T_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$.*

Proof. To prove this lemma, we first establish the following claim.

▶ **Claim 6.** *For any $i + 1 \leq t \leq j$, there exists a point $p \in \mathcal{Z}_t$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$.*

Proof. We prove this claim using induction on the iteration number. In base case consider the iteration i. Let $X = L_i \cup O_i \cup \mathcal{Z}_i$. As $c(\mathcal{O}) \in B_i$ and $c(\mathcal{L}) \in B_j$, $c(\mathcal{O}), c(\mathcal{L}) \in X$. Now by Corollary 2, $d(c, T_i) \leq \max\{d(c, X \setminus B_i), d(c, B_i \cap X)\}$. As $c(\mathcal{O})$ is the nearest neighbor of c in $\mathcal{O}$ and $c(\mathcal{O}) \in B_i$, $d(c, B_i \cap X) \leq ||c - c(\mathcal{O})||$. Also $c(\mathcal{L})$ is the nearest neighbor of c in $\mathcal{L}$ and $c(\mathcal{L}) \notin B_i$. Thus $d(c, X \setminus B_i) \leq ||c - c(\mathcal{L})||$. Hence $d(c, T_i) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let p be the point in T_i nearest to c. As $T_i \subseteq \mathcal{Z}_{i+1}$, $p \in \mathcal{Z}_{i+1}$ and the base case holds.

Now suppose the claim is true for any iteration $t < j - 1 \leq I - 1$. We show that the claim is also true for iteration $t + 1$. By induction, there is a point $p \in \mathcal{Z}_{t+1}$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Now there can be two cases: (i) $p \notin B_{t+1}$, and

(ii) $p \in B_{t+1}$. Consider the first case. In this case, by definition of $\mathcal{Z}_{t+2}$, $p \in \mathcal{Z}_{t+2}$ and the claim holds. Thus consider the second case. Let $X = L_{t+1} \cup O_{t+1} \cup \mathcal{Z}_{t+1}$. By Corollary 2, $d(c, T_{t+1}) \leq \max\{d(c, X \setminus B_{t+1}), d(c, B_{t+1} \cap X)\}$. As $p \in B_{t+1} \cap X$, $d(c, B_{t+1} \cap X) \leq ||c-p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Now $c(\mathcal{L}) \notin B_{t+1}$, as $t < j - 1$. Also $c(\mathcal{L}) \in L_{t+1} \subseteq X$. Thus $d(c, X \setminus B_{t+1}) \leq ||c - c(\mathcal{L})||$. It follows that $d(c, T_{t+1}) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let q be the point in T_{t+1} nearest to c. As $T_{t+1} \subseteq \mathcal{Z}_{t+2}$, $q \in \mathcal{Z}_{t+2}$ and the claim holds also for this case. ◀

Consider the iteration j. From Claim 6 it follows that there exists a point $p \in \mathcal{Z}_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Thus if $p \in B_j$, then $p \in \mathcal{Z}_j \cap B_j$, and we are done. Note that the way B_I is chosen, $\mathcal{Z}_I \subseteq B_I$. Thus in case $j = I$, $p \in \mathcal{Z}_j \cap B_j$. Hence consider the case when $j \neq I$ and $p \notin B_j$. Let $X = L_j \cup O_j \cup \mathcal{Z}_j$. As $p \in \mathcal{Z}_j$, $p \in X$. Also by Corollary 2, $d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}$. As $c(\mathcal{L}) \in B_j \cap X$, $d(c, B_j \cap X) \leq ||c - c(\mathcal{L})||$. Now $p \in X \setminus B_j$. Thus $d(c, X \setminus B_j) \leq ||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Hence $d(c, T_j) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$ and the lemma follows. ◀

Now we extend Lemma 5 for any client whose nearest neighbor in $\mathcal{L}$ is in $\mathcal{L}_j$, but the nearest neighbor in $\mathcal{O}$ is not in $\mathcal{O}_j$, where $1 \leq j \leq I$.

▶ **Lemma 7.** *Consider the sets T_j, $\mathcal{Z}_j$ and the ball B_j computed by PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$, where $1 \leq j \leq I$. There is an assignment g of the clients in $C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$ to $T_j \cup (\mathcal{Z}_j \cap B_j)$ with the following properties:*

1. *for $c \in (C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)) \cap C_l$, $||c - g(c)||^2 \leq c_O$;*
2. *for $c \in (C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)) \cap C_o$, $||c - g(c)||^2 \leq c_L$.*

Proof. We show how to construct the assignment g for each client in $C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$. Consider any client $c \in C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$. Let $\mathcal{O}_i$ be the subset of $\mathcal{O}$ that contains $c(\mathcal{O})$. If j is equal to I, then $i < j$. Otherwise, there could be two cases: (i) $i < j$, and (ii) $i > j$. Consider the case when $i < j$ for $1 \leq j \leq I$. By Lemma 5, there is a point $p \in (\mathcal{Z}_j \cap B_j) \cup T_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let $g(c)$ be p in this case. Now consider the case when $i > j$ such that $j < I$. Let $X = L_j \cup O_j \cup \mathcal{Z}_j$. By Corollary 2, $d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}$. As $c(\mathcal{L}) \in B_j \cap X$, $d(c, B_j \cap X) \leq ||c - c(\mathcal{L})||$. Now note that $c(\mathcal{O}) \in X$. Also $c(\mathcal{O}) \notin B_j$, as $c(\mathcal{O}) \in \mathcal{O}_i$ and $i > j$. Thus $c(\mathcal{O}) \in X \setminus B_j$ and $d(c, X \setminus B_j) \leq ||c - c(\mathcal{O})||$. Hence $d(c, T_j) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let $g(c)$ be the point in T_j nearest to c in this case.

In both cases $||c - g(c)|| \leq \max\{||c - c(\mathcal{L})||, ||c - c(\mathcal{O})||\}$. If $c \in C_l$, $||c - g(c)||^2 \leq ||c - c(\mathcal{O})||^2 = c_O$. Otherwise, $c \in C_o$, and thus $||c - g(c)||^2 \leq ||c - c(\mathcal{L})||^2 = c_L$. Hence the lemma holds. ◀

2.2.3 Approximation Bound

The next lemma gives an upper bound on the quality of the local search solution.

▶ **Lemma 8.** $cost(\mathcal{L}) \leq (1 + O(\varepsilon))cost(\mathcal{O})$.

Proof. Fix an iteration i, where $1 \leq i \leq I$. Consider the solution $S_i = (\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$. By Observation 4, $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$. Thus $|\mathcal{L} \setminus S_i| + |S_i \setminus \mathcal{L}| \leq \frac{\beta}{\varepsilon^d}$. By choosing the constant c in Algorithm 1 sufficiently large, one can ensure that $\beta \leq c$. Hence due to the local optimality condition in Algorithm 1 it follows that,

$$cost(S_i) \geq (1 - \frac{1}{n})cost(\mathcal{L})\,. \tag{1}$$

To upper bound the cost of S_i we use an assignment of the clients to the facilities in S_i. Consider a client c. There can be three cases: (i) c is nearer to a facility of $\mathcal{O}_i$ than the facilities in $(\mathcal{O} \setminus \mathcal{O}_i) \cup (\mathcal{L} \setminus \mathcal{L}_i)$, that is, $c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$, (ii) c is not in $C_{\mathcal{O}}(\mathcal{O}_i)$ and c is nearer to a facility of $\mathcal{L}_i$ than the facilities in $\mathcal{L} \setminus \mathcal{L}_i$, that is, $c \in C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$, (iii) c does not appear in case (i) and (ii), that is, c is not in the union of $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$. Let R_i be the set of the clients that appear in case (iii). Note that a client cannot appear in both cases (i) and (ii), as the sets $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$ are disjoint. Also note, that if $c \in C_{\mathcal{L}}(\mathcal{L}_i)$, c must appear in case (i) or (ii). Thus if c is corresponding to case (iii), its nearest neighbor $c(\mathcal{L})$ in $\mathcal{L}$ must be in $\mathcal{L} \setminus \mathcal{L}_i$.

Now we describe the assignment. Note that we can assign the clients only to the facilities in $S_i = (\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$. For a client of type (i), assign it to a facility in $\mathcal{O}_i$ nearest to it. For a client of type (ii), use the assignment g in Lemma 7 to assign it to a point in $T_i \cup (\mathcal{Z}_i \cap B_i)$. For a client of type (iii), assign it to a facility in $\mathcal{L} \setminus \mathcal{L}_i$ nearest to it. Thus by Inequality 1,

$$|(\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} c_O + \sum_{c \in C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)} ||c - g(c)||^2 + \sum_{c \in R_i} c_L \geq (1 - \frac{1}{n})(|\mathcal{L}|f + \sum_{c \in C} c_L)\,. \quad (2)$$

By Lemma 7, for a client c in $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_o$, $||c - g(c)||^2$ is at most c_L; for a client c in $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l$, $||c - g(c)||^2$ is at most c_O. It follows that,

$$|\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} (c_O - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_o} (c_L - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f \quad (3)$$

$$\Rightarrow |\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} (c_O - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f \quad (4)$$

$$\Rightarrow |\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f\,.$$

The last inequality follows by noting that the union of $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l$ is equal to the set $C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)$. Summing over all i we get,

$$\sum_{i=1}^{I} |\mathcal{O}_i|f + \sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{i=1}^{I} \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)} (c_O - c_L) \geq -\frac{I}{n} cost(\mathcal{L}) + \sum_{i=1}^{I} |\mathcal{L}_i|f\,. \quad (5)$$

By Observation 4, $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|))$. Hence we get,

$$|\mathcal{O}|f + O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|))f + \sum_{c \in C} (c_O - c_L) \geq -\frac{I}{n} cost(\mathcal{L}) + |\mathcal{L}|f\,.$$

Since $I = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)) = O(\varepsilon n)$ (Observation 4), we obtain

$$cost(\mathcal{L}) \leq (1 + O(\varepsilon))cost(\mathcal{O})\,.$$ ◀

Algorithm 3 Local Search

Require: A set of points $P \subset \mathcal{R}^d$, an integer k, a constant $\varepsilon > 0$.
Ensure: A set of centers.
1: $K \leftarrow$ the solution returned by the algorithm in [21]
2: Add arbitrary centers to K to ensure $|K| = (1 + 5\varepsilon)k$
3: **while** $\exists$ a set K_1 s.t. $|K_1| \leq (1+5\varepsilon)k$, $cost(K_1) < (1-\frac{1}{n})cost(K)$ and $|K_1 \setminus K| + |K \setminus K_1| \leq \frac{c}{\varepsilon^{2d}}$ **do**
4: $K \leftarrow K_1$
5: If needed, add arbitrary centers to K to ensure $|K| = (1 + 5\varepsilon)k$
6: **return** K

As mentioned before, the running time of Algorithm 1 is polynomial for fixed d, and hence we have established the following theorem.

▶ **Theorem 9.** *There is a local search algorithm for SOS-FL which yields a PTAS.*

3 Bi-criteria Approximation scheme for k-means

In this section we describe a local search algorithm for k-means which uses $(1 + O(\varepsilon))k$ centers and yields a solution whose cost is at most $(1 + O(\varepsilon))$ times the cost of an optimal k-means solution. The local search algorithm and its analysis are very similar to the ones for SOS-FL. Recall that in k-means we are given a set P of n points in $\mathcal{R}^d$ and an integer $k > 0$.

3.1 The Local Search Algorithm

Fix an $\varepsilon > 0$. The local search algorithm (see Algorithm 3) starts with the solution computed by the $9 + \varepsilon$ factor approximation algorithm in [21]. Upon termination, the locally optimal solution K has exactly $(1 + 5\varepsilon)k$ centers. Using an argument similar to the one in case of SOS-FL one can show that this algorithm also runs in polynomial time.

3.2 Analysis of the Local Search Algorithm

Let $\mathcal{L}$ be the solution computed by Algorithm 3. For the sake of analysis fix an optimal solution $\mathcal{O}$. We use the procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$ to compute the sets $\mathcal{L}_i$, $\mathcal{O}_i$, $\mathcal{Z}_i \cap B_i$ and T_i for $1 \leq i \leq I$. We use the same $\mu = \frac{\gamma}{\varepsilon^d}$ in this procedure. We note, that Observation 4, Lemma 5, and Lemma 7 hold in this case also, as they directly follow from the PARTITION procedure, which works on any two input sets of points designated by $\mathcal{L}$ and $\mathcal{O}$. Let $R_i = \mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$ for $1 \leq i \leq I$. Note, by Observation 4, that $|R_i| \leq \frac{\beta}{\varepsilon^d}$. Also note that $\mathcal{Z}_i \cap B_i \subseteq T$ for each $1 \leq i \leq I$, where $T = \cup_{i=1}^{I} T_i$. Now we use the following lemma to group the balls returned by PARTITION into groups of "small" size. This lemma is similar to the Balanced Clustering Lemma in [10].

▶ **Lemma 10.** *Consider the collection $R = \{R_1, \ldots, R_I\}$ of sets with $R_j = \mathcal{L}_j \cup \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ and $|R_j| \leq \frac{\beta}{\varepsilon^d}$ for $1 \leq j \leq I$, where β is the constant in Observation 4. There exists a collection $\mathcal{P} = \{P_1, \ldots, P_p\}$, with $P_i \subseteq R$ for $1 \leq i \leq p$, $P_i \cap P_j = \emptyset$ for any $1 \leq i < j \leq p$, and $\cup_{i=1}^{p} P_i = R$, which satisfies the following properties:*
1. $|P_i| \leq \frac{2\beta}{\varepsilon^d}$ *for* $1 \leq i \leq p$;
2. $\sum_{R_j \in P_i} |\mathcal{L} \cap R_j| \geq \sum_{R_j \in P_i} |(\mathcal{O} \cup T) \cap R_j|$ *for* $1 \leq i \leq p$.

We note that $\mathcal{L} \cap R_j = \mathcal{L}_j$ and $(\mathcal{O} \cup T) \cap R_j = \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ for $1 \leq j \leq I$. Before proving the lemma we use it to get an approximation bound on the quality of the local search solution.

▶ **Lemma 11.** *cost*$(\mathcal{L}) \leq (1 + O(\varepsilon))$*cost*$(\mathcal{O})$.

Proof. Consider the collection $\mathcal{P}$ as mentioned in Lemma 10. Also consider any element J of $\mathcal{P}$. Let $\mathcal{L}_J = \cup_{R_i \in J}(\mathcal{L} \cap R_i)$, $\mathcal{O}_J = \cup_{R_i \in J}(\mathcal{O} \cap R_i)$, and $T_J = \cup_{R_i \in J}(T \cap R_i)$. Now consider the solution $S_J = (\mathcal{L} \setminus \mathcal{L}_J) \cup \mathcal{O}_J \cup T_J$. By Lemma 10, and using the fact that $\mathcal{L} \cap R_i$ and $\mathcal{L} \cap R_j$ are disjoint for $i \neq j$, $|\mathcal{L}_J| = \sum_{R_j \in J} |\mathcal{L} \cap R_j| \geq \sum_{R_j \in J} |(\mathcal{O} \cup T) \cap R_j| \geq |\mathcal{O}_J \cup T_J|$. Thus by definition of S_J, $|S_J| \leq |\mathcal{L}| = (1 + 5\varepsilon)k$. Now J contains at most $\frac{2\beta}{\varepsilon^d}$ sets and thus $|\mathcal{L} \setminus S_J| + |S_J \setminus \mathcal{L}| \leq |\mathcal{L}_J \cup \mathcal{O}_J \cup T_J| \leq \frac{2\beta}{\varepsilon^d}\frac{\beta}{\varepsilon^d} \leq \frac{2\beta^2}{\varepsilon^{2d}}$. By choosing the constant c in Algorithm 3 sufficiently large, one can ensure that $2\beta^2 \leq c$. Hence the local optimality condition in Algorithm 3 implies that

$$cost(S_J) \geq (1 - \frac{1}{n})cost(\mathcal{L})\,. \tag{6}$$

To upper bound the cost of S_J we use an assignment of the clients to the facilities in S_J. Consider a client c. There can be three cases: (i) c is nearer to a facility of $\mathcal{O}_J$ than the facilities in $(\mathcal{O} \setminus \mathcal{O}_J) \cup (\mathcal{L} \setminus \mathcal{L}_J)$, that is, $c \in C_1 = C_{\mathcal{O}}(\mathcal{O}_J) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_J))$, (ii) c is not in $C_{\mathcal{O}}(\mathcal{O}_J)$ and c is nearer to a facility of $\mathcal{L}_J$ than the facilities in $\mathcal{L} \setminus \mathcal{L}_J$, that is, $c \in C_2 = C_{\mathcal{L}}(\mathcal{L}_J) \setminus C_{\mathcal{O}}(\mathcal{O}_J)$, (iii) c does not appear in case (i) and (ii), that is, $c \in C_3 = C \setminus (C_1 \cup C_2)$. Note that if $c \in C_{\mathcal{L}}(\mathcal{L}_J)$, then $c \in C_1 \cup C_2$. Thus if $c \in C_3$, its nearest neighbor in $\mathcal{L}$ must be in $\mathcal{L} \setminus \mathcal{L}_J$. Also note that for a client $c \in C_2$, it is the case that $c(\mathcal{L}) \in \mathcal{L}_i \subseteq \mathcal{L}_J$ and $c(\mathcal{O}) \in \mathcal{O}_j \subseteq (\mathcal{O} \setminus \mathcal{O}_J)$ for some $1 \leq i \neq j \leq I$. Thus Lemma 7 is applicable to c, and $g(c) \in T_i \cup (\mathcal{Z}_i \cap B_i) \subseteq T_J$.

Now we describe the assignment. For a client in C_1, assign it to a facility in $\mathcal{O}_J$ nearest to it. For a client $c \in C_2$, use the assignment g in Lemma 7. For a client in C_3, assign it to a facility in $\mathcal{L} \setminus \mathcal{L}_J$ nearest to it. Thus by Inequality 6,

$$\sum_{c \in C_1} c_O + \sum_{c \in C_2} ||c - g(c)||^2 + \sum_{c \in C_3} c_L \geq (1 - \frac{1}{n}) \sum_{c \in C} c_L$$

$$\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_o)} (c_L - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})\,.$$

The last inequality follows by noting that $C_1 \cup (C_2 \cap C_l) = C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)$. Adding over all $J \in \mathcal{P}$ we get,

$$\sum_{J \in \mathcal{P}} \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)} (c_O - c_L) \geq -\frac{|\mathcal{P}|}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C}(c_O - c_L) \geq -\frac{|\mathcal{P}|}{n} cost(\mathcal{L}) .$$

From Observation 4, it follows that $|\mathcal{P}| \leq I = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)) = O(\varepsilon k)$. Thus,

$$cost(\mathcal{L}) \leq (1 + O(\varepsilon)) cost(\mathcal{O}) .$$ ◀

As mentioned before, the running time of Algorithm 1 is polynomial for fixed d, and hence we have established the following theorem.

▶ **Theorem 12.** *There is a polynomial time local search algorithm for k-means that uses $(1 + O(\varepsilon))k$ facilities and returns a solution with cost at most $(1 + O(\varepsilon))$ times the cost of the optimal k-means solution.*

Next we present the proof of Lemma 10.

3.2.1 Proof of Lemma 10

For purposes of exposition we introduce some more notation. For a set $r \subseteq \mathcal{L} \cup \mathcal{O} \cup T$, let $u(r) = |\mathcal{L} \cap r| - |(\mathcal{O} \cup T) \cap r|$. For a collection Ψ of sets, let $u(\Psi) = \sum_{r \in \Psi} |\mathcal{L} \cap r| - |(\mathcal{O} \cup T) \cap r|$. Using this notation we rewrite the statement of Lemma 10 as follows.

▶ **Lemma 13.** *Consider the collection $R = \{R_1, \ldots, R_I\}$ of sets with $R_j = \mathcal{L}_j \cup \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ and $|R_j| \leq \frac{\beta}{\varepsilon^d}$ for $1 \leq j \leq I$, where β is the constant in Observation 4. There exists a collection $\mathcal{P} = \{P_1, \ldots, P_p\}$, with $P_i \subseteq R$ for $1 \leq i \leq p$, $P_i \cap P_j = \emptyset$ for any $1 \leq i < j \leq p$, and $\cup_{i=1}^{p} P_i = R$, which satisfies the following properties:*
1. *$|P_i| \leq \frac{2\beta}{\varepsilon^d}$ for $1 \leq i \leq p$, where β is the constant in Observation 4;*
2. *$u(P_i) \geq 0$ for $1 \leq i \leq p$.*

Proof. Note that for each j, $u(R_j) \in [-\frac{\beta}{\varepsilon^d}, \frac{\beta}{\varepsilon^d}]$, as $|R_j| \leq \frac{\beta}{\varepsilon^d}$. Now by Observation 4, $\sum_{j=1}^{I} |T_j \cup (\mathcal{Z}_j \cap B_j)| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5 \leq \varepsilon((1 + 5\varepsilon)k + k)/5 \leq 2\varepsilon k$. Thus,

$$u(R) \geq |\mathcal{L}| - |\mathcal{O}| - \sum_{j=1}^{I} |T_j \cup (\mathcal{Z}_j \cap B_j)| \geq (1 + 5\varepsilon)k - k - 2\varepsilon k \geq 3\varepsilon k.$$

Now we describe the construction of the collection $\mathcal{P}$. For any j, if $u(R_j)$ equals 0, we add $\{R_j\}$ to $\mathcal{P}$ as an element. Note that such an element satisfies the desired properties. Now consider all the sets $R_j \in R$ such that $|u(R_j)| \geq 1$. Denote by R' the collection of such sets. Note that $u(R') = u(R)$. We process R' using the construction in Algorithm 4.

In each iteration of the outer while loop in line 2, we remove at most $l = \frac{2\beta}{\varepsilon^d}$ sets from R' and add the collection Ψ of these sets to $\mathcal{P}$ as an element. These l sets are chosen carefully so that $u(\Psi)$ is non-negative. To ensure this, at first $l/2$ sets are chosen to get the collection Ψ' such that $-l/2 \leq u(\Psi') \leq l/2$. Then we add at most $l/2$ more sets $\{r\}$ with $u(r) > 0$ (while loop in lines 17-19) to obtain a collection Ψ with $u(\Psi) \geq 0$. Assuming the loop invariant $u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such r must exist. Later we will argue that this loop invariant holds. This ensures that the algorithm exits the while loop in lines 17-19 with a Ψ such that $u(\Psi) \geq 0$. Also we stop the addition of sets as soon as $u(\Psi)$ becomes non-negative. This ensures that $u(\Psi) \leq \frac{\beta}{\varepsilon^d}$. Note that $|\Psi| \leq l = \frac{2\beta}{\varepsilon^d}$. Thus Ψ satisfies all the desired properties.

Now consider the selection of the $l/2$ sets of Ψ'. We select these sets sequentially, one in each iteration of the for loop in lines 4-15. Consider a particular iteration of this for loop. There can be two cases: (i) $u(\Psi') \geq 0$, and (ii) $u(\Psi') < 0$. In case (i) if there is a set r with

Algorithm 4

1: $l = \frac{2\beta}{\varepsilon^d}$
2: **while** $|R'| > l$ **do**
3: $\Psi' \leftarrow \{r\}$, where r is any element in R' with $u(r) > 0$; $R' \leftarrow R' \setminus \{r\}$
4: **for** $i = 1$ to $l/2$ **do**
5: **if** $u(\Psi') \geq 0$ **then**
6: **if** $u(r) > 0$ for each $r \in R'$ **then**
7: Add Ψ' to $\mathcal{P}$
8: **for each** $r \in R'$ **do**
9: Add $\{r\}$ to $\mathcal{P}$; $R' \leftarrow R' \setminus \{r\}$
10: **return**
11: $r \leftarrow$ any element in R' with $u(r) < 0$
12: $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
13: **else if** $u(\Psi') < 0$ **then**
14: $r \leftarrow$ any element in R' with $u(r) > 0$
15: $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
16: $\Psi \leftarrow \Psi'$
17: **while** $u(\Psi) < 0$ **do**
18: $r \leftarrow$ any element in R' with $u(r) > 0$
19: $\Psi \leftarrow \Psi \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
20: Add Ψ to $\mathcal{P}$
21: Add R' to $\mathcal{P}$

$u(r) < 0$, we choose it. If there is no such set r, we add Ψ' to $\mathcal{P}$ and for any set $r \in R'$, $\{r\}$ is added to $\mathcal{P}$ as an element. The algorithm terminates. In case (ii) we choose a set r with $u(r) > 0$. Assuming the loop invariant $u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such an r must exist. Hence in both cases we can ensure that at the end of each iteration of the for loop in lines 4-15 $u(\Psi') \in [-\frac{\beta}{\varepsilon^d}, \frac{\beta}{\varepsilon^d}]$.

Let M denote the number of iterations of the outer while loop. Then in each step except the last, we remove at least $l/2$ sets from R'. Since R' has at most I sets initially, $(M-1)\frac{l}{2} \leq I \Rightarrow M \leq \frac{2I}{l} + 1$.

Now we argue that after iteration $0 \leq j \leq M$,

$$u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2}. \tag{7}$$

Since $(\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2} \geq (\frac{2I}{l} + 1)\frac{l}{2} - M\frac{l}{2} \geq 0$, this would imply $u(R') \geq 0$ after iteration $j \leq M$. This establishes the loop invariant and also shows that the set R' added to $\mathcal{P}$ in line 21 has $u(R') \geq 0$, completing the proof of the lemma.

We now show (7) by induction. The inequality is true for $j = 0$, since before iteration 1,

$$u(R') = u(R) \geq 3\varepsilon k \geq I + \frac{\beta}{\varepsilon^d} = (\frac{2I}{l} + 1)\frac{l}{2}.$$

Consider a j such that $1 \leq j \leq M$ and suppose (7) is true after iteration $j-1$. Then at the beginning of iteration j, we have $u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - (j-1)\frac{l}{2}$. If the condition in line 6 is true in iteration j, then the algorithm terminates. Since R' becomes empty after this iteration, (7) trivially holds. If in iteration j we add Ψ to $\mathcal{P}$ in line 20, then $u(\Psi) \leq \frac{\beta}{\varepsilon^d} = \frac{l}{2}$. Thus, after iteration j,

$$u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - (j-1)\frac{l}{2} - \frac{l}{2} = (\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2}.$$ ◀

References

1. Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009. doi:10.1007/s10994-009-5103-0.
2. Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.
3. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC'98, pages 106–113, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276718.
4. Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.
5. Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318, 2010. doi:10.1109/FOCS.2010.36.
6. Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 754–767, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.
7. Vijay V. S. P. Bhattiprolu and Sariel Har-Peled. Separating a voronoi diagram via local search. *CoRR*, abs/1401.0174, 2014. URL: http://arxiv.org/abs/1401.0174.
8. Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997. doi:10.1016/S0169-7552(97)00031-7.
9. Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.
10. Vincent Cohen-Addad and Claire Mathieu. Effectiveness of local search for geometric optimization. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 329–343, 2015. doi:10.4230/LIPIcs.SOCG.2015.329.
11. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990. doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.
12. Richard O Duda, Peter E Hart, et al. Pattern classification and scene analysis. *J. Wiley and Sons*, 1973.
13. Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *J. Intell. Inf. Syst.*, 3(3/4):231–262, 1994. doi:10.1007/BF00962238.
14. Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
15. Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008. URL: http://arxiv.org/abs/0809.2554.
16. Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007. doi:10.1007/s00454-006-1271-x.

17 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300, 2004. doi:10.1145/1007352.1007400.

18 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based *k*-clustering (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*, pages 332–339, 1994. doi:10.1145/177424.178042.

19 Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000. doi:10.1109/34.824819.

20 Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999. doi:10.1145/331499.331504.

21 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. doi:10.1016/j.comgeo.2004.03.003.

22 L. Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

23 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007. doi:10.1137/S0097539702404055.

24 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 1374–1385, 2005. doi:10.1007/11523468_111.

25 S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982. doi:10.1109/TIT.1982.1056489.

26 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The planar k-means problem is np-hard. *Theor. Comput. Sci.*, 442:13–21, 2012. doi:10.1016/j.tcs.2010.05.034.

27 Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k means. *CoRR*, abs/1507.04227, 2015. URL: http://arxiv.org/abs/1507.04227.

28 J. Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84. doi:10.1007/s004540010019.

29 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

30 Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28, 2012. doi:10.1145/2395116.2395117.

31 Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991. doi:10.1007/BF00130487.

Incremental Voronoi diagrams

Sarah R. Allen[*1], **Luis Barba**[2], **John Iacono**[3], **and Stefan Langerman**[†4]

1 **Computer Science Department, Carnegie Mellon University, Pittsburgh, USA**
srallen@cs.cmu.edu

2 **Départment d'Informatique, Université Libre de Bruxelles, Brussels, Belgium; and**
School of Computer Science, Carleton University, Ottawa, Canada
lbarbafl@ulb.ac.be

3 **Tandon School of Engineering, New York University, New York, USA**
iacono@nyu.edu

4 **Départment d'Informatique, Université Libre de Bruxelles, Brussels, Belgium**
stefan.langerman@ulb.ac.be

Abstract

We study the amortized number of combinatorial changes (edge insertions and removals) needed to update the graph structure of the Voronoi diagram $\mathrm{VD}(S)$ (and several variants thereof) of a set S of n sites in the plane as sites are added to the set. To that effect, we define a general update operation for planar graphs that can be used to model the incremental construction of several variants of Voronoi diagrams as well as the incremental construction of an intersection of halfspaces in $\mathbb{R}^3$. We show that the amortized number of edge insertions and removals needed to add a new site to the Voronoi diagram is $O(\sqrt{n})$. A matching $\Omega(\sqrt{n})$ combinatorial lower bound is shown, even in the case where the graph representing the Voronoi diagram is a tree. This contrasts with the $O(\log n)$ upper bound of Aronov et al. (2006) for farthest-point Voronoi diagrams in the special case where points are inserted in clockwise order along their convex hull.

We then present a semi-dynamic data structure that maintains the Voronoi diagram of a set S of n sites in convex position. This data structure supports the insertion of a new site p (and hence the addition of its Voronoi cell) and finds the asymptotically minimal number K of edge insertions and removals needed to obtain the diagram of $S \cup \{p\}$ from the diagram of S, in time $O(K \operatorname{polylog} n)$ worst case, which is $O(\sqrt{n} \operatorname{polylog} n)$ amortized by the aforementioned combinatorial result.

The most distinctive feature of this data structure is that the graph of the Voronoi diagram is maintained explicitly at all times and can be retrieved and traversed in the natural way; this contrasts with other known data structures supporting nearest neighbor queries. Our data structure supports general search operations on the current Voronoi diagram, which can, for example, be used to perform point location queries in the cells of the current Voronoi diagram in $O(\log n)$ time, or to determine whether two given sites are neighbors in the Delaunay triangulation.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Voronoi diagrams, dynamic data structures, Delaunay triangulation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.15

[*] Supported by NSF grants CCF-0747250, CCF-1116594, and the Graduate Research Fellowship Program under Grant No. DGE-1252522.

[†] Directeur de Recherches du F.R.S.-FNRS.

© Sarah R. Allen, Luis Barba, John Iacono, and Stefan Langerman;
licensed under Creative Commons License CC-BY

1 Introduction

Let S be a set of n sites in the plane. The graph structures of the Voronoi diagram $\mathrm{VD}(S)$ and its dual the Delaunay triangulation $\mathrm{DT}(S)$ capture much of the proximity information of that set. They contain the nearest neighbor graph, the minimum spanning tree, and the Gabriel graph of S, and have countless applications in computational geometry, shape reconstruction, computational biology, and machine learning.

One of the most popular algorithms for constructing a Voronoi diagram inserts sites in random order, incrementally updating the diagram [8]. In that case, backward analysis shows that the expected number of changed edges in $\mathrm{VD}(S)$ is constant, offering some hope that an efficient dynamic (or at least semi-dynamic) data structure for maintaining $\mathrm{VD}(S)$ could exist. These hopes, however, are rapidly squashed, as it is easy to construct examples where the complexity of each successively added face is $\Omega(n)$, and thus each insertion changes the position of $\Omega(n)$ vertices and edges of $\mathrm{VD}(S)$. The goal of this paper is to show that despite this worst-case behavior, the amortized number of *structural changes* to the graph of the Voronoi diagram of S, i.e., the minimum number of edge insertions and deletions needed to update $\mathrm{VD}(S)$ throughout a sequence of site insertions to S, is much smaller.

This might come as a surprise in light of the fact that the number of combinatorial changes (usually modeled as *flips*) to the Delaunay triangulation of S upon the insertion of a point can be $\Omega(n)$ with each insertion, even when the sites are in convex position and are added in clockwise order. (Note that in that case the Voronoi diagram of S is a tree and the standard flip operation is a rotation in the tree.)

To overcome this worst-case behavior, Aronov et al. [2] studied what happens in this specific case (points in convex position added in clockwise order) if the rotation operation is replaced by the more elementary link (add an edge) and cut (delete an edge) operations in the tree. They show that, in that model, it is possible to reconfigure the tree after each site insertion while performing $O(\log n)$ links and cuts, amortized; however their proof is existential and no algorithm is provided to find those links and cuts. Pettie [13] shows both an alternate proof of that fact using forbidden 0-1 matrices and a matching lower bound.

One important application of Voronoi diagrams is to solve *nearest-neighbor* (or *farthest-neighbor*) queries: given a point in the plane, find the site nearest (or farthest) to this point. In the static case, this is done by preprocessing the (nearest or farthest point) Voronoi diagram to answer point-location queries in $O(\log n)$ time. Without the need to maintain $\mathrm{VD}(S)$ explicitly, the problem of nearest neighbor queries is a *decomposable search problem* and can be made semi-dynamic using the standard dynamization techniques of Bentley and Saxe [4]. The best incremental data structure supporting nearest-neighbor queries performs queries and insertions in $O(\log^2 n/\log\log n)$ time [7, 12]. Recently, Chan [6] developed a randomized data structure supporting nearest-neighbor queries in $O(\log^2 n)$ time, insertions in $O(\log^3 n)$ expected amortized time, and deletions in $O(\log^6 n)$ expected amortized time.

Flarbs. In the mid-1980's it was observed that a number of variants of Voronoi diagrams and Delaunay triangulations using different metrics (Euclidean distance, L_p norms, convex distance functions) or different kinds of sites (points, segments, circles) could all be handled using similar techniques. To formalize this, several abstract frameworks were defined, such as the one of Edelsbrunner and Seidel [9] and the two variants of abstract Voronoi diagrams of Klein [11, 10]. In this paper we define a new abstract framework to deal with Voronoi diagrams constructed incrementally by inserting new sites.

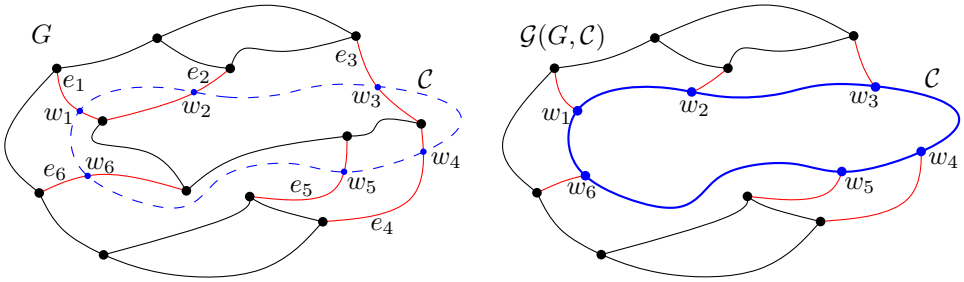


Figure 1 The flarb operation on a graph G induced by a flarbable curve $\mathcal{C}$, produces a graph $\mathcal{G}(G, \mathcal{C})$ with 2 more vertices. Fleeq-edges crossed by $\mathcal{C}$ are shown in red.

Let G be a 3-regular embedded plane graph with n vertices[1]. We seek to bound the number of edge removals and insertions needed to implement the following operation, hereafter referred to as a *flarb*[2]: Given a simple closed curve $\mathcal{C}$ in the plane whose interior intersects G in a connected component, split both $\mathcal{C}$ and all the edges that it crosses at the point of intersection, remove every edge and vertex that lies in the interior of $\mathcal{C}$, and add each curve in the subdivision of $\mathcal{C}$ as a new edge; see Figure 1. This operation can be used to represent the insertion of new cells in different types of Voronoi diagrams. It can also be used to represent the changes to the 1-skeleton of a polyhedron in $\mathbb{R}^3$ after it is intersected with a halfspace.

Results. We show that the amortized *cost* of a flarb operation, where the combinatorial cost is defined to be the minimum number of edge insertions and removals needed to perform it, is $O(\sqrt{n})$. We also show a matching lower bound: some sequences of flarbs require $\Omega(\sqrt{n})$ links and cuts per flarb, even when the graph is a tree (or more precisely a Halin graph – a tree with all leaves connected by a cycle to make it 3-regular). This contrasts with the $O(\log n)$ upper bound of Aronov et al. [2] for the Voronoi diagram of points in convex position (also a tree) when points are added in clockwise order.

We complement these combinatorial bounds with an algorithmic result. We present an output-sensitive data structure that maintains the nearest- or farthest-point Voronoi diagram of a set S of n points in convex position as new points are added to S. Upon an insertion, the data structure finds the minimum number K (up to within a constant factor) of edge insertions and deletions necessary to update the Voronoi diagram of S. The running time of each insertion is $O(K \log^7 n)$, and by our combinatorial bounds, $K = O(\sqrt{n})$. This solves the open problem posed by Aronov et al. [2].

The distinguishing feature of this data structure is that it explicitly maintains the graph structure of the Voronoi diagram after every insertion, a property that is not provided by any nearest neighbor data structure that uses decomposable searching problem techniques. Further, the data structure also maintains the Voronoi diagram in a *grappa tree* [2], a variant of the link-cut trees of Sleator and Tarjan [14] that allows a powerful query operation called *oracle-search*. Roughly speaking, the oracle-search query has access to an oracle specifying a vertex to find. Given an edge of the tree, the oracle determines which of the two subtrees attached to its endpoints contains that vertex. Grappa trees use $O(\log n)$ time and oracle

[1] While the introduction used n for the number of sites in S, the combinatorial part of this article uses n for the number of vertices in the Voronoi diagram. By Euler's formula, those two values are asymptotically equivalent, up to a constant factor.

[2] Although the last two authors are honored by the flattering renaming of the flarb operation in the literature [13], this paper uses original terminology.

calls to find the sought vertex. A grappa tree is in some sense a dynamic version of the centroid decomposition for trees, which is used in many algorithms for searching in Voronoi diagrams. Using this structure, it is possible to solve a number of problems for the set S at any moment during the incremental construction, for example:

- Report whether two sites p and q are connected by a Delaunay edge in $O(\log n)$ time.
- Given a point q, find the Voronoi cell containing q in $O(\log n)$ time. This not only gives the nearest neighbor of q, but a pointer to the explicit description of its cell.
- Find the smallest disk enclosing S centered on a query segment $[pq]$, in $O(\log n)$ time [5].
- Find the smallest disk enclosing S, centered on a query circle C, in $O(\log n)$ time [3].
- Given a convex polygon P (counterclockwise array of its m vertices), find the smallest disk enclosing S and excluding P in $O(\log n + \log m)$ time [1].

The combinatorial bound for Voronoi diagrams also has direct algorithmic consequences, the most important being that it is possible to store all versions of this graph throughout a sequence of insertions using persistence in $O(n^{3/2})$ space. Since the entire structure of the graph is stored for each version, this provides a foundation for many applications that, for instance would require searching the sequence of insertions for the moment during which a specific event occurred.

Outline. The main approach used to bound the combinatorial cost of a flarb is to examine how the complexity of the faces changes. Notice that faces whose size remains the same do not require edge insertions and deletions. The other faces either grow or shrink, and a careful counting argument reveals that the cost of a flarb is at most the number faces that shrink (or disappear) upon execution of the flarb (Section 2). By using a potential function that sums the sizes of all faces, the combinatorial cost of shrinking faces is paid for by the reduction of their potential. To avoid incurring a high increase in potential for a large new face, the potential of each face is capped at $\sqrt{n}$. Then at most $O(\sqrt{n})$ large faces can shrink without changing potential and are accounted for separately (Section 3). The matching $\Omega(\sqrt{n})$ lower bound is presented in Section 4, and Section 5 presents the data structure for performing flarbs for the Voronoi diagrams of points in convex position.

2 The flarb operation

In this section we formalize the flarb operation that models the insertion of new sites in Voronoi diagrams, and present a preliminary analysis of the cost of a flarb.

Let $G = (V, E)$ be a planar 3-regular graph embedded in $\mathbb{R}^2$ (not-necessarily with a straight-line embedding). Let $\mathcal{C}$ be a simple closed Jordan curve in the plane. Define $\textsc{in}(\mathcal{C})$ to be the set of vertices of G that lie in the interior of $\mathcal{C}$ and let $\textsc{ex}(\mathcal{C}) = V \setminus \textsc{in}(\mathcal{C})$. We say that $\mathcal{C}$ is *flarbable* for G if the following conditions hold: (1) the graph induced by $\textsc{in}(\mathcal{C})$ is connected, (2) $\mathcal{C}$ intersects each edge of G either at a single point or not at all, (3) $\mathcal{C}$ passes through no vertex of G, and (4) the intersection of $\mathcal{C}$ with each face of G is path-connected.

In the case where the graph G is clear from context, we simply say that $\mathcal{C}$ is flarbable. The *fleeq* of $\mathcal{C}$ is the circular sequence $\mathcal{E}_\mathcal{C} = e_1, \ldots, e_k$ of edges in E that are crossed by $\mathcal{C}$; we call the edges in $\mathcal{E}_\mathcal{C}$ *fleeq-edges*. A face whose interior is crossed by $\mathcal{C}$ is called a *$\mathcal{C}$-face*. We assume without loss of generality that $\mathcal{C}$ is oriented clockwise and that the edges in $\mathcal{E}_\mathcal{C}$ are ordered according to their intersection with $\mathcal{C}$. Given a flarbable curve $\mathcal{C}$ on G, we present the following definition.

▶ **Definition 1.** For a planar graph G and a curve $\mathcal{C}$ that is flarbable for G, we define a *flarb* operation $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$ which produces a new 3-connected graph $\mathcal{G}(G, \mathcal{C})$ as follows (see Figure 1 for a depiction): (1) For each edge $e_i = (u_i, v_i)$ in $\mathcal{E}_\mathcal{C}$ such that $u_i \in \text{IN}(\mathcal{C})$ and $v_i \in \text{EX}(\mathcal{C})$, create a new vertex $w_i = \mathcal{C} \cap e_i$ and connect it to v_i along e_i. (2) For each pair e_i, e_{i+1} of successive edges in $\mathcal{E}_\mathcal{C}$, create a new edge (w_i, w_{i+1}) between them along $\mathcal{C}$. We call (w_i, w_{i+1}) a $\mathcal{C}$*-edge* (all indices are taken modulo k). (3) Delete all vertices of $\text{IN}(\mathcal{C})$ along with their incident edges.

Note that at most two new vertices are created. Since each newly created vertex has degree three and all remaining vertices are unaffected, the new graph is 3-regular. In other words, the flarb operation $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$ creates a cycle along $\mathcal{C}$ and removes the portion of the graph enclosed by $\mathcal{C}$. Note that for any point set in general position (no four points lie on the same circle), its Voronoi diagram is a 3-regular planar graph, assuming we use the line at infinity to join the endpoints of its unbounded edges in clockwise order. Therefore, a flarb can be used to represent the changes to the Voronoi diagram upon insertion of a new site.

▶ **Observation 2.** *Given a set S of points in general position, let $\mathcal{V}(S)$ be the graph of the Voronoi diagram of S. For a new point q, there exists some curve $\mathcal{C}_S^q$ such that $\mathcal{G}(\mathcal{V}(S), \mathcal{C}_S^q) = \mathcal{V}(S \cup \{q\})$; namely, $\mathcal{C}_S^q$ is the boundary of the Voronoi cell of q in $\mathcal{V}(S \cup \{q\})$.*

More generally, convex polytopes defined by the intersection of halfspaces in $\mathbb{R}^3$ behave similarly: the intersection of a new halfspace with a convex polytope modifies the structure of its 1-skeleton by adding a new face. This structural change can be obtained performing a flarb operation in which the flarbable curve consists of the boundary of the new face.

▶ **Definition 3.** Given a $\mathcal{C}$-face f of G, the *modified face* of f is the face f' of $\mathcal{G}(G, \mathcal{C})$ that coincides with f outside of $\mathcal{C}$. In other words, f' is the face that remains from f after performing the flarb $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$. We say that a $\mathcal{C}$-face f is *preserved* (by the flarb $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$) if $|f| = |f'|$. Moreover, we say that each edge in a preserved face is *preserved* (by $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$). Denote by $\mathcal{P}(G, \mathcal{C})$ the set of faces preserved by $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$ and let $\mathcal{B}(G, \mathcal{C})$ be the set of faces wholly contained in the interior of $\mathcal{C}$.

Since a preserved $\mathcal{C}$-face bounded by two fleeq-edges e_i and e_{i+1} has the same size before and after the flarb, there must be an edge e of G connecting e_i with e_{i+1} which is replaced by a $\mathcal{C}$-edge e^* after the flarb. In this case, we say that the edge e *reappears* as e^*.

The following auxiliary lemma will help us bound the number of operations needed to produce the graph $\mathcal{G}(G, \mathcal{C})$, and follows directly from the Euler characteristic of connected planar graphs:

▶ **Lemma 4.** *Let H be a connected planar graph with vertices of degree either 1, 2 or 3. For each $i \in \{1, 2, 3\}$, let δ_i be the number of vertices of H with degree i. Then, H has exactly $2\delta_1 + \delta_2 + 3F_H - 3$ edges, where F_H is the number of bounded faces of H.*

Given a 3-regular graph $G = (V, E)$ and a flarbable curve $\mathcal{C}$ we want to analyze the number of structural changes that G needs to undergo to perform $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$. To this end, we define the combinatorial *cost* of $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$, denoted by $\text{COST}(G, \mathcal{C})$, to be the minimum number of links and cuts needed to transform G into $\mathcal{G}(G, \mathcal{C})$ (note that an algorithm may not implement the flarb operation exactly as described in Definition 1). We assume that any other operation has no cost and is hence not included in the cost of the flarb.

Consider the fleeq $\mathcal{E}_\mathcal{C} = e_1, \ldots, e_k$ and the $\mathcal{C}$-edges created by $\mathcal{F}(G, \mathcal{E}_\mathcal{C})$. Let e be an edge adjacent to some e_i and e_{i+1} that reappears as the $\mathcal{C}$-edge e^*. Notice that we can obtain e^* without any links or cuts to G: simply shrink e_i and e_{i+1} so that their endpoints in $\text{IN}(\mathcal{C})$

now coincide with their intersections with $\mathcal{C}$. Then modify e to coincide with the portion of $\mathcal{C}$ connecting the new endpoints of e_i and e_{i+1}. Using this *preserving operation*, we obtain the $\mathcal{C}$-edge e^* with no cost to the flarb. Intuitively, preserved edges are cost-free in a flarb while non-preserved edges have a nonzero cost. This notion is formalized as follows.

▶ **Lemma 5.** *For a flarbable curve $\mathcal{C}$, $(|\mathcal{E}_\mathcal{C}| + |\mathcal{B}(G,\mathcal{C})| - |\mathcal{P}(G,\mathcal{C})|)/2 \leq \textsc{cost}(G,\mathcal{C})$ and $\textsc{cost}(G,\mathcal{C}) \leq 4|\mathcal{E}_\mathcal{C}| + 3|\mathcal{B}(G,\mathcal{C})| - 4|\mathcal{P}(G,\mathcal{C})|$.*

3 The combinatorial upper bound

In this section, we define a potential function to bound the amortized cost of each operation in a sequence of flarb operations. For a 3-regular embedded planar graph $G = (V, E)$, we define two potential functions: a local potential function μ to measure the potential of each face, and a global potential function Φ to measure the potential of the whole graph.

▶ **Definition 6.** Let F be the set of faces of a 3-regular embedded planar graph $G = (V, E)$. For each face $f \in F$, let $\mu(f) = \min\{\lceil\sqrt{|V|}\rceil, |f|\}$, where $|f|$ is the number of edges on the boundary of f. The potential $\Phi(G)$ of G is defined as follows: $\Phi(G) = \lambda\sum_{f\in F}\mu(f)$, for some sufficiently large positive constant λ to be defined later.

Recall that the potential $\mu(f)$ of a $\mathcal{C}$-face f remains unchanged as long as $|f|, |f'| \geq \sqrt{|V|}$, where f' is the modified face of f after the flarb. Since there is no change in potential that we can use within large $\mathcal{C}$-faces, we exclude them from our analysis and focus only on smaller $\mathcal{C}$-faces. However, by excluding these large faces, we effectively split $\mathcal{C}$ into smaller curves that cross only $\mathcal{C}$-faces with less than $\sqrt{|V|}$ edges. We formalize this notion in the following section.

3.1 Flarbable sub-curves

Given a flarbable curve $\mathcal{C}$, a (connected) curve $\gamma \subseteq \mathcal{C}$ is a *flarbable sub-curve*. Let $\epsilon_\gamma = e_1, \ldots, e_k$ (or simply ϵ) be the set of fleeq-edges intersected by γ given in order of intersection after orienting γ arbitrarily. We call ϵ the *subfleeq* induced by γ. We say that a face is a γ-face if two of its fleeq-edges are crossed by γ (if γ has an endpoint in the interior of this face, it is not a γ-face).

Consider the set of all edges of G intersected or enclosed by $\mathcal{C}$ that bound some γ-face. Since $\mathcal{E}_\mathcal{C}$ is flarbable, these edges induce a connected subgraph Y_γ of G with $|\epsilon| = k$ leaves (vertices of degree 1), namely the endpoints outside of $\mathcal{C}$ of each fleeq-edge in ϵ; see Figure 2. Notice that Y_γ may consist of some bounded faces contained in the interior of $\mathcal{C}$. Let H_γ be the set of bounded faces of Y_γ and let δ_2 be the number of vertices of degree 2 of Y_γ. Since Y_γ consists of k vertices of degree 1, Lemma 4 implies the following result.

▶ **Corollary 7.** *The graph Y_γ consists of exactly $2k + \delta_2 + 3|H_\gamma| - 3$ edges.*

Recall that a $\mathcal{C}$-face f is preserved if its corresponding modified face f' in $\mathcal{G}(G,\mathcal{C})$ has the same number of edges, i.e., $|f'| = |f|$. We say that f is *augmented* if $|f'| = |f| + 1$ and we call f *shrinking* if $|f'| < |f|$. Notice that these are all the possible cases as f gains at most one new edge during the flarb, namely the $\mathcal{C}$-edge crossing this face.

In the context of a particular flarbable sub-curve γ, let a_γ, s_γ and p_γ be the number of augmented, shrinking and preserved γ-faces, respectively (or simply a, s and p if γ is clear from the context). We further differentiate among the s shrinking γ-faces. A shrinking γ-face is *interior* if it contains no vertex of degree 2 of Y_γ and does not share an edge with

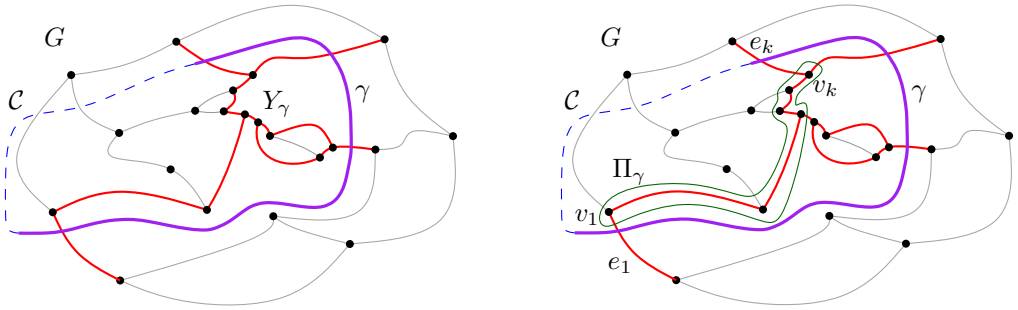


Figure 2 Left: A flarbable sub-curves γ is contained in a flarbable curve $\mathcal{C}$. The graph Y_γ is the union of all edges bounding a γ-face. Right: The path Π_γ connects the endpoints of the first and last fleeq-edges crossed by γ by going along the boundary of the outer-face of Y_γ.

an augmenting face. Let s_a be the number of shrinking γ-faces that share an edge with an augmented face, let s_b be the number of shrinking γ-faces not adjacent to an augmented face that have a vertex of degree 2 of Y_γ, and let s_c be the number of interior shrinking γ-faces. Therefore, $s = s_a + s_b + s_c$ is the total number of shrinking γ-faces.

Since each augmented face has at most two edges and because there are a augmented faces, we know that $s_a \leq 2a$. Let v_1 and v_k be the endpoints of the edges e_1 and e_k that lie inside $\mathcal{C}$. Let Π_γ be the unique path connecting v_1 and v_k in Y_γ that traverses along the boundary of the outer face of Y_γ and stays in the interior of $\mathcal{C}$; see Figure 2.

Notice that Π_γ contains all the edges of γ-faces that may bound a $\mathcal{C}$-face that is not crossed by γ. In the end, we aim to have bounds on the number of edges that will be removed from the γ-faces during the flarb, but some of these edges may be double counted if they are shared with other $\mathcal{C}$-faces. Therefore, we aim to bound the length of Π_γ and count precisely these possible double-counted edges.

▶ **Lemma 8.** *The path Π_γ has length at most $k + 3|H_\gamma| + \delta_2 - a - s_c$.*

3.2 How much do faces shrink in a flarb?

In order to analyze the effect of the flarb operations on flarbable sub-curves, we think of each edge as consisting of two *half-edges*, each adjacent to one of the two faces incident to this edge. For a given edge, the algorithm may delete its half-edges during two separate flarbs of different flarbable sub-curves.

We define the operation $\mathcal{F}(G, \gamma)$ to be the operation which executes steps 1 and 2 of the flarb on the flarbable sub-curve γ and then deletes each half-edge with both endpoints in $\text{IN}(\mathcal{C})$ adjacent to a γ-face. Since $\mathcal{F}(G, \gamma)$ removes and adds half-edges, we are interested in bounding the net balance of half-edges throughout the flarb. To do this, we measure the change in size of a face during the flarb.

Recall that a, s and p are the number of augmented, shrinking and preserved γ-faces, respectively. The following result provides a bound on the total "shrinkage" of the faces crossed by a given flarbable sub-curve.

▶ **Theorem 9.** *Given a flarbable curve $\mathcal{C}$ on G and a flarbable sub-curve γ crossing the fleeq-edges $\epsilon = e_1, \ldots, e_k$, let $f_1, \ldots, f_k$ be the sequence of γ-faces and let $f'_1, \ldots, f'_k$ be their corresponding modified faces after the flarb $\mathcal{F}(G, \gamma)$. Then, $\sum_{i=1}^{k}(|f_i| - |f'_i|) \geq s/2$.*

Proof. Recall that no successive γ-faces can both be augmented unless $\mathcal{E}_\mathcal{C}$ consists of three edges incident to a single vertex. In this case, at most 3 γ-faces can be augmented, so

$\sum_{i=1}^{k}(|f_i| - |f'_i|) = 3$ and the result holds trivially; hence, we assume from now on that no two successive faces are both augmented.

Let Δ be the number of half-edges removed during $\mathcal{F}(G, \gamma)$. Notice that to count how much a face f_i shrinks when becoming f'_i after the flarb, we need to count the number of half-edges of f_i that are deleted and the number that are added in f'_i. Since exactly one half-edge is added in each f'_i, we know that $\sum_{i=1}^{k}(|f_i| - |f'_i|) = \Delta - k$. We claim that $\Delta \geq k + s/2$. If this claim is true, then $\sum_{i=1}^{k}(|f_i| - |f'_i|) \geq s/2$ as stated in the theorem. In the remainder of this proof, we show this bound on Δ.

Let $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ be the subgraph of Y_γ obtained by removing its k fleeq-edges. Therefore, we know that $|E_{\mathcal{T}}| = k + 3|H_\gamma| + \delta_2 - 3$ by Corollary 7. To have a precise counting of Δ, notice that for some edges of $\mathcal{T}$, $\mathcal{F}(G, \gamma)$ removes only one of their half-edges and for others it will remove both of them. Since the fleeq-edges are present in each of the faces $f_1, \ldots, f_k$ before and after the flarb, we get that $\Delta = 2|E_{\mathcal{T}}| - S_{\mathcal{T}}$, where $S_{\mathcal{T}}$ denotes the number of edges in $\mathcal{T}$ with only one half-edge incident to a face of $f_1, \ldots, f_k$.

Note that the edges of $S_{\mathcal{T}}$ are exactly the edges on the path Π_γ bounded in Lemma 8. Therefore, $S_{\mathcal{T}} \leq k + 3|H_\gamma| + \delta_2 - a - s_c$. By using this bound, we get $\Delta \geq 2(k + 3|H_\gamma| + \delta_2 - 3) - (k + 3|H_\gamma| + \delta_2 - a - s_c) = k + 3|H_\gamma| + \delta_2 + a + s_c - 6$. Since each shrinking γ-face accounted for by s_b has a vertex of degree 2 in Y_γ, we know that $\delta_2 \geq s_b$. Moreover, $s_a \leq 2a$ as each shrinking γ-face can be adjacent to at most two augmenting γ-faces. Thus, since $s = s_a + s_b + s_c$, we get that $\Delta \geq k + 3|H_\gamma| + s_a/2 + s_b + s_c \geq k + s/2$, where s is the number of shrinking γ-faces proving the claimed bound on Δ. ◀

3.3 Flarbable sequences

Let $\mathcal{G}^0 = G$. A sequence of curves $\mathcal{C} = \mathcal{C}_1, \ldots, \mathcal{C}_k$ is *flarbable* if for each $i \in [k]$, $\mathcal{C}_i$ is a flarbable on $\mathcal{G}^i = \mathcal{G}(\mathcal{G}^{i-1}, \mathcal{C}_i)$. As a notational shorthand, let $\mathcal{F}^i$ denote the flarb operation $\mathcal{F}(\mathcal{G}^{i-1}, \mathcal{C}_i)$ when $\mathcal{C}$ is a flarbable sequence for G.

▶ **Theorem 10.** *For a 3-regular planar graph $G = (V, E)$ and some flarbable sequence $\mathcal{C} = \mathcal{C}_1, \ldots, \mathcal{C}_N$ of flarbable fleeqs, for all $i \in [N]$, $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) + \Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq O(\sqrt{|V_i|})$, where V_i is the set of vertices of $\mathcal{G}^i$.*

Proof. Partition $\mathcal{C}_i$ into smaller curves $\gamma_1, \ldots, \gamma_h$ such that for all $j \in [h]$, γ_j is a maximal curve contained in $\mathcal{C}_i$ that does not intersect the interior of a face with more than $\sqrt{|V_i|}$ edges. Since there can be at most $\sqrt{|V_i|}$ faces of size $\sqrt{|V_i|}$, we know that $h \leq \sqrt{|V_i|}$. Let ϵ_j be the subfleeq containing each fleeq-edge crossed by γ_j. Let a_j, s_j and p_j be the number of augmented, shrinking and preserved γ_j-faces, respectively. Notice that $|\epsilon_j| = a_j + s_j + p_j + 1$. Moreover, since each augmented face is adjacent to a shrinking face, we know that $a_j \leq s_j + 1$. Therefore, $|\epsilon_j| \leq 2s_j + p_j + 2$.

Let $\mathcal{L}_i$ be the set of $\mathcal{C}_i$-faces with at least $\sqrt{|V_i|}$ edges and let ω_i be the set of all faces of $\mathcal{G}^{i-1}$ completely enclosed in the interior of $\mathcal{C}_i$.

First, we upper bound $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i)$. By Lemma 5, we know that $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) \leq 4|\mathcal{E}_{\mathcal{C}_i}| + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)| - 4|\mathcal{P}(\mathcal{G}^{i-1}, \mathcal{C}_i)| = 4\sum_{j=1}^{h}|\epsilon_j| + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)| - 4|\mathcal{P}(\mathcal{G}^{i-1}, \mathcal{C}_i)| \leq 4\sum_{j=1}^{h}(2s_j + p_j + 2) + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)| - 4|\mathcal{P}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$. Because each preserved face is crossed by exactly one flarbable sub-curve, $\sum_{j=1}^{h} p_j = |\mathcal{P}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$. Therefore, $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) \leq 4\sum_{j=1}^{h}(2s_j + 2) + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)| = 8h + 8\sum_{j=1}^{h} s_j + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$. Since $h \leq \sqrt{|V_i|}$, we conclude that

$$cost[\mathcal{G}^{i-1}, \mathcal{C}_i] \leq 8\sqrt{|V_i|} + 8\sum_{j=1}^{h} s_j + 3|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|. \tag{1}$$

Next, we upper bound the change in potential $\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1})$. Given a flarbable curve or sub-curve γ, let $\mathcal{A}(\gamma)$ denote the set of γ-faces. Recall that for a γ-face $f \in \mathcal{A}(\gamma)$, f' is the modified face of f. Also, let f_n be the new face created by $\mathcal{F}^i$, i.e., the face of $\mathcal{G}^i$ bounded by $\mathcal{C}_i$. Recall that for each face $f \in \mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)$, f disappears and there is a drop in potential of $\mu(f) \geq 1$. Using this, we can break up the summation so that

$$\begin{aligned}\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) &= \mu(f_n) + \lambda \sum_{f \in \mathcal{A}(\mathcal{C}_i)} (\mu(f') - \mu(f)) - \lambda \sum_{f \in \mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)} \mu(f) \\ &\leq \mu(f_n) + \lambda \sum_{f \in \mathcal{A}(\mathcal{C}_i)} (\mu(f') - \mu(f)) - \lambda |\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)| \ .\end{aligned}$$

We now break up the first summation by independently considering the large faces in $\mathcal{L}_i$ and the remaining smaller faces which are crossed by some flarbable sub-curve. We get that $\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq \ \mu(f_n) + \lambda \sum_{j=1}^{h} \left(\sum_{f \in \mathcal{A}(\gamma_j)} (\mu(f') - \mu(f)) \right) + \lambda \sum_{f \in \mathcal{L}_i} (\mu(f') - \mu(f)) - \lambda |\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$.

Since each face can gain at most one edge, in particular we know that $\mu(f') - \mu(f) \leq 1$ for each $f \in \mathcal{L}_i$. Moreover, $\mu(f_n) \leq \sqrt{|V_i|}$ by the definition of μ. Thus, $\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq \sqrt{|V_i|} + \lambda \sum_{j=1}^{h} \left(\sum_{f \in \mathcal{A}(\gamma_j)} (\mu(f') - \mu(f)) \right) + \lambda |\mathcal{L}_i| - \lambda |\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$.

Note that $\mu(f) = |f|$ for each face $f \in \mathcal{A}(\gamma_j)$, $1 \leq j \leq h$. Thus, applying Theorem 9 to the first summation, we get $\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq \sqrt{|V_i|} - \frac{\lambda}{2} \sum_{j=1}^{h} s_j + \lambda |\mathcal{L}_i| - \lambda |\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$. Since there can be at most $\sqrt{|V_i|}$ faces of size $\sqrt{|V_i|}$, we know that $|\mathcal{L}_i| \leq \sqrt{|V_i|}$. Therefore, $\Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq (\lambda + 1)\sqrt{|V_i|} - \frac{\lambda}{2} \sum_{j=1}^{h} s_j - \lambda |\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$.

Putting this together with (1), we get that $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) + \Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) \leq (\lambda + 9)\sqrt{|V_i|} + (8 - \frac{\lambda}{2}) \sum_{j=1}^{h} s_j + (3 - \lambda)|\mathcal{B}(\mathcal{G}^{i-1}, \mathcal{C}_i)|$. By letting λ be a sufficiently large constant (namely $\lambda = 16$), we get that $\textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) + \Phi(\mathcal{G}^i) - \Phi(\mathcal{G}^{i-1}) = O(\sqrt{|V_i|})$. ◀

▶ **Corollary 11.** *Let G be a 3-regular plane graph with ν vertices. For a sequence $\mathcal{C} = \mathcal{C}_1, \ldots, \mathcal{C}_N$ of flarbable fleeqs for graph $G = (V, E)$ where $\nu = |V|$, $\sum_{i=1}^{N} \textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) = O(\nu + N\sqrt{\nu + N})$.*

Proof. Using the result of Theorem 10, we can write $\sum_{i=1}^{N} \textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) + \Phi(\mathcal{G}^N) - \Phi(G) = O(N\sqrt{|V_i|})$. Because $\Phi(G) = \lambda \sum_{f \in F} \mu(f)$, we know that $\Phi(G) = O(\nu)$. Analogously, since each flarb operation adds at most 2 vertices, we know that the number of vertices in $\mathcal{G}^N$ is $O(\nu + N)$ which, in turn, implies that $\Phi(\mathcal{G}^N) = O(\nu + N)$. Therefore, $\sum_{i=1}^{N} \textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) = O(N\sqrt{|V_i|} + \Phi(G) - \Phi(\mathcal{G}^N)) = O(\nu + N\sqrt{\nu + N})$. ◀

4 The lower bound

In Section 4, we present an example of a 3-regular Halin graph G with ν vertices – a tree with all leaves connected by a cycle to make it 3-regular – and a corresponding flarb operation with cost $\Omega(\sqrt{\nu})$ that yields a graph isomorphic to G. Because this sequence can be repeated, the amortized cost of a flarb is $\Theta(\sqrt{\nu})$.

Let $\nu = 2k(k + 1) - 2$ for some positive integer k. The construction of the 3-regular graph with ν vertices is depicted in Figure 3. In this graph, we show the existence of a flarbable curve $\mathcal{C}$ (dashed in the figure) such that the flarb operation on G produces a graph $\mathcal{G}(G, \mathcal{C})$ isomorphic to G. Moreover, $\mathcal{C}$ crosses at least k augmented $\mathcal{C}$-faces and k shrinking $\mathcal{C}$-faces. Therefore, $\textsc{cost}(G, \mathcal{C}) \geq k = \Omega(\sqrt{\nu})$ by Lemma 5. Since we end up with a graph that is isomorphic to the original, we can produce a new flarbable curve having

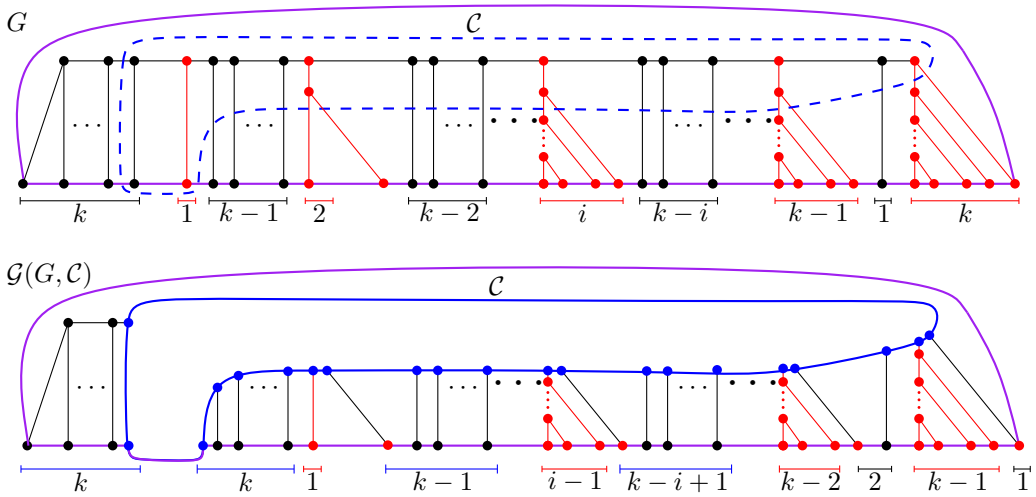


Figure 3 A 3-regular graph G with $\nu = 2k(k+1) - 2$ vertices. A flarbable curve $\mathcal{C}$ induces a flarb such that $\mathcal{G}(G, \mathcal{C})$ is isomorphic with G.

the same effect. That is, there is a sequence of N flarbable curves $\mathcal{C}_1, \ldots, \mathcal{C}_N$ such that $\sum_{i=1}^{N} \textsc{cost}(\mathcal{G}^{i-1}, \mathcal{C}_i) = \Omega(N\sqrt{\nu})$,

5 Computing the flarb

In this section, we describe a data structure to maintain the Voronoi diagram of a set S of n sites in convex position as new sites are added to S. Our structure allows us to find the edges of each preserved face and ignore them, thereby focusing only on necessary modifications to the combinatorial structure. The time we spend in these operations is then proportional to the number of non-preserved edges. Since this number is proportional to the cost of the flarb, our data structure supports site insertions in time that is almost optimal (up to a polylogarithmic factor).

Grappa trees. Grappa trees [2] are a modification of link-cut trees, a data structure introduced by Sleator and Tarjan [14] to maintain the combinatorial structure of trees. They support the creation of new isolated vertices, the *link* operation which adds an edge between two vertices in disjoint trees, and the *cut* operation which removes an edge, splitting a tree into two trees.

We use Grappa trees to maintain the combinatorial structure of the incrementally constructed Voronoi diagram $\mathcal{V}(S)$ of a set S of sites in convex position. Recall that each insertion defines a flarbable curve $\mathcal{C}$, namely the boundary of the Voronoi cell of the inserted site. Our algorithm performs this flarb operation in time $O(\textsc{cost}(\mathcal{V}(S), \mathcal{C}) \log^7 n)$, where n is the number of vertices inserted so far. That is, we obtain an algorithm whose running time depends on the minimum number of link and cut operations that the Voronoi diagram must undergo after each insertion. Moreover, this Voronoi diagram answers nearest neighbor queries in $O(\log n)$ time among other related queries.

A grappa tree, as introduced by Aronov et al. [2], is a data structure based on the worst-case version of the link-cut tree construction of Sleator and Tarjan [14]. This structure maintains a forest of fixed-topology rooted binary trees subject to many operations, including Make-Tree, Link, Cut and Evert (which changes the root of a tree) each in $O(\log n)$ worst-case time while using $O(n)$ space.

As in [2, 14], we decompose each rooted binary tree into a set of maximal vertex-disjoint downward paths, called *heavy paths*, connected by tree edges called *light edges*. Each heavy path is in turn represented by a biased binary tree whose leaf-nodes correspond to the vertices of the heavy path. Non-leaf nodes represent edges of this heavy path, ordered in the biased tree according to their depth along the path. Therefore, vertices that are higher (closer to the root) in the path correspond to leaves farther left in the biased tree. Each leaf node ℓ of a biased tree B represents an internal vertex v of the tree which has a unique light edge l_v adjacent to it. We keep a pointer from ℓ to this light edge. Note that the other endpoint of l_v is the root of another heavy path which in turn is represented by another biased tree, say B'. We merge these two biased trees by adding a pointer from ℓ to the root of B'. After merging all the biased trees in this way, we obtain the grappa tree of T. A node of the grappa tree that is non-leaf in its biased tree represents a heavy edge and has two children, whereas a node that is a leaf of its biased tree represents a vertex of the heavy path (and its unique adjacent light edge) and has only one child. By a suitable choice of paths and biasing, as described in [14], the grappa-tree has height $O(\log n)$.

In addition, grappa trees allow us to store left and right marks on each of its nodes, i.e., on each edge of T. To assign the mark of a node, grappa trees support the $O(\log n)$-time operation LEFT-MARK(T, v, m_l) which sets the mark m_l to every edge in the path from v to the root of T (RIGHT-MARK(T, v, m_l) is defined analogously). In our setting, we use the marks of an edge e to keep track of the faces adjacent to this edge in a geometric embedding of T. Since T is rooted, we can differentiate between the left and the right faces adjacent to e. For a full description of the operations supported by grappa trees refer to [2].

▶ **Theorem 12** (Theorem 7 from [2]). *A grappa-tree maintains the combinatorial structure of a forest and supports each operation in $O(\log n)$ worst-case time per operation, where n is the total size of the trees affected by the operation.*

The Voronoi diagram. Let S be a set of n sites in convex position and let $\mathcal{V}(S)$ be the binary tree representing the Voronoi diagram of S. We store $\mathcal{V}(S)$ using a grappa tree. In addition, we assume that each edge of $\mathcal{V}(S)$ has two *face-markers*, its left and right markers which respectively store the site of S whose Voronoi region is adjacent to this edge. While a grappa tree stores only the topological structure of $\mathcal{V}(S)$, with the aid of the face-markers we can retrieve the geometric representation of $\mathcal{V}(S)$. Namely, for each vertex v of $\mathcal{V}(S)$, we can look at its adjacent edges and their face-markers to retrieve the point in the plane representing the location of v in the Voronoi diagram of S in $O(1)$ time. Therefore, we refer to v also as a point in the plane. Recall that each vertex v of $\mathcal{V}(S)$ is the center of a circle that passes through at least three sites of S, we call these sites the *definers* of v and we call this circle the *definer circle* of v.

▶ **Observation 13.** *Given a new site q in the plane such that $S' = S \cup \{q\}$ is in convex position, the vertices of $\mathcal{V}(S)$ that are closer to q than to any other point of S' are exactly the vertices whose definer circle encloses q.*

Let q be a new site such that $S' = S \cup \{q\}$ is in convex position. Let CELL(q, S') be the Voronoi region of q in the Voronoi diagram of S' and let ∂CELL(q, S') denote its boundary. Recall that we can think of $\mathcal{V}(S)$ as a Halin graph by connecting all its leaves by a cycle to make it 3-regular. While we do not explicitly use this cycle, we need it to make our definitions consistent. In this Halin graph, the curve ∂CELL(q, S') can be made into a closed curve by going around the leaf of $\mathcal{V}(S)$ contained in CELL(q, S'); namely the point at infinity of the bisector between the two neighbors of q along the convex hull of S'. In this way,

$\partial\text{CELL}(q, S')$ becomes a flarbable curve. Therefore, we are interested in performing the flarb operation it induces which leads into a transformation of $\mathcal{V}(S)$ into $\mathcal{V}(S')$.

Heavy paths in Voronoi diagrams. Recall that for the grappa tree of $\mathcal{V}(S)$, we computed a heavy path decomposition of $\mathcal{V}(S)$. In this section, we first identify the portion of each of these heavy paths that lies inside $\text{CELL}(q, S')$. Once this is done, we test if any edge adjacent to an endpoint of these paths is preserved. Then within each heavy path, we use the biased trees built on it to further find whether there are non-preserved edges inside this heavy path. After identifying all the non-preserved edges, we remove them which results in a split of $\mathcal{V}(S)$ into a forest where each edge inside $\text{CELL}(q, S')$ is preserved. Finally, we show how to link back the disjoint components into the tree resulting from the flarb operation.

▶ **Observation 14.** *Given a 3-regular graph G and a flarbable curve $\mathcal{C}$, if we can test if a point is enclosed by $\mathcal{C}$ in $O(1)$ time, then we can test if an edge is preserved in $O(1)$ time.*

As a first step, we find the heavy paths of $\mathcal{V}(S)$ whose roots lie in $\text{CELL}(q, S')$. Additionally, we find the portion of each of these heavy paths that lies inside $\text{CELL}(q, S')$.

Recall that there is a leaf ρ of $\mathcal{V}(S)$ that lies in $\text{CELL}(q, S')$ being the point at infinity of the bisector between the two neighbors of q along the convex hull of S'. As a first step, we root $\mathcal{V}(S)$ at ρ by calling Evert(ρ). In this way, ρ becomes the root of $\mathcal{V}(S)$ and all the heavy paths have a *root* which is their endpoint closest to ρ.

Let R be the set the of roots of all heavy paths of $\mathcal{V}(S)$, and let $R_q = \{r \in R : r \in \text{CELL}(q, S')\}$. We focus now on computing the set R_q. By Observation 13, each root in R_q has a definer circle that contains q. We use a dynamic data structure that stores the definer circles of the roots in R and returns those circles containing a given query point in poly-logarithmic time [6]. Using this data structure, we can find a root of R_q whose definer contains q, remove it and repeat the process until finding all the roots in R_q. We obtain the following result.

▶ **Lemma 15.** *We can compute each root in R_q in total $O(|R_q| \log^6 n)$ amortized time.*

Given a root $r \in R$, let h_r be the heavy path whose root is r. Because the portion of $\mathcal{V}(S)$ that lies inside $\text{CELL}(q, S')$ is a connected subtree, we know that, for each $r \in R_q$, the portion of the path h_r contained in $\text{CELL}(q, S')$ is also connected. In order to compute this connected subpath, we want to find the last vertex of h_r that lies inside of $\text{CELL}(q, S')$, or equivalently, the unique edge of h_r having exactly one endpoint in the interior of $\text{CELL}(q, S')$. We call such an edge the *q-transition edge* of h_r (or simply transition edge). Using the search properties of the biased tree representing the heavy path h_r, we obtain the following result.

▶ **Lemma 16.** *For a root $r \in R_q$, we can compute the transition edge of h_r in $O(\log n)$ time.*

Finding non-preserved edges. Let $\mathcal{V}_q(S)$ be the subtree induced by all the edges of $\mathcal{V}(S)$ that intersect $\text{CELL}(q, S')$. Now, we work towards showing how to identify each non-preserved edge of $\mathcal{V}_q(S)$ in the fleeq induced by $\partial\text{CELL}(q, S')$. For each root $r \in R_q$, we compute the transition edge e_r of h_r using Lemma 16 in $O(\log n)$ time per edge. Assume that w is the vertex of e_r that is closer to r (or is equal to r). We consider each edge adjacent to w and test whether or not it is preserved. Since each vertex of $\mathcal{V}_q(S)$ has access to its definers via the faces markers of its adjacent edges, we can test if this vertex lies in $\text{CELL}(q, S')$. Thus, by Observation 14, we can decide whether an edge of $\mathcal{V}_q(S)$ is preserved in $O(1)$ time.

We mark each non-preserved edge among them as *shadow*. Because we can test if an edge is preserved in $O(1)$ time, and since computing e_r takes $O(\log n)$ time by Lemma 16, this

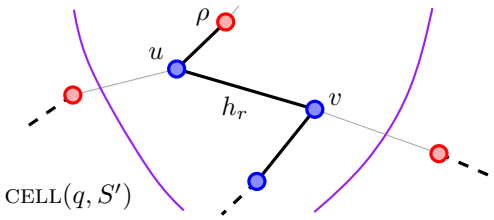


Figure 4 Path h_r contains two adjacent vertices u and v such that the light edge of u is a left edge while the light edge of v is a right edge. The edge uv cannot be preserved.

can be done in total $O(|R_q| \log n)$ time amortized. In addition, notice that if h_r contains two adjacent vertices u and v such that the light edge of u is a left edge while the light edge of v is a right edge (or vice versa), then the edge uv cannot be preserved; see Figure 4. In this case, we say that uv is a *bent edge*. We want to mark all the bent edges in $\mathcal{V}_q(S)$ as shadow, but first we need to identify them efficiently.

Note that it suffices to find all the bent edges of h_r for a given root $r \in R_q$, and then repeat this process for each root in R_q. To find the bent edges in h_r, we further extend the grappa-tree in such a way that the biased tree representing h_r allows us to search for bent edges in $O(\log n)$ time. This extension is described as follows. Recall that each leaf s_v of a biased tree corresponds to a vertex v of the heavy path and has a pointer to the unique light edge adjacent to v. Since each light edge is either left or right, we can extend the biased tree to allow us to search in $O(\log n)$ time for the first two consecutive leaves where a change in direction occurs. From there, standard techniques allow us to find the next change in direction in additional $O(\log n)$ time. Therefore, we can find all the bent edges of a heavy path h_r in $O(\log n)$ time per bent edge. After finding each bent edge in h_r, we mark it is as a shadow edge.

▶ **Lemma 17.** *An edge of $\mathcal{V}_q(S)$ is a preserved edge if and only if it was not marked as a shadow edge.*

Let σ be the number of shadow edges of $\mathcal{V}(S)$, which is equal to the number of non-preserved edges by Lemma 17. The following relates the size of R_q with the value of σ and the cost of the flarb.

▶ **Lemma 18.** *It holds that $|R_q| = O(\sigma \log n)$. Moreover, $\sigma = \Theta(\text{COST}(\mathcal{V}(S), \partial\text{CELL}(q, S')))$.*

The compressed tree. Let $\mathcal{F}$ be the forest obtained from $\mathcal{V}_q(S)$ by removing all the shadow edges (this is just for analysis purposes, so far no cut has been performed). Note that each connected components of $\mathcal{F}$ consists only of preserved edges that intersect $\text{CELL}(q, S')$. Thus, each component inside $\text{CELL}(q, S')$ is a comb, with a path as *spine* and each child of a spine vertex pointing to the same side; see Figure 5. Thus, we have right and left combs, depending on whether the children of the spine are left or right children.

Our objective in the long term is to cut all the shadow edges and link the remaining components in the appropriate order to complete the flarb. To this end, we would like to perform an Eulerian tour on the subtree $\mathcal{V}_q(S)$ to find the order in which the subtrees of $\mathcal{V}(S) \setminus \mathcal{V}_q(S)$ that hang from the leaves of $\mathcal{V}_q(S)$ appear along this tour. However, this may be too expensive as we want to perform this in time proportional to the number of shadow edges, and the size of $\mathcal{V}_q(S)$ can be way larger. To make this process efficient, we compress $\mathcal{V}_q(S)$ by contracting each comb of $\mathcal{F}$ into a single super node. By performing an Eulerian

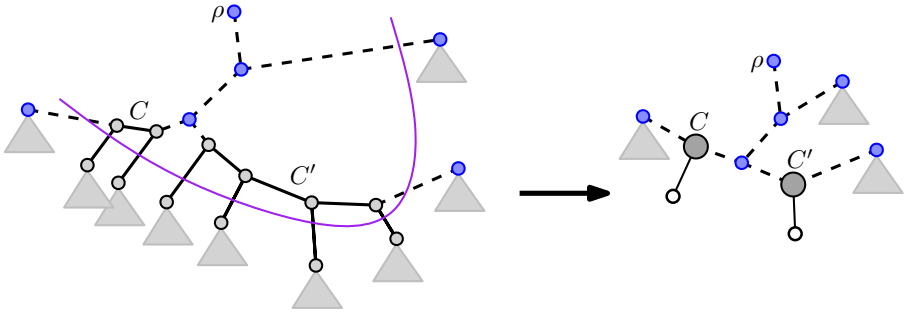


Figure 5 Two combs of $\mathcal{F}$ that are compressed into super nodes with their respective dummy leaves. An Eulerian tour around the compressed tree provides us with the order in which the trees hanging outside of $\textsc{cell}(q, S')$ should be attached.

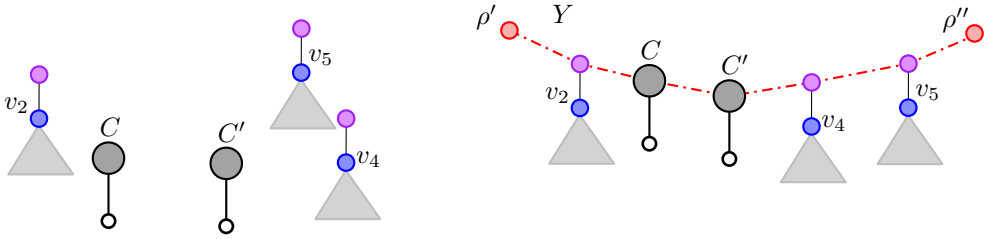


Figure 6 Left: An anchor node is created for each isolated leaf of $\mathcal{V}_q(S)$ and attached as its parent. Other isolated nodes are ignored. Right: A super comb is created connecting two new leaves ρ' and ρ'' through a path. This path connects anchor and super nodes in the order retrieved by the Eulerian tour around the compressed tree.

tour around this compressed tree, we obtain the order in which each component needs to be attached. We construct the compressed flarb and then we decompress it as follows.

Note that each comb has exactly two shadow edges that connect it with the rest of the tree. Thus, we contract the entire component containing the comb into a single super node and add a left or right dummy child to it depending on whether this comb was left or right, respectively; see Figure 5. After the compression, the shadow edges together with the super nodes and the dummy vertices form a tree, called the *compressed tree*, that has $O(\sigma)$ vertices and edges, where σ is the total number of shadow edges.

▶ **Lemma 19.** *We can compute the compressed tree in $O(\sigma \log \sigma)$ time.*

The compressed tree is then a binary tree where each super node has degree three and each edge is a shadow edge. We now perform an Eulerian tour around this compressed tree and retrieve the order in which the leaves of this tree are visited. Some leaves are dummy leaves and some of them are original leaves of $\mathcal{V}_q(S)$; see Figure 5.

Completing the flarb. We now proceed to remove each of the shadow edges which results in a (compressed) forest with $O(\sigma)$ components. Note that each of the original leaves of $\mathcal{V}_q(S)$ was connected with its parent via a shadow edge and hence it lies now as a single component in the resulting forest. For each of these original leaves of $\mathcal{V}_q(S)$, we create a new *anchor* node and link it as the parent of this leaf. Moreover, there could be internal vertices that become isolated. In particular this will be the case of the root ρ. These vertices are deleted and ignored for the rest of the process.

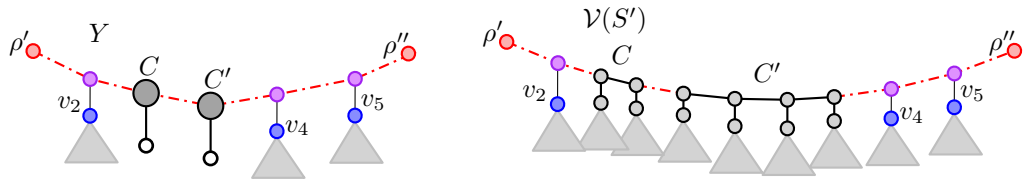


Figure 7 The tree $\mathcal{V}(S')$ achieved after the decompression.

To complete the flarb, we create two new nodes ρ' and ρ'' which will be the two new leaves of the Voronoi diagram, one of them replacing ρ. Then, we construct a path with endpoints ρ and ρ' that connects the super nodes and the anchor nodes according to the traversal order of their leaves; see Figure 6. The resulting tree is a *super comb* Y, where each vertex on the spine is either a super node or an anchor node, and all the leaves are either dummy leaves or original leaves of $\mathcal{V}_q(S)$. Since we glued $O(\sigma)$ components into a tree, we need $O(\sigma)$ time.

We proceed now to decompress Y. To decompress a super node of Y that corresponds to a comb, we consider the two neighbors of the super node in Y and attach each of them to the ends of the spine of the comb. For an anchor node, we simply note that there is a component of $\mathcal{V}(S)$ hanging from its leaf; see Figure 7. In this way, we obtain all the edges that need linking. After the decompression, we end with the tree $\mathcal{V}(S')$ resulting from the flarb. Thus, the flarb operation of inserting q can be implemented with $O(\sigma)$ link and cuts.

Recall that any optimal algorithm needs to perform a cut for each edge that is not preserved. Since each non-preserved edge is shadow by Lemma 17, the optimal algorithm needs to perform at least $\Omega(\sigma)$ operations. Therefore, our algorithm is optimal and computes the flarb using $\Theta(\sigma)$ link and cuts. Moreover, by Lemmas 15 and 16 we can compute the flarb in $O(|R_q| \log^6 n + \sigma \log n)$ amortized time using $\Theta(\sigma)$ link and cuts. Since $|R_q| = O(\sigma \log n)$ and $\sigma = \Theta(\text{COST}(\mathcal{V}(S), \partial\text{CELL}(q, S')))$ by Lemma 18, we obtain the following.

▶ **Theorem 20.** *The flarb operation corresponding to the insertion of q can be implemented with $O(K)$ link and cuts, where K is the cost of the flarb. Moreover, it can be implemented in $O(K \log^7 n)$ amortized time.*

References

1 Greg Aloupis, Luis Barba, and Stefan Langerman. Circle separability queries in logarithmic time. In *Proceedings of the 24th Canadian Conference on Computational Geometry*, CCCG'12, pages 121–125, August 2012.

2 Boris Aronov, Prosenjit Bose, Erik D Demaine, Joachim Gudmundsson, John Iacono, Stefan Langerman, and Michiel Smid. Data structures for halfplane proximity queries and incremental Voronoi diagrams. In *LATIN 2006: Theoretical Informatics*, pages 80–92. Springer, 2006.

3 Luis Barba. Disk constrained 1-center queries. In *Proceedings of the 24th Canadian Conference on Computational Geometry*, CCCG'12, pages 15–19, August 2012.

4 Jon Louis Bentley and James B Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

5 P. Bose, S. Langerman, and S. Roy. Smallest enclosing circle centered on a query line segment. In *Proceedings of the 20th Canadian Conference on Computational Geometry (CCCG 2008)*, pages 167–170, 2008.

6 Timothy M Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *Journal of the ACM (JACM)*, 57(3):16, 2010.

7 Yi-Jen Chiang and Roberto Tamassia. Dynamic algorithms in computational geometry. *Proceedings of the IEEE*, 80(9):1412–1434, 1992.

8 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

9 Herbert Edelsbrunner and Raimund Seidel. Voronoi diagrams and arrangements. *Discrete & Computational Geometry*, 1(1):25–44, 1986.

10 Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

11 Rolf Klein, Elmar Langetepe, and Zahra Nilforoushan. Abstract voronoi diagrams revisited. *Computational Geometry*, 42(9):885–902, 2009.

12 Mark H Overmars. *The design of dynamic data structures*, volume 156. Springer Science & Business Media, 1983.

13 Seth Pettie. Applications of forbidden 0–1 matrices to search tree and path compression-based data structures. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1457–1467, 2010.

14 Daniel D Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.

Dimension Reduction Techniques for ℓ_p $(1 \leq p \leq 2)$, with Applications

Yair Bartal[1] **and Lee-Ad Gottlieb**[2]

1 **Hebrew University, Jerusalem, Israel**
yair@cs.huji.ac.il
2 **Ariel University, Ariel, Israel**
leead@ariel.ac.il

Abstract

For Euclidean space (ℓ_2), there exists the powerful dimension reduction transform of Johnson and Lindenstrauss [26], with a host of known applications. Here, we consider the problem of dimension reduction for all ℓ_p spaces $1 \leq p \leq 2$. Although strong lower bounds are known for dimension reduction in ℓ_1, Ostrovsky and Rabani [40] successfully circumvented these by presenting an ℓ_1 embedding that maintains fidelity in only a bounded distance range, with applications to clustering and nearest neighbor search. However, their embedding techniques are specific to ℓ_1 and do not naturally extend to other norms.

In this paper, we apply a range of advanced techniques and produce bounded range dimension reduction embeddings for all of $1 \leq p \leq 2$, thereby demonstrating that the approach initiated by Ostrovsky and Rabani for ℓ_1 can be extended to a much more general framework. We also obtain improved bounds in terms of the intrinsic dimensionality. As a result we achieve improved bounds for proximity problems including snowflake embeddings and clustering.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Dimension reduction, embeddings

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.16

1 Introduction

Dimension reduction for normed space is a fundamental tool for algorithms and related fields. A much celebrated result for dimension reduction is the well-known l_2 flattening lemma of Johnson and Lindenstrauss [26]: For every n-point subset of l_2 and every $0 < \varepsilon < 1$, there is a mapping into l_2^k that preserves all interpoint distances in the set within factor $1 + \varepsilon$, with target dimension $k = O(\varepsilon^{-2} \log n)$. The dimension reducing guarantee of the Johnson-Lindenstrauss (JL) transform is remarkably strong, and has the potential to make algorithms with a steep dependence on the dimension tractable. It can be implemented as a simple linear transform, and has proven to be a very popular tool in practice, even spawning a stream of literature devoted to its analysis, implementation and extensions (see for example [5, 4, 2]).

Given the utility and impact of the JL transform, it is natural to ask whether these strong dimension reduction guarantees may hold for other ℓ_p spaces as well (see [23] for further motivation). This is a fundamental open problem in embeddings, and has attracted significant research. Yet the dimension reduction bounds known for ℓ_p norms other than ℓ_2 are much weaker than those given by the JL transform, and it is in fact known that a linear dimension reduction mapping in the style the JL transform is quite unique to the ℓ_2-norm [27]. Further, strong lower bounds on dimension reduction are known for ℓ_1 [7] and for ℓ_∞


© Yair Bartal and Lee-Ad Gottlieb;
licensed under Creative Commons License CC-BY

[34], and it is a plausible conjecture that ℓ_2 is the only ℓ_p space which admits the strong distortion and dimension properties of the JL transform.

Ostrovsky and Rabani [30, 39, 40] successfully circumvented the negative results for ℓ_1, and presented a dimension reduction type embedding for the Hamming cube – and by extension, all of l_1 – which preserves fidelity only in a bounded range of distances. They further demonstrated that their embedding finds use in algorithms for nearest neighbor search (NNS) and clustering, as these can be reduced to subproblems where all relevant distance are found in a bounded range. In fact, a number of other proximity problems can also be reduced to bounded range subproblems, including the construction of distance oracles and labels [22, 10], snowflake embeddings [19, 11], and ℓ_p-difference of a pair of data streams. Hence, we view a bounded range embedding as a framework for the solution of multiple important problems. Note also that for spaces with fixed aspect ratio (a fixed ratio between the largest and smallest distances in the set – a common scenario in the literature, see [25]), a bounded range mapping is in effect a complete dimension reduction embedding.

The dimension reduction embedding of Ostrovsky and Rabani is very specific to ℓ_1, and does not naturally extend to other norms. The central contribution of the paper is to bring advanced techniques to bear on this problem, thereby extending this framework to all $1 \leq p \leq 2$.

1.1 Our contribution

We first present a basic embedding in Section 3, which shows that we can reduce dimension while realizing a certain distance transform with low distortion. Using this result, we are able to derive two separate dimension-reducing embeddings:

- Range embedding. In Theorem 4, we present an embedding which preserves distances in a given range with $(1+\varepsilon)$ distortion. The target dimension is $O(\log n)$, with dependence on the range parameter and ε. This generalizes the approach of Ostrovsky and Rabani [39, 40] to all $1 \leq p \leq 2$. This embedding can be applied in the streaming setting as well, and it can also be modified to achieve target dimension polynomial in the *doubling dimension* of the set (Lemma 5).[1]
- Snowflake embedding. An α-snowflake embedding is one in which each inter-point distance t in the origin space is replaced by distance t^α in the embedded space (for $0 < \alpha < 1$). It was observed in [19, 11] that the snowflake of a finite metric space in l_2 may be embedded in dimension which is close to the intrinsic dimension of the space (measured by its doubling dimension), and this may be independent of n. In [19] the case of l_1 was considered as well, however the resulting dimension had doubly exponential dependence on the doubling dimension. We demonstrate that the basic embedding can be used to build a snowflake for ℓ_p for all $1 \leq p \leq 2$ with dimension polynomial in the doubling dimension; this is found in Lemma 7. For ℓ_1 this provides a *doubly exponential* improvement over

[1] Our range embedding has the additional property that it can be used to embed ℓ_p^m into $\ell_q^{O(\log n)}$ (that is, m-dimensional ℓ_p into $O(\log n)$-dimensional ℓ_q, for $1 \leq q < p$) with $(1+\varepsilon)$-distortion in time $O(m \log n)$, with dependence on the range parameter and ε. This is a fast version of the embedding of [28], a common tool for embedding ℓ_p into more malleable spaces such as l_1. (The embedding of [28] is particularly useful for nearest neighbor search, see [30, 21].) Note that [28] features a large overhead cost $O(m^2)$, and since m can be as large as $\Theta(n)$, this overhead can be the most expensive step in algorithms for ℓ_p. We also note that the embedding of Theorem 4 can be used to produce efficient algorithms for approximate nearest neighbor search and ℓ_p difference, although other efficient techniques have already been developed for these specific problems (see for example [8, 37] for NNS, and [24, 32] for ℓ_p difference).

the previously known dimension bound [19], while generalizing the $p \in \{1, 2\}$ results of [19, 11] to all $1 \leq p \leq 2$.

To highlight the utility of our embeddings, we demonstrate (in Section 5) their applicability in deriving better runtime bounds for clustering problems:

We first consider the k-center problem, and show that our snowflake embedding can be used to provide an efficient algorithm. For ℓ_p-spaces of low doubling dimension and fixed p, $1 \leq p \leq 2$, we apply our snowflake embedding in conjunction with the clustering algorithm of Agarwal and Procopiuc [3], and obtain a $(1+\varepsilon)$-approximation to the k-center problem in time $O(n(2^{\tilde{O}(\mathrm{ddim}(S))} + \log k)) + (k \cdot \varepsilon^{-\mathrm{ddim}(S)/\varepsilon^2})k^{1-(\varepsilon/\mathrm{ddim}(S))^{O(1)}}$.

We then consider the min-sum clustering problem, and show that our snowflake embedding can be used to provide an efficient algorithm for this problem. For ℓ_p-spaces of low doubling dimension ($1 \leq p \leq 2$), we apply our snowflake embedding in conjunction with the clustering algorithm of Schulman [44], and obtain a $(1+\varepsilon)$-approximation to the min-sum clustering problem in randomized time $n^{O(1)} + 2^{2^{(O(d'))}}(\varepsilon \log n)^{O(d')}$ where $d' = (\mathrm{ddim}/\varepsilon)^{O(1)}$.

1.2 Related work

For results on dimension reduction for ℓ_p spaces, see [43] for $p < 2$, and also [45] for $p = 1$, [46] for $1 < p < 2$, and [34] for $p = \infty$. Other related notions of dimension reduction have been suggested in the literature. Indyk [24] devised an analogue to the JL-Lemma which uses p-stable distributions to produce estimates of interpoint distances; strictly speaking, this is not an embedding into ℓ_p (e.g. it uses median over the coordinates). Motivated by the nearest neighbor search problem, Indyk and Naor [25] proposed a weaker form of dimension reduction, and showed that every doubling subset $S \subset \ell_2$ admits this type of dimension reduction into ℓ_2 of dimension $O(\mathrm{ddim}(S))$. Roughly speaking, this notion is weaker in that distances in the target space are allowed to err in one direction (be too large) for all but one pair of points. Similarly, dimension reduction into ordinal embeddings (where only relative distance is approximately preserved) was considered in [6, 14]. Bartal, Recht and Schulman [11] developed a variant of the JL-Lemma that is local – it preserves the distance between every point and the $\hat{k}$ points closest to it. Assuming $S \subset \ell_2$ satisfies a certain growth rate condition, they achieve, for any desired $\hat{k}$ and $\varepsilon > 0$, an embedding of this type with distortion $1+\varepsilon$ and dimension $O(\varepsilon^{-2} \log \hat{k})$. The same paper built on a result of [19] to show that for ℓ_2 a $(1+\varepsilon)$-approximate α-snowflake embeds into $\tilde{O}(\varepsilon^{-3}\,\mathrm{ddim}(S))$ dimensions (for fixed α). For ℓ_1, [19] showed that a $(1+\varepsilon)$-approximate α-snowflake embeds into $2^{\varepsilon^{-\tilde{O}(\mathrm{ddim}(S))}}$ dimensions.

2 Preliminaries

2.1 Embeddings and metric transforms

Following [13], we define an *oblivious* embedding to be an embedding which can be computed for any point of a database set X or query set Y, without knowledge of any other point in X or Y. (This differs slightly from the definition put forth by Indyk and Naor [25].) Familiar oblivious embeddings include standard implementations of the JL-Lemma for l_2, the dimension reduction mapping of Ostrovsky and Rabani [40] for the Hamming cube, and the embedding of Johnson and Schechtman [28] for ℓ_p, $p \leq 2$. A *transform* is a function mapping from the positive reals to the positive reals, and a *metric transform* maps a metric distance function to another metric distance function on the same set of points. An embedding is

transform preserving with respect to a transform if it achieves the distances defined by that transform.

Let $(X, d_X), (Y, d_Y)$ be metric spaces. For distance scales $0 \leq a \leq b \leq \infty$, an $[a, b]$-embedding of X into Y with distortion D is a mapping $f : X \to Y$ such that for all $x, y \in X$ such that $d_X(x, y) \in [a, b]$: $1 \leq c \cdot \frac{d_Y(f(x), f(y))}{d_X(x,y)} \leq D$. (Here, c is any scaling constant.) Then f is a *range embedding* with range $[a, b]$. If f has the additional property that for all x, y such that $d(x, y) < a$ we have $c \cdot \frac{d_Y(f(x), f(y))}{a} \leq D$, then we say that f is range preserving from below. Similarly, if for all x, y such that $d(x, y) > b$ we have $c \cdot \frac{d_Y(f(x), f(y))}{b} \geq 1$, then we say that f is range preserving from above. And if f is range preserving from above and below, then we say that f is a *strong* range embedding.

Let $R > 1$ be a parameter. We say that X admits an R-range embedding into Y with distortion D if for every $u > 0$ there exists an $[u, uR]$ embedding of X into Y with distortion D. As above, an R-range embedding may be range preserving from above or below. We will usually take $u = 1$.

2.2 Doubling dimension, nets and hierarchies

For a metric $(\mathcal{X}, \rho)$, let $\lambda > 0$ be the smallest value such that every ball in $\mathcal{X}$ can be covered by λ balls of half the radius. The *doubling dimension* of $\mathcal{X}$ is $\mathrm{ddim}(\mathcal{X}) = \log_2 \lambda$. Note that $\mathrm{ddim}(\mathcal{X}) \leq \log n$. The following packing property can be shown (see, for example [29]): Suppose that $S \subset \mathcal{X}$ has a minimum interpoint distance of at least α. Then $|S| \leq \left(\frac{2\,\mathrm{diam}(S)}{\alpha}\right)^{\mathrm{ddim}(\mathcal{X})}$.

Given a metric space S, $S' \subset S$ is a γ-net of S if the minimum interpoint distance in S' is at least γ, while the distance from every point of S to its nearest neighbor in S' is less than γ. Let S have minimum inter-point distance 1. A *hierarchy* is a series of $\lceil \log \Delta \rceil$ nets (Δ being the aspect ratio of S), where each net S_i is a 2^i-net of the previous net S_{i-1}. The first (or bottom) net is $S_0 = S$, and the last (or top) net S_t contains a single point called the *root*. For two points $u \in S_i$ and $v \in S_{i-1}$, if $d(u, v) < 2^i$ then we say that u *covers* v, and this definition allows v to have multiple covering points in S_i. The closest covering point of v is its *parent*. The distance from a point in S_i to its ancestor in S_j is at most $\sum_{k=i+1}^{j} 2^k = 2 \cdot (2^j - 2^{i+1}) < 2 \cdot 2^j$.

Given S, a hierarchy for S can be built in time $2^{O(\mathrm{ddim}(S))} n$, and this term bounds the size of the hierarchy as well [22, 16]. The height of the hierarchy is $O(\min\{n, \log \Delta\})$.

2.3 Probabilistic partitions

Probabilistic partitions are a common tool used in embeddings. Let (X, d) be a finite metric space. A partition P of X is a collection of non-empty pairwise disjoint clusters $P = \{C_1, C_2, \ldots, C_t\}$ such that $X = \cup_j C_j$. For $x \in X$ we denote by $P(x)$ the cluster containing x. We will need the following decomposition lemma due to Gupta, Krauthgamer and Lee [20], Abraham, Bartal and Neiman [1], and Chan, Gupta and Talwar [15].[2] Let ball $B(x, r) = \{y | \, \|x - y\| \leq r\}$.

▶ **Theorem 1** (Padded Decomposition of doubling metrics [20, 1, 15]). *There exists a constant $c_0 > 1$, such that for every metric space (X, d), every $\varepsilon \in (0, 1)$, and every $\delta > 0$, there is a*

[2] [20] provided slightly different quantitative bounds than in Theorem 1. The two enumerated properties follow, for example, from Lemma 2.7 in [1], and the bound on support-size m follows by an application of the Lovász Local Lemma sketched therein.

multi-set $\mathcal{D} = [P_1, \ldots, P_m]$ *of partitions of* X*, with* $m \leq c_0 \varepsilon^{-1} \dim(X) \log \dim(X)$*, such that* (i) *Bounded radius:* $\operatorname{diam}(C) \leq \delta$ *for all clusters* $C \in \bigcup_{i=1}^{m} P_i$*; and* (ii) *Ball padding: If* P *is chosen uniformly from* $\mathcal{D}$*, then for all* $x \in X$*,* $\Pr_{P \in \mathcal{D}}[B(x, \frac{\delta}{c_0 \dim(X)}) \subseteq P(x)] \geq 1 - \varepsilon$.

2.4 Stable distributions

The density of symmetric p-stable random variables ($0 < p \leq 2$) is $h(x) = \frac{1}{\pi} \int_0^\infty \cos(tx) e^{-t^p} dt$ [47, Section 2.2]. The density function $h(x)$ is unimodal and bell-shaped. It is well known that $\frac{c_p}{1+x^{p+1}} \leq h(x) \leq \frac{c'_p}{1+x^{p+1}}$ for constants c_p, c'_p that depend only on p [38] (and we may ignore the dependence on p for the purposes of this paper). Using this approximation for $h(x)$, it is easy to see that for $0 < q < p$ and p-stable random variable g, $\mathbb{E}[g^q] = \int_0^\infty x^q h(x) dx \approx \int_0^1 x^q dx + \int_1^\infty x^{q-(p+1)} dx \approx \frac{1}{p-q}$. (Here we use the notation $\approx$ in the same sense as $\Theta(\cdot)$.) Also, for $b > 1$, $\int_0^b x^p h(x) dx \approx \int_0^1 x^p dx + \int_1^b \frac{1}{x} dx \approx 1 + \ln b$. Let $g_1, g_2, \ldots, g_m \in G$ be a set of i.i.d. symmetric p-stable random variables, and let v be a real m-length vector. A central property of p-stable random variables is that $\sum_{j=1}^m g_j v_j$ is distributed as $g(\sum_{j=1}^m v_j^p)^{1/p} = g\|v\|_p$, for all $g \in G$. When all $g_i \in G$ are normalized as $\mathbb{E}[|g_i|^q] = 1$, we have that $\mathbb{E}\left[\left|\sum_{j=1}^m g_j v_j\right|^q\right] = \mathbb{E}[|g|^q]\|v\|_p^q = \|v\|_p^q$.

3 Basic transform-preserving embedding

In Theorem 3 below, we present an embedding which realizes a certain distance transform with low distortion, while also reducing dimension to $O(\log n)$ (with dependence on the range and the desired distortion). We then demonstrate that the transform itself has several desirable properties. In the next section, we will use this transform-preserving embedding to obtain a range embedding and a snowflake embedding.

3.1 Embedding

We first present a randomized embedding into a single coordinate, and show that it is transform-preserving in expectation only (Lemma 2). We then show that a concatenation of many single-coordinate embeddings yields a single embedding which (approximately) preserves the bounded transform with high probability, and this gives Theorem 3.

The following single-coordinate embedding is inspired by the Nash device of Bartal *et al.* [11] (see also [42]), and is related to the spherical threshold function of Mendel and Naor [35, Lemma 5.9]. Broadly speaking, our embedding uses the sine function as a dampening tool, which serves to mitigate undesirable properties of p-stable distributions (i.e., their heavy tails). Our central contribution is to give a tight analysis of the transform preserved by our embedding.[3]

Let $S \subset \ell_p$ be a set of m-dimensional vectors for $1 \leq p \leq 2$. Let $g_j \in G$ be k i.i.d. symmetric p-stable random variables, and fix parameters s (the *threshold*) and $0 \leq \phi \leq 2\pi$. Further define the fixed constant $P_q = \mathbb{E}[|\cos\theta|^q] = \frac{1}{2\pi} \int_0^{2\pi} |\cos\theta|^q d\theta$ for $1 \leq q \leq p$ (and note that $\frac{1}{2} = \frac{1}{2\pi} \int_0^{2\pi} \cos^2\theta d\theta \leq P_q \leq \frac{1}{2\pi} \int_0^{2\pi} |\cos\theta| d\theta = \frac{2}{\pi}$). Then the embedding $F_{\phi,s} : S \to L_q^1$

[3] It may be possible to replace our embedding with that of Mendel and Naor [35], but one would still require a tighter analysis, and the final dimension would likely increase. Note also that our embedding is into the reals, while [35] embed into the complex numbers; as ℓ_p^d over $\mathbb{C}$ embeds into $\ell_p^{O(\epsilon^{-2}\sqrt{p}2^{p/2}d)}$ over $\mathbb{R}$ with distortion $1+\epsilon$ (a consequence of Dvoretzky's theorem [36]), the embedding of [35] can in fact be used to achieve an embedding into the reals with increased dimension.

(i.e., into a single coordinate of L_q) for vector $v \in S$ is defined by $F_s(v) = F_{\phi,s}(v) = \frac{s}{2P_q^{1/q}} \sin\left(\phi + \frac{2}{s}\sum_{i=1}^m g_i v_i\right)$. Note that $0 \leq F_s(v) < s$. For vectors $v, w \in S$ we have that $|F_s(v) - F_s(w)|^q = \frac{s^q}{2^q P_q} \left|\sin\left(\phi + \frac{2}{s}\sum_{i=1}^m g_i v_i\right) - \sin\left(\phi + \frac{2}{s}\sum_{i=1}^m g_i w_i\right)\right|^q$ $= \frac{s^q}{P_q}\left|\sin\left(\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)\right)\cos\left(\phi + \frac{1}{s}\sum_{i=1}^m g_i(v_i + w_i)\right)\right|^q$. Now, when ϕ is a random variable chosen uniformly from the range $[0, 2\pi]$ and independently of set G, we have that $\mathbb{E}[|F_s(v) - F_s(w)|^q] = \frac{s^q}{P_q}\mathbb{E}\left[\left|\sin\left(\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)\right)\cos\left(\phi + \frac{1}{s}\sum_{i=1}^m g_i(v_i + w_i)\right)\right|^q\right] = \frac{s^q}{P_q}\mathbb{E}\left[\left|\sin\left(\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)\right)\cos(\phi)\right|^q\right] = \frac{s^q}{P_q}\mathbb{E}\left[\left|\sin\left(\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)\right)\right|^q\right] \cdot \mathbb{E}\left[|\cos(\phi)|^q\right] = s^q\mathbb{E}\left[\left|\sin\left(\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)\right)\right|^q\right]$, where the second equality follows from the periodicity of the cosine function, and the independence of ϕ and G.

This is our single-coordinate embedding. In Lemma 2 below, we will describe its behavior on interpoint distances – that is, we derive useful bounds on $\mathbb{E}[|F_s(v) - F_s(w)|^q$. Now $\frac{1}{s}\sum_{i=1}^m g_i(v_i - w_i)$ is distributed as $g\frac{\|v-w\|_p}{s}$ for random variable $g \in G$, so we will set $a = \frac{\|v-w\|_p}{s}$ and will derive bounds for $H(a) = \mathbb{E}[|\sin(ag)|^q] = s^{-q}\mathbb{E}[|F_s(v) - F_s(w)|^q]$. But first we introduce the full embedding f, which is a scaled concatenation of k single-coordinate embeddings: Independently for each coordinate i, create a function $F_{\phi_i} = F_{\phi_i,s}$ by fixing a random angle $0 \leq \phi_i \leq 2\pi$ and a family of k i.i.d. symmetric p-stables. Then coordinate $f(v)_i$ is defined by $f(v)_i = k^{-1/p}F_{\phi_i,s}(v)$ which can be constructed in $O(m)$ time per coordinate. Note that for $t = \|v - w\|_p$, $\mathbb{E}[|f(v)_i - f(w)_i|^q] = \frac{1}{k}\mathbb{E}[|F_{\phi_i}(v) - F_{\phi_i}(w)|^q] = \frac{s^q}{k}H(t/s)$, and so $\mathbb{E}[\|f(v) - f(w)\|_p^q] = s^q H(t/s)$. Below, we will show that with high probability f is is transform-preserving with respect to H with low distortion.

3.2 Analysis of basic embedding

We now show that both the single-coordinate embedding and the full embedding have desirable properties. Recall that $h(u)$ is the density function of p-stable random variables. Set $Q = 2\int_0^\infty u^q h(u)du \approx \frac{1}{p-q}$. Also, for $a < 1$ and some fixed $a^2 < \varepsilon < 1$, set $Q_a = \frac{1}{2}\int_0^{\sqrt{\varepsilon}/a} u^p h(u)du = \frac{1}{2}\left[\int_0^1 u^p h(u)du + \Theta(\ln(\sqrt{\varepsilon}/a))\right] \approx 1 + \ln(\sqrt{\varepsilon}/a)$.

▶ **Lemma 2.** *Let g be a symmetric p-stable random variable. For $1 \leq q \leq p \leq 2$ and any fixed $0 < \varepsilon < \frac{1}{2}$, $H(a) = \mathbb{E}[|\sin(ag)|^q]$ obeys the following:*

- **(a)** *Threshold: $H(a) \leq 1$.*
- **(b)** *Bi-Lipschitz for small scales: When $q < p$ and $a \leq \min\{\varepsilon^{\frac{1}{2}+\frac{1}{p-q}}, \sqrt{\varepsilon}(1 + (p-q)\varepsilon^{-(\frac{q}{2}+1)})^{\frac{-1}{p-q}}\}$, we have $1 - O(\varepsilon) \leq \frac{H(a)}{a^q Q} \leq 1 + O(\varepsilon)$. When $q = p$ and $a \leq \sqrt{\varepsilon}e^{-\varepsilon^{-(\frac{q}{2}+1)}}$, we have $1 - O(\varepsilon) \leq \frac{H(a)}{a^q Q_a} \leq 1 + O(\varepsilon)$.*
- **(c)** *Bi-Lipschitz for intermediate scales: When $q < p$ and $a < 1$, let $\delta = 1 - a^{p-q}$, and we have $H(a) = \Theta(1 + \delta Q)a^q$. When $q = p$ we have $H(a) = \Theta\left(1 + \ln(1/a)\right)a^q$.*
- **(d)** *Lower bound for large scales: When $a \geq 1$, $H(a) > \frac{1}{8}$.*
- **(e)** *Smoothness: When $a \leq 1$, $\frac{|H((1+\varepsilon)a) - H(a)|}{H(a)} = O(\varepsilon)$.*

It follows that the distance transform implied by our single-coordinate embedding F_s achieves the bounds of Lemma 2 scaled by s^q, if only in expectation. Item 2 implies that for very small a (i.e., when the inter-point distance under consideration is sufficiently small with respect to the parameter s) the embedding has very small expected distortion (at least when $q < p$). Weaker distortion bounds hold for distances smaller than s (item 3). For distances greater than s, these are contracted to the threshold (item 1) or slightly smaller (item 4). The smoothness property will be useful for constructing the snowflake in Section 4.3. We proceed to consider the full embedding:

▶ **Theorem 3.** *Let $1 \leq q \leq p \leq 2$ and $0 < \varepsilon < \frac{1}{2}$, and consider an n-point set $S \in l_p^m$. Set threshold $s > 1$. Then with constant probability the oblivious embedding $f : S \to l_q^k$ for $k = O\left(\frac{\log n}{\varepsilon^2} \cdot \min\left\{s^{2q}, \max\left\{\frac{s^{2q-p}}{2q-p}, \varepsilon s^q\right\}\right\}\right)$, satisfies the following for each point pair $v, w \in S$, where $t = \|v - w\|_p$:*

1. *Threshold: $\|f(v) - f(w)\|_q^q < s^q$.*
2. *Bi-Lipschitz for large scales: When $t \geq 1$, we have $(1-\varepsilon)s^q H(t/s) \leq \|f(v) - f(w)\|_q^q \leq (1+\varepsilon)s^q H(t/s)$.*
3. *Bounded expansion for small scales: When $t < 1$, we have $\|f(v) - f(w)\|_q^q \leq s^q H(1/s) + \varepsilon$.*

The embedding can be constructed in $O(mk)$ time per point.

Theorem 3 demonstrates that in a certain range, there exists a transform-preserving embedding with high probability. (We note that when $2q$ is close to p, better dimension bounds can be obtained by embedding into an interim value $q + c$ for some c and then embedding into q.)

Proof of Lemma 2.

Item 1. This follows trivially from the fact that $|\sin(x)| \leq 1$.

Item 2. Note that since the density function h is symmetric about 0,

$$H(a) = 2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du + 2\int_{\sqrt{\varepsilon}/a}^{\infty} |\sin(au)|^q h(u)du. \tag{1}$$

We show that under the conditions of the item, the second term in Equation (1) is dominated by the first. Considering the second term, we have that $2\int_{\sqrt{\varepsilon}/a}^{\infty} |\sin(au)|^q h(u)du < 2\int_{\sqrt{\varepsilon}/a}^{\infty} h(u)du < 2c'_p \int_{\sqrt{\varepsilon}/a}^{\infty} \frac{1}{u^{p+1}}du = \frac{2c'_p}{p}\left(\frac{a}{\sqrt{\varepsilon}}\right)^p$.

Turning to the first term, recall the Taylor series expansion $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots$; so when $x < \sqrt{\varepsilon}$ we have that $x(1 - \frac{\varepsilon}{6}) < \sin(x) < x$. Also note that the conditions of the item give that $a < \sqrt{\varepsilon}$, and so $\frac{\sqrt{\varepsilon}}{a} > 1$. Hence, when $q < p$ we have $2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du > 2(1-\varepsilon/6)^q a^q \int_0^{\sqrt{\varepsilon}/a} u^q h(u)du > 2(1-\varepsilon/3)a^q c_p \left[\int_0^1 \frac{u^q}{1+u^{p+1}}du + \int_1^{\sqrt{\varepsilon}/a} \frac{u^q}{1+u^{p+1}}du\right] > 2(1-\varepsilon/3)a^q c_p \left[\int_0^1 \frac{u^q}{2}du + \int_1^{\sqrt{\varepsilon}/a} \frac{u^{q-p-1}}{2}du\right] = (1-\varepsilon/3)a^q c_p \left[\frac{1}{q+1} + \frac{1-(a/\sqrt{\varepsilon})^{p-q}}{p-q}\right]$. When $q = p$, the same analysis gives $2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du > (1-\varepsilon/3)a^q c_p \left[\frac{1}{q+1} + \ln(\sqrt{\varepsilon}/a)\right]$

We proceed to show that the first term of Equation (1) exceeds the second by a factor of $\Omega(\varepsilon^{-1})$. For $q = p$ this holds trivially when $a^q \ln(\sqrt{\varepsilon}/a) \geq \frac{1}{\varepsilon}\left(\frac{a}{\sqrt{\varepsilon}}\right)^p$ – or equivalently, when $\ln(\sqrt{\varepsilon}/a) \geq \varepsilon^{-(\frac{q}{2}+1)}$ – which in turn holds exactly when $a \leq \sqrt{\varepsilon}e^{-\varepsilon^{-(\frac{q}{2}+1)}}$. For $q < p$, the condition is fulfilled when $a^q \geq \frac{1}{\varepsilon}\left(\frac{a}{\sqrt{\varepsilon}}\right)^p$, which holds exactly when $a \leq \varepsilon^{\frac{p/2+1}{p-q}}$. Better, the condition is also fulfilled when $a^q \left[\frac{1-(a/\sqrt{\varepsilon})^{p-q}}{p-q}\right] \geq \frac{1}{\varepsilon}\left(\frac{a}{\sqrt{\varepsilon}}\right)^p$, and this is equivalent to satisfying $a^{p-q}\left[\frac{p-q}{\varepsilon^{\frac{p}{2}+1}} + \frac{1}{\varepsilon^{(p-q)/2}}\right] \leq 1$; this holds exactly when $a \leq \sqrt{\varepsilon}(1 + (p-q)\varepsilon^{-(q/2+1)})^{-1/(p-q)}$.

As the first term of Equation (1) dominates the second, it follows that when $q < p$ then for some constant c $H(a) = 2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du + 2\int_{\sqrt{\varepsilon}/a}^{\infty} |\sin(au)|^q h(u)du \leq 2(1 + c\varepsilon)\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du < 2(1+c\varepsilon)a^q \int_0^{\infty} u^q h(u)du = (1+c\varepsilon)a^q Q$. Further, we have that $H(a) = 2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du + 2\int_{\sqrt{\varepsilon}/a}^{\infty} |\sin(au)|^q h(u)du > 2\int_0^{\sqrt{\varepsilon}/a} |\sin(au)|^q h(u)du > 2(1-\varepsilon/3)a^q \int_0^{\sqrt{\varepsilon}/a} u^q h(u)du > (1-\varepsilon/3)(1-c'\varepsilon)a^q Q$, where the final inequality follows

from noting that since $a \leq \varepsilon^{\frac{1}{2}+\frac{1}{p-q}}$, $\int_{\sqrt{\varepsilon}/a}^{\infty} u^q h(u)du \leq \int_{\varepsilon^{-1/(p-q)}}^{\infty} u^q h(u)du \approx \frac{\varepsilon}{p-q} \approx \varepsilon Q$, and so $\int_0^{\sqrt{\varepsilon}/a} u^q h(u)du = \int_0^{\infty} u^q h(u)du - \int_{\sqrt{\varepsilon}/a}^{\infty} u^q h(u)du > (1-c'\varepsilon)Q$ for some c'. This completes the analysis for $q < p$. The same analysis gives that when $q = p$, $H(a) < 2(1+c\varepsilon)a^q \int_0^{\sqrt{\varepsilon}/a} u^q h(u)du = (1+c\varepsilon)a^q Q_a$, and $H(a) > 2(1-\varepsilon/3)a^q \int_0^{\sqrt{\varepsilon}/a} u^q h(u)du = (1-\varepsilon/3)a^q Q_a$.

Item 3. The analysis is similar to that presented in the proof of Item 2. Noting that when $0 \leq x \leq 1$, $\sin(x) \approx x$, and recalling that under the conditions of the item $a \leq 1$, we have for $q < p$ that $H(a) = 2\int_0^{1/a} |\sin(au)|^q h(u)du + 2\int_{1/a}^{\infty} |\sin(au)|^q h(u)du. = \Theta\left(a^q \int_0^{1/a} u^q h(u)du\right) + O\left(\int_{1/a}^{\infty} h(u)du\right) = \Theta\left(\left(1+\frac{1-a^{p-q}}{p-q}\right)a^q\right) + O(a^p) = \Theta\left(\left(1+\frac{\delta}{p-q}\right)a^q\right)$
$+ O(a^p) = \Theta\left(\left(1+\frac{\delta}{p-q}\right)a^q\right)$. Similarly, for $q = p$ we have $H(a) = \Theta\left(a^q \int_0^{1/a} u^p h(u)du\right) + O\left(\int_{1/a}^{\infty} h(u)du\right) = \Theta\left(a^q \int_0^{1/a} u^p h(u)du\right) \approx (1+\ln(1/a))\, a^p$.

Item 4. First note that when $p \geq 1$, $h(x) = \frac{1}{\pi}\int_0^{\infty} \cos(tx)e^{-t^p}dt < \frac{1}{\pi}\int_0^{\infty} e^{-t^p}dt < \frac{1}{\pi}$, so $\int_0^b h(u)du < \frac{b}{\pi}$. Since $h(x)$ is a symmetric density function, we have $\int_0^{\infty} h(u)du = \frac{1}{2}$, and so $\int_b^{\infty} h(u)du > \frac{1}{2} - \frac{b}{\pi}$. We have for any $0 < \theta < \frac{\pi}{2}$,

$$\begin{aligned}
H(a) &\geq 2\int_{\theta/a}^{\infty} |\sin(au)|^q h(u)du \;>\; 2\sum_{i=0}^{\infty} \int_{\frac{i\pi+\theta}{a}}^{\frac{(i+1)\pi-\theta}{a}} |\sin(au)|^q h(u)du \\
&> 2|\sin(\theta)|^q \sum_{i=0}^{\infty} \int_{\frac{i\pi+\theta}{a}}^{\frac{(i+1)\pi-\theta}{a}} h(u)du \;>\; 2|\sin(\theta)|^q \left[1-\frac{2\theta}{\pi}\right] \sum_{i=0}^{\infty} \int_{\frac{i\pi+\theta}{a}}^{\frac{(i+1)\pi+\theta}{a}} h(u)du \\
&= 2|\sin(\theta)|^q \left[1-\frac{2\theta}{\pi}\right] \int_{\theta/a}^{\infty} h(u)du \;>\; |\sin(\theta)|^q \left[1-\frac{2\theta}{\pi}\right]\left[1-\frac{2\theta}{a\pi}\right],
\end{aligned}$$

where the fourth inequality follows from the fact that $h(x)$ is monotone decreasing for $x \geq 0$. The claimed result follows by taking $a = 1$ (the maximum value of a) and $\theta = \frac{\pi}{4}$, and recalling that $q \leq 2$.

Item 5. As in the proof of Item 3 above, we have that for $q \leq p$ and $a \leq 1$, $H(a) \approx \int_0^{1/a} |\sin(au)|^q h(u)du + \int_{1/a}^{\infty} h(u)du = \int_0^{1/a} |\sin(au)|^q h(u)du + O(a^p) \approx \int_0^{1/a} |\sin(au)|^q h(u)du$. Now recall the Taylor series expansion $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots > 1 - \frac{x^2}{2}$ (when $0 \leq x \leq 1$), and note that as a consequence of the Mean Value Theorem, $||A|^q - |B|^q| = qC^{q-1}||A| - |B||$ for some $|A| \geq C \geq |B|$. Further noting that when $u \leq \frac{1}{a}$ we have $au \leq 1$ and that when $u \leq \frac{1}{\varepsilon a}$ we have $\varepsilon au \leq 1$, we conclude that

$$\begin{aligned}
H(a(1+\varepsilon)) - H(a) &= 2\int_0^{\infty} [|\sin(a(1+\varepsilon)u)|^q - |\sin(au)|^q]h(u)du \\
&\leq 2\int_0^{\infty} q[\max\{|\sin(a(1+\varepsilon)u)|, |\sin(au)|\}]^{q-1} \cdot \\
&\qquad [||\sin(a(1+\varepsilon)u)| - |\sin(au)||]h(u)du \\
&\leq 2\int_0^{\infty} q[\max\{|\sin(au)\cos(\varepsilon au)| + |\sin(\varepsilon au)\cos(au)|, |\sin(au)|\}]^{q-1} \cdot \\
&\qquad [||\sin(au)\cos(\varepsilon au)| + |\sin(\varepsilon au)\cos(au)| - |\sin(au)||]h(u)du
\end{aligned}$$

$$
\begin{aligned}
&\leq 2\int_0^\infty q[|\sin(au)|^{q-1}+|\sin(\varepsilon au)|^{q-1}][|\sin(\varepsilon au)|]h(u)du\\
&= O\left(\int_0^{1/a}\varepsilon|\sin(au)|^q h(u)du+\varepsilon a\int_{1/a}^{1/(\varepsilon a)}uh(u)du+\int_{1/(\varepsilon a)}^\infty h(u)du\right)\\
&= O\left(\int_0^{1/a}\varepsilon|\sin(au)|^q h(u)du+\varepsilon a^p+\varepsilon^p a^p\right)\\
&= O(\varepsilon H(a)).
\end{aligned}
$$

◀

Proof of Theorem 3. The first claim of the theorem follows from the fact that for all i, $0 \leq F_{\phi_i,s}(v) < s$. To prove the rest of the theorem, we may make use of Hoeffding's inequality. When $1 \leq t \leq s$, we have by Lemma 2234 that $s^qH(t/s) = s^q\Omega(1) = \Omega(1)$. Then Claim 10 implies that for some $k = O\left(\frac{s^{2q}}{\varepsilon^2}\log n\right)$, $\Pr\left[|\|f(v)-f(w)\|_q^q - s^qH(t/s)| > \varepsilon s^qH(t/s)\right] = \Pr\left[|\|f(v)-f(w)\|_q^q - s^qH(t/s)| > \varepsilon\Omega(1)\right] \leq \frac{1}{n^2}$, so this distortion guarantee can hold simultaneously for all point pairs. When $t < 1$, first note that when $H(t/s) = \Theta(1)$, we can use the same proof as for $1 \leq t \leq s$ above. If $H(t/s) = o(1)$, $\Pr\left[\|f(v)-f(w)\|_q^q > s^qH(1/s)+\varepsilon\right] = \Pr\left[\|f(v)-f(w)\|_q^q - s^qH(t/s) > s^qH(1/s) - s^qH(t/s)+\varepsilon\right]$
$< \Pr\left[\|f(v)-f(w)\|_q^q - s^qH(t/s) > \varepsilon\right] \leq \frac{1}{n^2}$.

An alternate bound can be derived using Bennett's inequality (Claim 11). For this, it suffices to take $k = O\left(\frac{s^{2q}\log n}{\sigma^2 V\left(\frac{s^q\varepsilon[s^qH(t/s)]}{\sigma^2}\right)}\right)$, with the variance term $\sigma^2 = \Theta\left(s^{2q}\mathbb{E}[|\sin(ag)|^{2q}]\right)$. We will prove the case $t \geq 1$, and the case $t < 1$ follows as above. Set $r = \frac{s^q\varepsilon[s^qH(t/s)]}{\sigma^2}$. If $r \geq 1$, then we have $V(r) = \Omega(r)$, and so $k = O\left(\frac{s^q\log n}{\varepsilon[s^qH(t/s)]}\right) = O\left(\frac{s^q\log n}{\varepsilon}\right)$. Otherwise $r < 1$, and we have $V(r) = \Theta(r^2)$, from which we derive $k = O\left(\frac{\sigma^2\log n}{\varepsilon^2[s^qH(t/s)]^2}\right)$. Now, if $t \geq s$, we recall by 24 that $H(t/s) = \Theta(1)$, and noting that $\sigma^2 \approx s^{2q}\mathbb{E}[|\sin(ag)|^{2q}] = O(s^{2q})$ we obtain $k = O\left(\frac{\log n}{\varepsilon^2}\right)$. When $1 \leq t < s$, we have by 223 that $H(t/s) = \Omega((t/s)^q)$, and so $k = O\left(\frac{\sigma^2\log n}{\varepsilon^2 t^{2q}}\right)$. In this case we require a better bound on σ^2: Setting $a = \frac{t}{s} < 1$ we have (by analysis similar to the proof of Lemma 2) $\sigma^2 \approx s^{2q}\mathbb{E}[|\sin(ag)|^{2q}] \approx s^{2q}\left[a^{2q}\int_0^1 u^{2q}du + a^{2q}\int_1^{1/a}u^{2q-p-1}du + \int_{1/a}^\infty u^{-p-1}du\right] \approx s^{2q}\left[\frac{a^{2q}}{2q+1}+\frac{a^p}{2q-p}+\frac{a^p}{p}\right] \approx s^{2q}\frac{a^p}{2q-p}$. It follows that $k = O\left(\frac{a^{p-2q}\log n}{(2q-p)\varepsilon^2}\right) = O\left(\frac{s^{2q-p}\log n}{(2q-p)\varepsilon^2}\right)$. ◀

4 Range and snowflake embeddings

In Section 3 above, we presented our basic embedding. Here, we show how to use the basic embedding to derive a dimension-reducing embedding that preserves distances in a fixed range (Section 4.1). We also show that the basic embedding can be used to derive a dimension-reducing snowflake embedding (Section 4.3). As a precursor to the snowflake embedding, we show that the basic and range embeddings can be improved to embed into the doubling dimension of the space (Section 4.2).

4.1 Range embedding

By combining Theorem 3 and Lemma 2 and choosing an appropriate parameter s, we can achieve a dimension-reducing oblivious strong range embedding for ℓ_p. This is the central contribution of our paper:

▶ **Theorem 4.** *Let $1 \le q \le p \le 2$ and $0 < \varepsilon < \frac{1}{2}$, and consider an n-point set $S \in l_p^m$. Fix range $R > 1$ and set threshold s as follows: When $q < p$, $s \approx R\varepsilon^{-1/2}(1 + (p-q)\varepsilon^{-(q/2+1)})^{1/(p-q)}$, and when $q = p$, $s \approx \max\{R^{1/\varepsilon}, R\varepsilon^{-1/2}e^{\varepsilon^{-(\frac{q}{2}+1)}}\}$. Then there exists an oblivious embedding $f : S \to l_q^k$ for $k = O\left(\frac{\log n}{\varepsilon^2} \cdot \min\left\{s^{2q}, \max\left\{\frac{s^{2q-p}}{2q-p}, \varepsilon s^q\right\}\right\}\right)$, which satisfies the following for each point pair $v, w \in S$, where $t = \|v - w\|_p$:*

- **(a)** *Threshold: When $q < p$ we have $\|f(v) - f(w)\|_q^q \le \frac{s^q}{Q}$.*
  When $q = p$ we have $\|f(v) - f(w)\|_q^q \le \frac{s^q}{Q_{R/s}}$.
- **(b)** *Bounded expansion and contraction for large scales: When $t > R$, we have $\|f(v) - f(w)\|_q^q = O(t^q)$, and $\|f(v) - f(w)\|_q^q \ge R^q + \varepsilon$.*
- **(c)** *Bi-Lipschitz for intermediate scales: When $1 \le t \le R$, we have $(1-\varepsilon)t^q \le \|f(v) - f(w)\|_q^q \le (1+\varepsilon)t^q$.*
- **(d)** *Bounded expansion for small scales: When $t < 1$, we have $\|f(v) - f(w)\|_q^q \le 1 + \varepsilon$.*

The embedding can be constructed in $O(mk)$ time per point.

Proof. We use the construction of Theorem 3 with the stated value of s, and then scale down by a factor of $Q^{1/q}$ or $Q_{R/s}^{1/q}$. (Note that $Q, Q_{R/s} = \Omega(1)$.) Then the threshold guarantee follows immediately from Theorem 31 and the scaling step.

We will now prove the rest of the theorem for the case $p < q$. First, the bi-Lipschitz claim for values $1 \le t \le R$ follows immediately from Theorem 32 and Lemma 22, when noting that for an appropriate choice of s, $\frac{t}{s} \le \varepsilon^{1/2}(1 + (p-q)\varepsilon^{-(q/2+1)})^{-1/(p-q)}$. Similarly, the bounded expansion claim for $t < 1$ follows from Theorem 33 and the aforementioned bi-Lipschitz guarantee at $t = 1$. Finally, the bounded expansion for $t > R$ follows from Theorem 32, and the bounded contraction follows from Theorem 32 combined with fact that when $a > R/s$, $H(a) > H(R/s)$ (as a consequence of Lemma 234 for an appropriate choice of s).

For $p = q$, we have essentially the same proof, only noting that the function $Q_a = Q_{t/s}$ is monotone decreasing in t, and since $s = O(R^{1/\varepsilon})$ we have for all $1 \le t \le R$ that $\frac{Q_{t/s}}{Q_{R/s}} \le \frac{Q_{1/s}}{Q_{R/s}} = O\left(\frac{1+\varepsilon^{-1}\log R}{1+(\varepsilon^{-1}-1)\log R}\right) = O(1+\varepsilon)$. ◀

4.2 Intrinsic dimensionality reduction

Here we show that the guarantees of Theorem 3 and Theorem 4 can be achieved by (non-oblivious) embeddings whose target dimension in independent of n, and depends only on the doubling dimension of the space. The following lemma is derived by applying the framework of [19] to Theorem 3.

▶ **Lemma 5.** *Let $1 \le q \le p \le 2$ and $0 < \varepsilon < \frac{1}{2}$, and consider an n-point set $S \in l_p^m$. Set threshold $s > 1$. There exists an embedding $f : S \to l_q^k$ for $k = \tilde{O}\Big(\frac{\operatorname{ddim}^2(S)}{\varepsilon^3} \cdot s \cdot \min\left\{s'^{2q}, \max\left\{\frac{s'^{2q-p}}{2q-p}, \varepsilon s'^q\right\}\right\}\Big)$ for $s' = \frac{s \operatorname{ddim}(S)}{\varepsilon}$, which satisfies the following for each pair $v, w \in S$, where $t = \|v - w\|_p$:*

- **(a)** *Threshold: $\|f(v) - f(w)\|_q^q < s^q$.*
- **(b)** *Bi-Lipschitz for intermediate scales: When $1 \le t \le s$, we have $(1-\varepsilon)s^q H(t/s) \le \|f(v) - f(w)\|_q^q \le (1+\varepsilon)s^q H(t/s)$.*
- **(c)** *Strong bounded expansion for small scales: When $t < 1$, for some constant c $\|f(v) - f(w)\|_q^q \le \min\{(1+\varepsilon)s^q H(1/s), c \operatorname{ddim}(S)t\}$.*

Given a point hierarchy for S, the embedding can be constructed in $2^{\tilde{O}\operatorname{ddim}(S)} + O(mk)$ time per point.

Proof. Similar to what was done in [19], we compute for S a padded decomposition with padding s. This is a multiset $[P_1, \ldots, P_m]$ where each partition P_i is a set of clusters, and

every point is s-padded in a $\left(1-\frac{\varepsilon}{s}\right)$ fraction of the partitions. Each cluster has diameter bounded by $O\left(s\,\mathrm{ddim}(S)\right)$, and the support is $m=\tilde{O}(s\varepsilon^{-1}\,\mathrm{ddim}(S))$. Using the hierarchy, this can be done in time $2^{\tilde{O}(\mathrm{ddim}(S))}$ per point [10].

We embed each partition P_i separately as follows: For each cluster $C\in P_i$, we extract from C an $\frac{\varepsilon}{\mathrm{ddim}(S)}$-net $N\subset C$ Now each cluster net has aspect ratio $\frac{s\,\mathrm{ddim}^2(S)}{\varepsilon}$, and so $|N|=\left(\frac{s}{\varepsilon}\right)^{\tilde{O}\,\mathrm{ddim}(S)}$. We then scale N up by a factor of $\frac{\mathrm{ddim}(S)}{\varepsilon}$ so that the minimum distance is 1, invoke the embedding of Theorem 3 with parameter s' on N, and scale back down. This procedure has a runtime cost of $O(mk)$ per point, thresholds all distances at s, and reduces dimension to $\tilde{O}\left(\frac{\mathrm{ddim}(S)}{\varepsilon^2}\cdot\min\left\{s'^{2q},\max\left\{\frac{s'^{2q-p}}{2q-p},\varepsilon s'^{q}\right\}\right\}\right)$. We then concatenate the m cluster partitions together and scale down by $m^{1/q}$, achieveing an embedding of dimension k for the net points. We then extend this embedding to all points using the $c\,\mathrm{ddim}(S)$-factor Lipschitz extension of Lee and Naor [31] for metric space.

For the net points, item 1 holds immediately. For item 2, we note that when $1\le t\le s$, the points fall in the same cluster in a fraction $\left(1-\frac{\varepsilon}{s}\right)$ of the partitions, and these partitions contribute the correct amount to the interpoint distance. However, in a fraction $\frac{\varepsilon}{s}$ of the partition the points are not found in the same cluster, and in these cases the contribution may be as large as s. These partitions account for an additive value of at most $\frac{\varepsilon}{s}\cdot s=\varepsilon\le\varepsilon t$. Item 3 follows in an identical manner.

For the non-net points, item 1 holds since we may assume that all interpoint distance are at most s, since the non-net points outside the convex hull of the net-points can all be projected onto the hull, and this can only improve the quality of the extension. Item 23 follow from the embedding of the $\frac{\varepsilon}{\mathrm{ddim}(S)}$-net: By the guarantees of the extension, an embedded non-net point may be at distance at most $O(\varepsilon)$ from its closest net-point, and then the items follow by an appropriate scaling down of ε. ◀

Similarly, the exact guarantees of Theorem 4 can be achieved by a non-oblivious embedding with dimension independent of n.

▶ **Corollary 6.** *Let $1\le q\le p\le 2$ and consider an n-point set $S\in l_p^m$. Set threshold $s>1$. Fix range $R>1$ and set threshold s as follows: When $q<p$, $s\approx R\varepsilon^{-1/2}(1+(p-q)\varepsilon^{-(q/2+1)})^{1/(p-q)}$, and when $q=p$, $s\approx\max\{R^{1/\varepsilon},R\varepsilon^{-1/2}e^{\varepsilon^{-(\frac{q}{2}+1)}}\}$. Then there exists an embedding $f:S\to l_q^k$ satisfying items 134 of Theorem 4. The target dimension is $k=\tilde{O}\left(\frac{\mathrm{ddim}^2(S)}{\varepsilon^3}\cdot s\cdot\min\left\{s'^{2q},\max\left\{\frac{s'^{2q-p}}{2q-p},\varepsilon s'^{q}\right\}\right\}\right)$ for $s'=\frac{s\,\mathrm{ddim}(S)}{\varepsilon}$. Given a point hierarchy for S, the embedding can be constructed in $2^{\tilde{O}\,\mathrm{ddim}(S)}+O(mk)$ time per point.*

Comment

We conjecture that the $\frac{\mathrm{ddim}^2(S)}{\varepsilon^3}$ terms can be reduced to $\frac{\mathrm{ddim}(S)}{\varepsilon^2}$ by combining the randomness used separately for the construction of the padded decomposition, threshold embedding f, and the Lipschitz extension (as was done in [11] for ℓ_2).

4.3 Snowflake embedding

The embedding of Lemma 5 implies a global dimension-reduction snowflake embedding for ℓ_p. The proof uses the framework presented in [19], and appears in the full version of this paper.

▶ **Lemma 7.** *Let $0<\varepsilon<1/4$, $0<\alpha<1$, and $\tilde{\alpha}=\min\{\alpha,1-\alpha\}$. Every finite subset $S\subset\ell_p$ admits an embedding $\Phi:S\to\ell_q^k$ $(1\le q\le p\le 2)$ for $k=\tilde{O}\Big(\frac{\mathrm{ddim}^6(S)}{\tilde{\alpha}^2\varepsilon^8}\cdot$*

$\min\left\{s'^{2q}, \max\left\{\frac{s'^{2q-p}}{2q-p}, \varepsilon s'^{q}\right\}\right\}\Big)$ *where* $s' = \left(\frac{\mathrm{ddim}(S)}{\varepsilon}\right)^4$ *and* $1 \le \frac{\|\Phi(x)-\Phi(y)\|_q}{\|x-y\|_p^\alpha} \le 1+\varepsilon$, *for all* $x, y \in S$. *Given a point hierarchy for* S, *the embedding can be constructed in* $2^{\tilde{O}\,\mathrm{ddim}(S)} + O(mk)$ *time per point.*

5 Clustering

Here we show that our snowflake embedding can be used to produce faster algorithms for the k-center and min-sum clustering problems. In both cases, we obtain improvements whenever $(\mathrm{ddim}/\varepsilon)^{\Theta(1)}$ is smaller than the ambient dimension.

5.1 k-center clustering

In the k-center clustering problem, the goal is to partition the input set into k clusters, where the objective function to be minimized is the maximum radius among the clusters. Agarwal and Procopiuc [3] considered this problem for d-dimensional set $S \subset \ell_p$, and gave an exact algorithm that runs in time $n^{O(k^{1-1/d})}$, and used this to derive a $(1+\varepsilon)$-approximation algorithm that runs in $O(nd \log k) + (k\varepsilon^{-d})^{O(k^{1-1/d})}$. Here, the cluster centers are points of the ambient space $\mathbb{R}^d$ in which S resides, chosen to minimize the maximum distance from points in S to their nearest center. The authors claim that the algorithm can be applied to the *discrete* problem as well, where all centers are chosen from S, and in fact the algorithm applies to the more general problem where the centers are chosen from a set S' satisfying $S \subset S' \subset \mathbb{R}^d$.[4] Clearly, the runtime of the algorithm can be improved if the dimension is lowered.

▶ **Theorem 8.** *Given* d*-dimensional point set* $S \subset \ell_p$ *for constant* p, $1 < p \le 2$, *a* $(1+\varepsilon)$*-approximate solution to the* k*-center problem on set* S *can be computed in time* $O(nd(2^{\tilde{O}(\mathrm{ddim}(S))} + \log k)) + (k \cdot \varepsilon^{-\mathrm{ddim}(S)\log(1/\varepsilon)/\varepsilon^2})^{k^{1-(\varepsilon/\mathrm{ddim}(S))^{O(1)}}}$. *This holds for the discrete case as well.*

Proof. We first consider the discrete case. Let r^* be the optimal radius. As in [3], we run the algorithm of Feder and Greene [18] in time $O(n \log k)$ to obtain a value $\tilde{r}$ satisfying $r^* \le \tilde{r} < 2r^*$. We then build a hierarchy and extract a $\frac{\varepsilon}{2}\tilde{r}$-net $V \subset S$ in time $2^{\tilde{O}(\mathrm{ddim}(S))}nd$ (assuming the word-RAM model [12]). Since all points of S are contained in k balls of radius r, we have $|V| = k\varepsilon^{-O(\mathrm{ddim}(S))}$. Further, a k-clustering for V is a $(1+O(\varepsilon))$-approximate k-clustering for S. We then apply the snowflake embedding of Lemma 7 to embed V into $(\mathrm{ddim}(S)/\varepsilon)^{O(1)}$-dimensional ℓ_p, and run the exact algorithm of [3] on V in the embedded space. Since a snowflake perserves the ordering of distances, the returned solution for the embedded space is a valid solution in the origin space as well.

Turning to the general (non-discrete) case, the above approach is problematic in that the embedding makes no guarantees on embedded points not in V. To address this, we construct a set W of candidate center points in the ambient space $\mathbb{R}^d$: Recall that the problem of finding the minimum enclosing ℓ_p-ball admits a core-set of size $O(\varepsilon^{-2})$ [9]. (That is, for any discrete point set there exists a subset of size $O(\varepsilon^{-2})$ with the property that the center of the subset is also the center of a $(1+\varepsilon)$-approximation to the smallest ball covering the original set.) We take all distinct subsets $C \subset V$ of size $|C| = O(\varepsilon^{-2})$ and radius at most $\tilde{r}$, compute

[4] The Euclidean core-set algorithm of [9] runs in time $k^{O(k/\varepsilon^2)} \cdot nd$, and can readily be seen to apply to all ℓ_p for constant p, $1 < p \le 2$ (see [41] for a simple approach). The algorithm of [3] compares favorably to the core-set algorithm when d is small.

the center point of each subset (see [41]), and add its candidate center to W. It follows that $|W| = k\varepsilon^{-O(\mathrm{ddim}(S)/\varepsilon^2)}$ and $\mathrm{ddim}(W) = \log \varepsilon^{-O(\mathrm{ddim}(S)/\varepsilon^2)} = O(\mathrm{ddim}(S)\log(1/\varepsilon)/\varepsilon^2)$. As above, we use the snowflake embedding of Lemma 7 to embed $V \cup W$ into $(\mathrm{ddim}(S)/\varepsilon)^{O(1)}$-dimensional ℓ_p and run the exact algorithm of [3] on the embedded space, covering the points of V with candidate centers from W. The returned solution is a valid solution in the origin space as well. ◀

5.2 Min-sum clustering

In the min-sum clustering problem, the goal is to partition the points of an input set into k clusters, where the objective function is the sum of distances between each intra-cluster point pair. Schulman [44] designed algorithms for min-sum clustering under ℓ_1, ℓ_2 and ℓ_2-squared costs, and their runtimes depend exponentially on the dimension. We will obtain faster runtimes for min-sum clustering for ℓ_p $(1 \le p \le 2)$ by using our snowflake embedding as a preprocessing step to reduce dimension. Ultimately, we will embed the space into ℓ_2 using a $\frac{1}{2}$-snowflake, and then solve min-sum clustering with ℓ_2-squared costs in the embedded space; this is equivalent to solving the original ℓ_p problem. We will prove the case of ℓ_1, and the other cases are simpler.

We are given an input set $S \in \ell_1$, and set $c = 2 - \frac{1}{d'}$ for some value $d' = \left(\frac{\mathrm{ddim}(S)}{\varepsilon}\right)^{O(1)}$. We note that the $\frac{1}{c}$-snowflake of ℓ_1 is itself in ℓ_c [17], and also that this embedding into ℓ_c can be computed in polynomial time (with arbitrarily small distortion) using semi-definite programming [19], although the target dimension may be large. We compute this embedding, and then reduce dimension by invoking our snowflake embedding (Lemma 7) to compute a $(\frac{c}{2} = 1 - \frac{1}{2d'})$-snowflake in ℓ_c, with dimension d'. We then consider the vectors to be in ℓ_2 instead of ℓ_c, which induces a distortion of $1 + O(\varepsilon)$. Finally, we run the algorithms of Schulman on the final Euclidean space. As we have replaced the original ℓ_1 distances with their $\left(\frac{1}{c} \cdot \frac{c}{2} = \frac{1}{2}\right)$-snowflake and embedded into ℓ_2, solving min-sum clustering with ℓ_2-squared costs on the embedded space solves the original ℓ_1 problem with distortion $1 + O(\varepsilon)$.

The following lemma follows from the embedding detailed above, in conjunction with [44, Propositions 14,28, full version]. For ease of presentation, we will assume that $k = O(1)$.

▶ **Lemma 9.** *Given a set of n points $S \in \mathbb{R}^d$, set $d' = (\mathrm{ddim}(S)/\varepsilon)^{O(1)}$. a $(1 + O(\varepsilon))$-approximation to the ℓ_p min-sum k-clustering for S, for $k = O(1)$, can be computed*

- *in deterministic time $n^{O(d')}2^{2^{(O(d'))}}$.*
- *in randomized time $n^{O(1)} + 2^{2^{(O(d'))}}(\varepsilon \log n/\delta)^{O(d')}$, with probability $1 - \delta$.*

Acknowledgements. We thank Piotr Indyk, Robi Krauthgamer and Assaf Naor for helpful conversations.

References

1 I. Abraham, Y. Bartal, and O. Neiman. Embedding metric spaces in their intrinsic dimension. In *SODA'08*, pages 363–372. SIAM, 2008.

2 D. Achlioptas. Database-friendly random projections. In *PODS'01*, pages 274–281, New York, NY, USA, 2001. ACM.

3 P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

4 N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *STOC'06*, pages 557–563. ACM, 2006.

5 N. Alon. Problems and results in extremal combinatorics, part I. *Disc. Math*, 273, 2003.

6 N. Alon, M. Bădoiu, E.D. Demaine, M. Farach-Colton, M. Hajiaghayi, and A. Sidiropoulos. Ordinal embeddings of minimum relaxation: General properties, trees, and ultrametrics. *ACM Transactions on Algorithms*, 4(4), 2008.

7 A. Andoni, M. Charikar, O. Neiman, and H.L. Nguyen. Near linear lower bounds for dimension reduction in ℓ_1. In *FOCS'11*. IEEE Computer Society, 2011.

8 Alexandr Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, MIT, 2009.

9 M. Bādoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *STOC'02*, pages 250–257, New York, NY, USA, 2002. ACM.

10 Y. Bartal, L. Gottlieb, T. Kopelowitz, M. Lewenstein, and L. Roditty. Fast, precise and dynamic distance queries. In *SODA'11*, pages 840–853. SIAM, 2011.

11 Y. Bartal, B. Recht, and L. Schulman. Dimensionality reduction: beyond the Johnson-Lindenstrauss bound. In *SODA'11*, pages 868–887. SIAM, 2011.

12 Yair Bartal and Lee-Ad Gottlieb. A linear time approximation scheme for euclidean tsp. In *FOCS'13*, pages 698–706, Washington, DC, USA, 2013. IEEE Computer Society.

13 T. Batu, F. Ergun, and C. Sahinalp. Oblivious string embeddings and edit distance approximations. In *SODA'06*, pages 792–801, 2006.

14 M. Bădoiu, E. Demaine, M. Hajiaghayi, A. Sidiropoulos, and M. Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *APPROX'08*, pages 21–34. 2008.

15 H. Chan, A. Gupta, and K. Talwar. Ultra-low-dimensional embeddings for doubling metrics. *J. ACM*, 57(4):1–26, 2010. doi:10.1145/1734213.1734215.

16 R. Cole and L. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *38th annual ACM symposium on Theory of computing*, pages 574–583, 2006.

17 M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer-Verlag, Berlin, 1997.

18 Tomás Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *STOC'88*, pages 434–444, New York, NY, USA, 1988. ACM.

19 L. Gottlieb and R. Krauthgamer. A nonlinear approach to dimension reduction. In *SODA'11*, pages 888–899. SIAM, 2011.

20 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS'03*, pages 534–543, 2003.

21 S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(14):321–350, 2012.

22 S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

23 P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *FOCS'01*, pages 10–33, 2001.

24 P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

25 P. Indyk and A. Naor. Nearest-neighbor-preserving embeddings. *ACM Trans. Algorithms*, 3(3):31, 2007.

26 W.B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conf. in modern analysis and probability*, pages 189–206. AMS, 1984.

27 W.B. Johnson and A. Naor. The Johnson-Lindenstrauss lemma almost characterizes Hilbert space, but not quite. In *SODA'09*, pages 885–891. SIAM, 2009.

28 W.B. Johnson and G. Schechtman. Embedding l_p^m into l_1^n. *Acta Math.*, 149(1-2):71–85, 1982.

29 R. Krauthgamer and J.R. Lee. Navigating nets: Simple algorithms for proximity search. In *SODA'04*, pages 791–801, January 2004.

30 E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. STOC'98, pages 614–623. ACM, 1998.

31 J.R. Lee and A. Naor. Extending Lipschitz functions via random metric partitions. *Inventiones Mathematicae*, 160:59–95, 2005.

32 Ping Li. Estimators and tail bounds for dimension reduction in ℓ_α $(0 < \alpha \leq 2)$ using stable random projections. In *SODA*, pages 10–19, 2008.

33 G. Lugosi. Concentration-of-measure inequalities. Manuscript, available at `http://www.econ.upf.edu/~lugosi/anu.ps`, 2004.

34 J. Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel J. Math.*, 93:333–344, 1996.

35 M. Mendel and A. Naor. Euclidean quotients of finite metric spaces. *Advances in Mathematics*, 189(2):451–494, 2004.

36 V. D. Milman and G. Schechtman. *Asymptotic Theory of Finite-dimensional Normed Spaces*. Springer-Verlag, Berlin, 1986. With an appendix by M. Gromov.

37 H. Nguyen. Approximate nearest neighbor search in ℓ_p. Manuscript, available at `http://arxiv.org/pdf/1306.3601v1`, 2013.

38 J.P. Nolan. *Stable Distributions – Models for Heavy Tailed Data*. Birkhauser, Boston, 2012. In progress, Chapter 1 online at academic2.american.edu/∼jpnolan.

39 R. Ostrovsky and Y. Rabani. Polynomial time approximation schemes for geometric k-clustering. In *FOCS'00*, pages 349–358. IEEE Computer Society, 2000.

40 R. Ostrovsky and Y. Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002.

41 R. Panigrahy. Minimum enclosing polytope in high dimensions. *CoRR*, 2004.

42 A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

43 G. Schechtman. Dimension reduction in l_p, $0 < p < 2$. Manuscript, available at `http://arxiv.org/pdf/1110.2148v1.pdf`, 2011.

44 L. J. Schulman. Clustering for edge-cost minimization (extended abstract). In *STOC*, pages 547–555. ACM, 2000. Full version available as ECCC report TR99-035. `doi:10.1145/335305.335373`.

45 M. Talagrand. Embedding subspaces of L_1 into l_1^N. *Proc. Amer. Math. Soc.*, 108(2):363–369, 1990.

46 M. Talagrand. Embedding subspaces of L_p in l_p^N. In *Geometric aspects of functional analysis*, volume 77 of *Oper. Theory Adv. Appl.*, pages 311–325. Birkhäuser, Basel, 1995.

47 V.M. Zolotarev. *One-dimensional Stable Distributions*. AMS, 1986.

A Probability theory

The following is a simplified version of Hoeffding's inequality [33]:

▶ **Claim 10.** *Let $X_1, \ldots, X_k$ be independent real-valued random variables, and assume $|X_i| \leq s$ with probability one. Let $\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$. Then for any $z > 0$, $\Pr\left[|\bar{X} - \mathbb{E}[\bar{X}]| \geq z\right] \leq 2\exp\left(-\frac{2kz^2}{s^2}\right)$*

The following is a restatement of Bennett's inequality [33]:

▶ **Claim 11.** *Let $X_1, \ldots, X_k$ be independent real-valued random variables, and assume $|X_i| \leq s$ with probability one. Let $\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$ and set $\sigma^2 = \frac{1}{k}\sum_{i=1}^{k} \mathrm{Var}\{X_i\}$. Then for any $z > 0$, $\Pr\left[|\bar{X} - \mathbb{E}[\bar{X}]| \geq z\right] \leq 2\exp\left(-\frac{k\sigma^2}{s^2} V\left(\frac{sz}{\sigma^2}\right)\right)$ where $V(u) = (1+u)\ln(1+u) - u$ for $u \geq 0$.*

Note that for $u \geq 1$, we have $V(u) = \Omega(u \log u)$, while for $0 \leq u < 1$, $V(u) = \Theta(u^2)$.

Testing Convexity of Figures Under the Uniform Distribution

Piotr Berman[1], Meiram Murzabulatov[*2], and Sofya Raskhodnikova[†3]

1 **Pennsylvania State University, University Park, USA**
berman@cse.psu.edu
2 **Pennsylvania State University, University Park, USA**
mzm269@psu.edu
3 **Pennsylvania State University, University Park, USA**
sofya@cse.psu.edu

Abstract

We consider the following basic geometric problem: *Given $\epsilon \in (0, 1/2)$, a 2-dimensional figure that consists of a black object and a white background is ϵ-far from convex if it differs in at least an ϵ fraction of the area from every figure where the black object is convex. How many uniform and independent samples from a figure that is ϵ-far from convex are needed to detect a violation of convexity with probability at least 2/3?* This question arises in the context of designing property testers for convexity. Specifically, a (1-sided error) tester for convexity gets samples from the figure, labeled by their color; it always accepts if the black object is convex; it rejects with probability at least 2/3 if the figure is ϵ-far from convex.

We show that $\Theta(\epsilon^{-4/3})$ uniform samples are necessary and sufficient for detecting a violation of convexity in an ϵ-far figure and, equivalently, for testing convexity of figures with 1-sided error. Our testing algorithm runs in time $O(\epsilon^{-4/3})$ and thus beats the $\Omega(\epsilon^{-3/2})$ sample lower bound for learning convex figures under the uniform distribution from [26]. It shows that, with uniform samples, we can check if a set is approximately convex much faster than we can find an approximate representation of a convex set.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Convex sets, 2D geometry, randomized algorithms, property testing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.17

1 Introduction

Convexity is a fundamental property of geometric objects that plays an important role in algorithms, learning, optimization, and image processing. Some algorithms require that their input is a convex set. However, it is infeasible to check whether an infinite (or a very large) set is indeed convex. How quickly can we check whether it is approximately convex? Can it be done faster than learning an approximate representation of a convex set?

Property testing [25, 12] is a formal study of fast algorithms that determine whether a given object approximately satisfies the desired property. There is a line of work on property testing and sublinear algorithms for geometric convexity[1] and other visual properties

* The second author was supported by NSF CAREER award CCF-0845701.
† The third author was supported by NSF award CCF-1422975 and NSF CAREER award CCF-0845701.
1 Property testing of convexity (and submodularity) of functions has also been investigated [18, 27, 22, 6, 5].


© Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova;
licensed under Creative Commons License CC-BY

(see [20, 19, 29, 14, 15, 16] and references therein). Previous works on testing geometric convexity [20, 19, 29] assume that the tester can query an arbitrary point in the input and find out whether it belongs to the object.

We study the problem of property testing convexity of 2-dimensional figures with only uniform and independent samples from the input. A figure (U, C) consists of a compact convex universe $U \subseteq \mathbb{R}^2$ and a measurable subset $C \subseteq U$. The set C can be thought of as a black object on a white background $U \setminus C$. A figure (U, C) is convex iff C is convex. The relative distance between two figures (U, C) and (U, C') over the same universe is the probability of the symmetric difference between them under the uniform distribution on U. A figure (U, C) is ϵ-far from convex if the relative distance from (U, C) to every convex figure (U, C') over the same universe is at least ϵ.

▶ **Definition 1.1.** Given a proximity parameter $\epsilon \in (0, 1/2)$ and error probability $\delta \in (0, 1)$, a (1-sided error) ϵ-tester for convexity accepts if the figure is convex and rejects with probability at least $1 - \delta$ if the figure is ϵ-far from convex[2]. A tester is *uniform* if it accesses its input via uniform and independent samples from U, each labeled with a bit indicating whether it belongs to C.

Our goal is to determine the smallest number of samples necessary and sufficient for ϵ-testing convexity under the uniform distribution.

An easy upper bound for this problem can be obtained from a connection between (proper) PAC-learning and property testing [12] and the work of Schmeltz [26] who gives a PAC-learner of convex d-dimensional sets under the uniform distribution. Specifically, for two dimensions, he shows that $\Theta(\epsilon^{-3/2})$ samples are necessary and sufficient[3]. In other words, Schmeltz [26] shows that it suffices to take $O(\epsilon^{-3/2})$ uniform and independent samples from a convex shape of unit area, so that the convex hull of these samples has area at least $1 - \epsilon$ with probability at least $2/3$; and moreover, for a disk, $\Omega(\epsilon^{-3/2})$ samples are necessary to satisfy this requirement.

We prove that $\Theta(\epsilon^{-4/3})$ uniform samples are necessary and sufficient for ϵ-testing 2-dimensional convexity under the uniform distribution with 1-sided error. Our algorithm runs in time $O(\epsilon^{-4/3})$ and thus beats the $\Omega(\epsilon^{-3/2})$ lower bound for learning convex figures under the uniform distribution. It shows that, with uniform samples, we can check if a set is approximately convex much faster than we can find an approximate representation of a convex set.

Our results imply the same upper and lower bounds for convexity testing in the pixel model of [20]. The input representation in that model differs from ours only in that the images are discretized, whereas we consider continuous figures. In the pixel model, an image is specified by an $n \times n$ matrix of Boolean pixel values, representing a discretization of a black-and-white image in $[0, 1]^2$. An algorithm is *adaptive* if it is allowed to query arbitrary entries in the matrix and its queries depend on answers to previous queries. In [20], an adaptive tester for convexity[4] that makes $O(\epsilon^{-2})$ queries is presented. Our upper and lower

[2] If δ is not specified, it is assumed to be $1/3$. By standard arguments, the error probability can be reduced from $1/3$ to an arbitrarily small δ by running the tester $O(\log 1/\delta)$ times.

[3] The PAC learner in [26] is not distribution-free—it works only with respect to the uniform distribution. The VC dimension of convexity, even in two dimensions, is infinite, so convexity is not PAC-learnable under arbitrary distributions.

[4] The VC dimension of convexity of $n \times n$ images is $\Theta(n^{2/3})$, since this is the maximum number of vertices of a convex lattice polygon in an $n \times n$ lattice [2]. Therefore, proper PAC-learners for convex images in the pixel model (that work with respect to all distributions) cannot have complexity independent of n.

bounds hold for the pixel model, provided that n is sufficiently large to ensure that every convex area we consider in our analysis has some pixels (i.e., non-zero probability mass when we sample uniformly from the $n \times n$ matrix). See the full version of this article for details.

Our techniques. We present a (1-sided error) uniform tester for convexity of 2-dimensional figures with sample and time complexity $O(\epsilon^{-4/3})$ and prove a matching lower bound.

Our tester is the natural one: it computes the convex hull of sampled black points and rejects iff it contains a sampled white point. In other words, it rejects only if it finds a convexity violation. How many points are needed to witness such a violation? The smallest number of points is three: a white point between two black points on the same line. However, a uniform tester is unlikely to sample three points on the same line. If the points are in general position, the smallest number is four: three black and one white in the triangle formed by the three black points. A natural way to exploit this in the analysis is to divide the figure into different parts (which we call patterns) with four regions each, such that we are likely to sample a 4-point witness of non-convexity from the corresponding regions of some pattern. However, the higher the number of regions in each pattern from which we require the tester to sample at least one point, the more samples it needs.

To reduce the number of regions in the patterns, we use a *central point* defined in terms of the Ham Sandwich cut of black points[5]. Such cuts have been studied extensively (see, e.g., [10, p. 356] and [17]), for example, in the context of range queries. Specifically, a *central point* is the intersection of two lines that partition the figure into four regions, each with black area at least $\epsilon/4$. A central point is overwhelmingly likely to end up in the convex hull of sampled black points. So, even though the central point itself is not likely to be sampled, it becomes a de facto part of a witness that comes nearly for free. Conditioned on the central point indeed being in the convex hull of sampled black points, our witness only needs 3 additional points: two black and one white, such that the white point is in the triangle formed by the two black points and the central point. This will ensure that the white point is in the convex hull of sampled black points, that is, a violation of convexity is detected.

The main technical part of the analysis is finding disjoint 3-region patterns in the figure, such that the algorithm is likely to sample a 3-point witness from at least one of the patterns. We can show that if the figure is ϵ-far from convex then the white regions of the patterns together occupy a fraction of the area proportional to ϵ. We have two separate lines of argument for the case when there are many white regions in the patterns that have large white area and for the case when the white area is distributed more evenly among the patterns. These two cases are analyzed in the recoloring and the sweeping phases of the analysis, respectively. The main geometric construction of the patterns appears in the sweeping phase (which uses sweeping lines to construct the patterns).

We remind the reader that these phases are only used in the analysis. Our algorithm is extremely simple and natural.

To prove our lower bound, we construct hard instances, for which a uniform tester for convexity needs to get a 3-point witness, with points coming from different specified regions, in order to detect a violation of convexity. Intuitively, the fact that the number of points in a witness is also 3, as in the analysis of the algorithm, allows us to get a matching lower bound.

Related Work in Property Testing. We already mentioned work on testing geometric convexity [20, 19, 29] in the model similar to ours, but where the tester can query arbitrary

[5] Our central points are related to the well studied centerpoints [10] and Tukey medians [30]. The guarantee for a centerpoint is that every line that passes through it creates a relatively balanced cut.

points in the input. There is another line of work on testing geometric properties, initiated by Czumaj, Sohler, and Ziegler [9] and Czumaj and Sohler [8], where the input is a set of points represented by their coordinates. The allowed queries and the distance measures on the input space considered in those works are different from ours. The most related problem to ours is that of testing whether points, represented by their coordinates, are in convex position or far from having that property (for example, in the sense that at least an ϵ fraction of points has to be changed to ensure that they are in a convex position). In [8], several sophisticated distance measures and powerful queries to the input are considered. For example, a *range query*, given a range and a natural number i, returns the i-th point in the range. Chazelle and Seshadhri [7], in another related work, give a property tester for convexity of polygons represented by doubly-linked lists of their edges. In contrast to these works, we consider only extremely simple access to the input, measure the distance between figures by the area on which they differ, and can deal with continuous figures.

Related Work in Computational Geometry. The random process of sampling uniform and independent points from a convex body has been studied extensively. (We stress that, in our problem, the input figure is not guaranteed to be convex. Instead, we are trying to distinguish convex figures from those that are far from convex.) The expected number of vertices of a convex hull of n such samples is well understood. For example, in 2 dimensions, it is $O(n^{1/3})$ when the object is a disk [23] and $O(k \log n)$ when the object is a convex k-gon [24]. (See also [13] for simple proofs of these statements.) Bárány and Füredi [3] analyze the probability that n points chosen from the d-dimensional unit ball are in the convex position. Eldan [11] shows that no algorithm can approximate the volume of a convex body in $\mathbb{R}^d$, with high probability and up to a constant factor, when provided only with a polynomial in d number of random points.

2 Preliminaries on Poissonization

The analysis of our algorithm uses a technique called *Poissonization* [28], in which one modifies a probabilistic experiment to replace a fixed quantity (e.g., the number of samples) with a variable one that follows a Poisson distribution. This breaks up dependencies between different events, and makes the analysis tractable. The Poisson distribution with parameter $\lambda \geq 0$, denoted $\mathrm{Po}(\lambda)$, takes each value $x \in \mathbb{N}$ with probability $e^{-\lambda}\lambda^x/x!$. The expectation and variance of a random variable distributed according to $\mathrm{Po}(\lambda)$ are both λ.

▶ **Definition 2.1.** A *Poisson-s tester* is a uniform tester that takes a random number of samples distributed as $\mathrm{Po}(s)$.

The following lemma is paraphrased from [21, Lemma 5.3], except that the terminology is adjusted to fit in with our application. The proofs from [21] work nearly verbatim. Even though part (a) is not stated in [21], the proof for this part is similar to the proof of part (b). We use part (a) to analyze our algorithm and part (b) to prove lower bounds on the sample complexity (so, we do not need a statement about the running time in part (b)).

We use $[k]$ to denote the integer set $\{1, \ldots, k\}$.

▶ **Lemma 2.2** (Poissonization Lemma)**.**

(a) *Poisson algorithms can simulate uniform algorithms. Specifically, for every Poisson-s tester $\mathcal{A}$ for property $\mathcal{P}$ with error probability δ, there is a uniform tester $\mathcal{A}'$ for $\mathcal{P}$ that uses at most $2s$ samples and has error probability at most $\delta + 4/s$. Moreover,*

- *if $\mathcal{A}$ has 1-sided error, so does $\mathcal{A}'$;*

- *if $\mathcal{A}$ runs in time $t(x)$ when it takes x samples, then $\mathcal{A}'$ has running time $O(t(2s))$.*

(b) *Uniform algorithms can simulate Poisson algorithms. Specifically, for every uniform tester $\mathcal{A}$ for property $\mathcal{P}$ that uses at most s samples and has error probability δ, there is a Poisson-$2s$ tester $\mathcal{A}'$ for $\mathcal{P}$ with error probability at most $\delta + 4/s$. Moreover,*

- *if $\mathcal{A}$ has 1-sided error, so does $\mathcal{A}'$.*

(c) *Let Ω be a sample space from which a Poisson-s algorithm makes uniform draws. Suppose we partition Ω into sets $\Omega_1, \ldots, \Omega_k$ (e.g., these sets can correspond to disjoint areas of the figure from which points are sampled), where each outcome is in set Ω_i with probability p_i for $i \in [k]$. Let X_i be the total number of samples in Ω_i seen by the algorithm. Then X_i is distributed as $\mathrm{Po}(p_i \cdot s)$. **Moreover, random variables X_i are mutually independent for all $i \in [k]$.***

3 Uniform Tester for Convexity

In this section, we give our optimal uniform convexity tester for figures.

▶ **Theorem 3.1.** *There is a uniform (1-sided error) ϵ-tester for convexity with sample and time complexity $O(\epsilon^{-4/3})$.*

Proof. We start by reducing the problem to the special case when the universe U is an axis-aligned rectangle of unit area. Consider a convex two-dimensional set U'. It is not hard to show that U' is contained in a rectangle U whose area is at most twice the area of U'. If we consider figures (U, C) instead of (U', C), relative distances between figures increase by at most a factor of 2. We can simulate a tester that works on (U, C) while having access to (U', C) without affecting asymptotic complexity. Therefore, we can assume w.l.o.g. that U is a rectangle. Finally, note that if U does not have unit area, the figure can be rescaled, and if U is not axis-aligned, the figure can be rotated.

By the Poissonization Lemma (Lemma 2.2), it is sufficient to prove that there is a 1-sided error Poisson-s convexity tester with $s = O(\epsilon^{-4/3})$, error probability $\delta \leq 0.333$, and linear running time in the number of samples[6]. By standard arguments, such a tester can be obtained from a tester as described, but with *expected* linear running time and error probability $\delta \leq 0.33$. Our Poisson convexity tester satisfying the latter requirements is Algorithm 1. To make the algorithm and its analysis easier to visualize, we color points in C black and points in $U \setminus C$ white. (In the analysis, we recolor some of the black points to make them violet.)

Query and Time Complexity. Algorithm 1 queries $q = \mathrm{Po}(s)$ points, where $s = O(\epsilon^{-4/3})$. Since the x-coordinates of the sampled q points are distributed uniformly in the interval corresponding to the length of the rectangle U, they can be sorted in expected time $O(q)$ by subdividing this interval into s subintervals of equal length, and using them as buckets in the bucket sort. Andrew's monotone chain algorithm finds the convex hull of a set of q sorted points in time $O(q)$. Since the expectation of q is $O(\epsilon^{-4/3})$, Algorithm 1 runs in expected time $O(\epsilon^{-4/3})$. By the discussion preceding the algorithm, we get a uniform algorithm with the worst case running time $O(\epsilon^{-4/3})$ and with a slightly larger error δ than in Algorithm 1.

[6] Our proof works for sufficiently small ϵ. Suppose an algorithm works for all $\epsilon \leq \epsilon_0$. For $\epsilon > \epsilon_0$, we can run it with parameter ϵ_0 in constant time and obtain the required guarantees, since an ϵ_0-tester is also an ϵ-tester for $\epsilon_0 < \epsilon$.

Algorithm 1: Uniform ϵ-*tester* for convexity (when U is an axis-aligned rectangle).

input : parameter $\epsilon \in (0, 1/2)$;
access to uniform and independent samples from (U, C).

1 Set $s = 50\epsilon^{-4/3}$. Sample Po(s) points from U uniformly and independently at random.
2 Bucket sort sampled black points by the x-coordinate into s bins to obtain list S_B. Similarly, compute S_W for the sampled white points.

`// Check if the convex hull of` S_B `contains a pixel from` S_W`.`

3 Use Andrew's monotone chain convex hull algorithm [1] to compute UH(S_B) and LH(S_B), the upper and the lower hulls of S_B, respectively, sorted by the x-coordinate.
4 Merge sorted lists S_W, UH(S_B) and LH(S_B) to determine for each point w in S_W its left and right neighbors in UH(S_B) and LH(S_B). If w lies between the corresponding line segments of the upper and lower hulls, **reject**.
5 **Accept**.

Correctness. If figure (U, C) is convex, Hull(S_B) contains only black points, and Algorithm 1 always accepts. From now on, we consider a figure (U, C) that is ϵ-far from convexity. We show that Algorithm 1 rejects it with probability at least 0.33.

For a set (region) R, let Hull(R) denote its convex hull and $A(R)$ denote its area or, equivalently, its probability mass under the uniform distribution of points in U. (It is equivalent because we assumed w.l.o.g. that U has unit area.) For a region R, its area of a certain color (e.g., its black area) is the probability mass of points of that color in R. For example, initially, the black area of R is $A(R \cap C)$.

We start by defining a special point, which belongs, with high probability, to Hull(S_B) constructed by Algorithm 1.

▶ **Definition 3.2** (Central point)**.** A point is *central* if it is the intersection of two lines such that each of the closed quadrants formed by these lines has black area at least $\epsilon/4$, i.e., the intersection of C and each quadrant has area at least $\epsilon/4$. We say that the two lines *define* this central point.

▶ **Claim 3.3.** If $A(C) \geq \epsilon$ then U contains a central point.

Proof. By a continuity argument, there exists a line that bisects C into two sets of area $A(C)/2$ each. By the Ham Sandwich Theorem, applied to the two resulting sets, for every such line, there exists another line that bisects both of the resulting sets into sets of area $A(C)/4$ each. By Definition 3.2, the intersection point of the two lines is a central point. ◀

Since the empty set is convex and (U, C) is ϵ-far from convex, $A(C) \geq \epsilon$. Thus, by Claim 3.3, there is a central point in U. Denote one such point by u. The central point u is fixed throughout the analysis of Algorithm 1. Next, we bound the probability that u is in Hull(S_B). Note that u is just a point in U, not necessarily a sample.

▶ **Lemma 3.4.** *The probability that the central point u is not in Hull(S_B), where S_B is the set of black points sampled by Algorithm 1, is at most 0.01.*

Proof. Let ℓ_1^u and ℓ_2^u be the two lines that define the central point u (see Definition 3.2). If the algorithm samples a black point from each open quadrant formed by ℓ_1^u and ℓ_2^u then the central point u is in the convex hull of the four points sampled from each quadrant, i.e., it is in the convex hull of all sampled black points. By the Poissonization Lemma 2.2, the

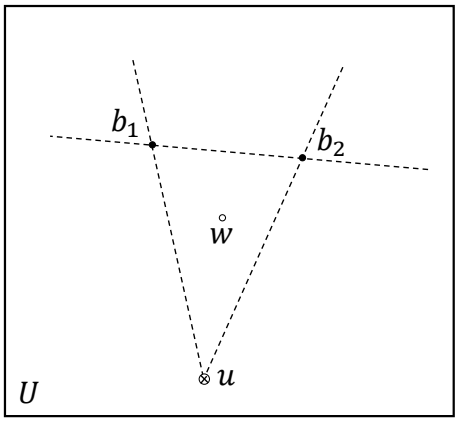


Figure 1 A witness triple (b_1, b_2, w).

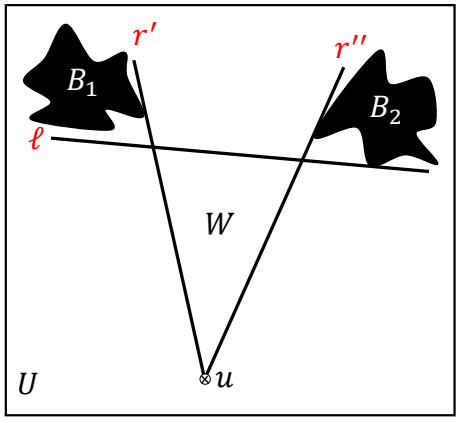


Figure 2 A pattern.

number of samples from each quadrant has distribution $\mathrm{Po}(p \cdot s)$, where $p \geq \epsilon/4$. Thus, the probability that the algorithm fails to sample a black point from one particular quadrant is at most $e^{-\epsilon \cdot s/4}$. For $s = 50 \cdot \epsilon^{-4/3}$, the value $e^{-\epsilon \cdot s/4} \leq e^{-6}$. By a union bound, the probability that the algorithm will not sample a black point from at least one of the four open quadrants is at most $4 \cdot e^{-6} < 0.01$. Thus, the probability that $u \notin \mathrm{Hull}(S_B)$ is at most 0.01. ◀

▶ **Definition 3.5** (A witness triple). Recall that u is a fixed central point. A triple of points (b_1, b_2, w) is a *witness triple* if b_1 and b_2 are black, and w is a white point contained in the triangle $\triangle u b_1 b_2$. (See Figure 1. Note that $\triangle u b_1 b_2$ could be degenerate, i.e., all three vertices could lie on the same line.)

If the central point u is indeed contained in the convex hull of all black points sampled by Algorithm 1 and if, in addition, the algorithm samples a witness triple, then the algorithm rejects because it found a white sample w in the convex hull of black samples. By Lemma 3.4, the first event fails to occur with probability at most 0.01. If we get a guarantee that the algorithm fails to sample a witness triple with probability at most 0.32 then, by a union bound, the algorithm fails to reject with probability at most $0.01 + 0.32 < 0.33$, as required.

The required guarantee follows from Propositions 3.8 and 3.9 that we will prove in Sections 3.1 and 3.2, respectively. This completes the proof of Theorem 3.1. ◀

Propositions 3.8 and 3.9 break the analysis into two cases, depending on the number of certain patterns in the input. Patterns are parts of the input from which, intuitively, we are likely to sample a witness triple.

▶ **Definition 3.6** (A pattern). A *pattern* consists of two rays r' and r'', emanating from the central point u, a line ℓ that crosses the two rays, and disjoint sets B_1 and B_2 of black points for which the following conditions hold. Set B_1 (respectively, B_2) has area $t = 0.025 \cdot \epsilon^{3/2}$ and is a subset of the infinite region formed by the line ℓ and the ray r' (respectively, by ℓ and r''). (See Figure 2.) If, in addition, the white area of the triangular region W, formed by ℓ, r', and r'', is at least $0.025 \cdot \epsilon^{4/3}$, then the pattern is called a *white-heavy* pattern. Otherwise, it is called a *white-light* pattern.

Observe that, for any pattern, a point from B_1, a point from B_2, and a white point from W form a witness triple.

▶ **Claim 3.7.** For a given pattern and $i \in [2]$, let E_i be the event that Algorithm 1 samples a point from B_i. Then $\Pr[E_1 \cap E_2] \geq 0.39 \cdot \epsilon^{1/3}$ for sufficiently small ϵ.

Proof. Recall that $t = 0.025 \cdot \epsilon^{3/2}$. By definition of a pattern, B_1 has area t. Therefore, by the Poissonization Lemma (Lemma 2.2), $\Pr[E_1] = 1 - e^{-ts} = 1 - e^{-1.25\epsilon^{1/6}}$. By Taylor expansion, $1 - e^{-x} \geq x - \frac{x^2}{2} \geq x/2$ for $x \in (0, 1.5)$. Therefore, $\Pr[E_1] \geq 0.625\epsilon^{1/6}$ for sufficiently small ϵ. The same bound holds for event E_2.

Since B_1 and B_2 are disjoint, events E_1 and E_2 are independent. Thus, $\Pr[E_1 \cap E_2] = \Pr[E_1] \cdot \Pr[E_2] \geq (0.625\epsilon^{1/6})^2 \geq 0.39 \cdot \epsilon^{1/3}$, as required. ◀

To explain the two cases considered in Propositions 3.8 and 3.9, we describe our analysis in two phases, *recoloring* and *sweeping*.

3.1 The Recoloring Phase

In the recoloring phase of the analysis, we change the color of some points in the input figure from black to violet. While there is a white-heavy pattern in the figure, we repeat the following mental experiment.

1. Choose a white-heavy pattern. Let B_1 and B_2 be the associate sets of black points.
2. If it is iteration i of the mental experiment, let $V_1^i = B_1$ and $V_2^i = B_2$.
3. Recolor violet all points in V_1^i and V_2^i (so that they are not used in subsequent iterations and the next phase of the analysis).

▶ **Proposition 3.8.** When the input figure is ϵ-far from convexity, if the number of iterations in the recoloring phase is at least $9 \cdot \epsilon^{-1/3}$ then the tester samples a witness triple with probability at least 0.68.

The proof of the proposition is deferred to the full version.

3.2 The Sweeping Phase

In this section, we prove Proposition 3.9, the main technical component in the proof of Theorem 3.1.

▶ **Proposition 3.9.** When the input figure is ϵ-far from convexity, if the number of iterations in the recoloring phase is less than $9 \cdot \epsilon^{-1/3}$ then the tester samples a witness triple with probability at least 0.68.

Proof. If the recoloring phase has less than $9 \cdot \epsilon^{-1/3}$ iterations then, for sufficiently small ϵ, the violet area (that was black in the original input) is at most

$$9 \cdot \epsilon^{-1/3}(2 \cdot 0.025 \cdot \epsilon^{3/2}) = 0.45 \cdot \epsilon^{7/6} < 0.04 \cdot \epsilon. \tag{1}$$

In the sweeping phase of the analysis, we iteratively construct a set of *sweeping* lines L. Each line $\ell \in L$ is associated with a set of black points S_ℓ of area at most $4t$. The set S_ℓ lies in the half-plane defined by ℓ that does not contain the central point u. Sets S_ℓ associated with different lines ℓ are disjoint. Lines ℓ whose sets S_ℓ have area exactly $4t$ are collected in L^*. For each such line ℓ, we define an *anchor* point p_ℓ. Later, we use the sets S_ℓ of lines $\ell \in L^*$ to create white-light patterns whose associated regions B_1, B_2, and W are all disjoint from each other and whose W regions jointly cover a large white area. Each S_ℓ will be partitioned into four sets of the form B_1, B_2 for the patterns. The anchor points are used to partition sets S_ℓ and to choose subsequent sweeping lines. We describe the construction

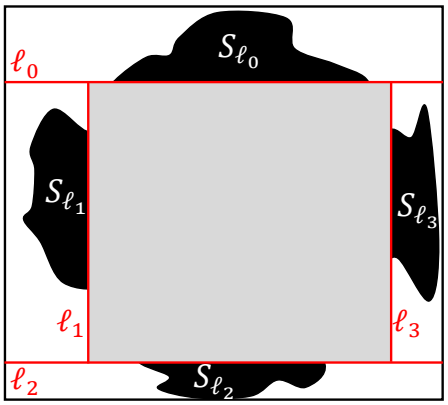


Figure 3 Bounding rectangle R.

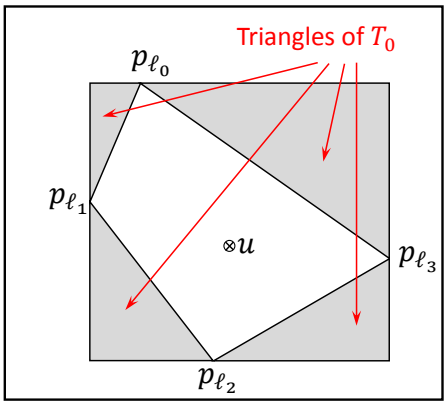


Figure 4 Triangles of T_0.

of sweeping lines next. We start by constructing a *bounding rectangle* R formed by initial sweeping lines and then add more sweeping lines.

Recall that $t = 0.025 \cdot \epsilon^{3/2}$ and that some of the originally black points became violet in the recoloring phase and thus are no longer black. Also, recall that w.l.o.g. we can assume that U is a rectangle. Now we construct lines $\ell_0, \ell_1, \ell_2, \ell_3$ that form a *bounding rectangle* R inside U. Let ℓ_0 and ℓ_2 be the horizontal lines such that $A(S_{\ell_0}) = A(S_{\ell_2}) = 4t$, where S_{ℓ_0} (respectively, S_{ℓ_2}) denote the set of all black points above ℓ_0 (respectively, below ℓ_2). Initially, $L = L^* = \{\ell_0, \ell_2\}$.

▶ **Definition 3.10** (Anchor points)**.** Consider a line ℓ that does not contain the central point u. Define H_ℓ^u (resp., H_ℓ) to be the closed half-plane formed by ℓ that contains (resp., does not contain) u. For a set S of points in H_ℓ, the *anchor point* of S on ℓ is the intersection of the line ℓ and the ray emanating from u that bisects the set S into two sets of equal area. For a sweeping line $\ell \in L^*$ and the associated set S_ℓ, let p_ℓ denote the anchor point of S_ℓ on ℓ.

Initially, the set of anchor points $P = \{p_{\ell_0}, p_{\ell_2}\}$. Now we define the vertical lines ℓ_1 and ℓ_3. The set S_{ℓ_1} (respectively, S_{ℓ_3}) will be the set of all black points to the left of ℓ_1 (respectively, to the right of ℓ_3) between ℓ_0 and ℓ_2. See Figure 3. Intuitively, for $i \in \{0, 1, 2, 3\}$, we move the line ℓ_i in parallel starting from the boundary of U and stop moving it immediately when it "sweeps" a set of black points (not "swept" by previous lines) whose area is equal to $4t$. However, the lines ℓ_1 and ℓ_3 will stop before "sweeping" black area $4t$ if they encounter an anchor point. Specifically, for $i = 1, 3$, we require that $A(S_{\ell_i}) \leq 4t$ and that the half-plane $H_{\ell_i}^u$ must contain both anchor points p_{ℓ_0} and p_{ℓ_2}. Let ℓ_i be the vertical line with the maximum $A(S_{\ell_i})$ that satisfies these requirements. If $A(S_{\ell_i}) < 4t$ then ℓ_i is added only to L. Otherwise, it is added to L and L^* and its anchor point p_{ℓ_i} (given by Definition 3.10) is added to P.

The bounding rectangle R is formed by the lines $\ell_0, \ell_1, \ell_2, \ell_3$. At this point, $2 \leq |P| \leq 4$. Let $\mathcal{T}_0$ be the set of (at most four) triangles formed by removing the (possibly degenerate) quadrilateral Hull(P) from the rectangle R. See Figure 4.

▶ **Definition 3.11** (Line and ray notation.)**.** For two points x and y, let $r(x, y)$ denote the ray that emanates from x and passes through y, and let $\ell(x, y)$ denote the line through x and y.

We describe a procedure that completes the construction of L, L^*, and P by inductively constructing sets $\mathcal{T}_i$, starting from the set $\mathcal{T}_0$, defined before. (Recall that this construction is needed only in the analysis, not in the algorithm.)

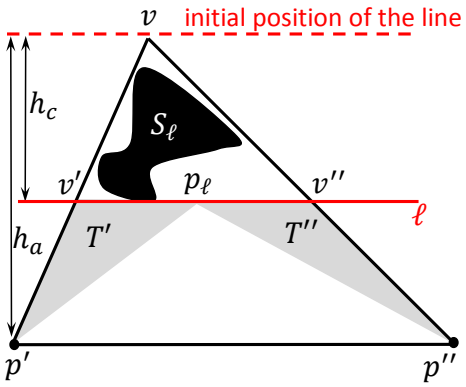


Figure 5 Constructing triangles of $\mathcal{T}_i$ in a triangle of $\mathcal{T}_{i-1}$.

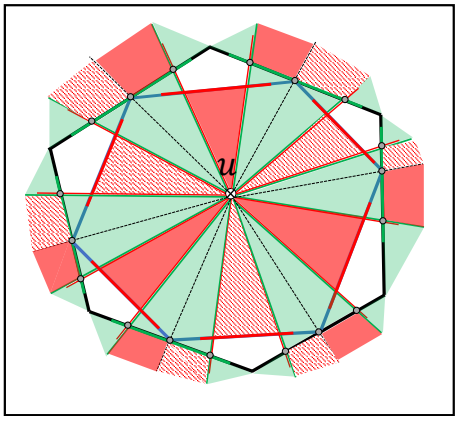


Figure 6 An illustration of type 1 and type 2 patterns when u is inside Hull(P).

1. Let $m = \log(2/\epsilon)/2$ (w.l.o.g. assume that $\log(2/\epsilon)/2$ is an integer[7]).
2. Initially, $\mathcal{T}_i = \emptyset$, for every $i \in [m]$, and sets L, L^*, P and $\mathcal{T}_0$ are as defined earlier.
3. For every $i = 1, 2, \ldots, m$ and every triangle $T \in \mathcal{T}_{i-1}$, do the following:
   a. Let v be the only vertex of T that is not in P; let $p', p'' \in P$ be its other two vertices.
   b. If the black area in $T = \triangle vp'p''$ is less than $4t$ then let $\ell = \ell(p', p'')$, define S_ℓ to be the set of black points in $\triangle vp'p''$ and add ℓ to L.
   c. Otherwise, let ℓ be the line parallel to the base $p'p''$ that intersects the sides vp' and vp'' at v' and v'', respectively, such that the black area of $\triangle vv'v''$ is $4t$ (see Figure 5). Let S_ℓ be the set of black points in $\triangle vv'v''$. Let p_ℓ be the anchor point of S_ℓ on ℓ. Add line ℓ to L and L^*, point p_ℓ to P, and triangles $\triangle p_\ell p'v'$ and $\triangle p_\ell p''v''$ to $\mathcal{T}_i$.
4. This completes the construction of L, L^*, P and $\mathcal{T}_i$, for every $i \in [m]$.

Intuitively, we move a line starting from the vertex v towards the base $p'p''$ keeping it parallel to the base. We stop moving it when it reaches the side $p'p''$ or when it "sweeps" a black area $4t$ in $\triangle vp'p''$.

The goal of sweeping is to eventually construct patterns. Black sets for the patterns will be obtained from the sets S_ℓ for lines in L^*, whereas white regions for the patterns will come from Hull(P). The area between the polygon formed by the sweeping lines and Hull(P) is "uninvestigated" and not useful in the construction of patterns. In order to reduce the uninvestigated area quickly (with a few sweeps), we take sweeping lines *parallel* to the bases of uninvestigated rectangles. After sweeping, only triangles in $\mathcal{T}_m$ remain uninvestigated.

▶ **Lemma 3.12.** *The sum of the areas of all triangles in $\mathcal{T}_m$ is at most $\epsilon/2$.*

Proof. Fix $i \in [m]$. Consider a triangle $T \in \mathcal{T}_{i-1}$ and the two triangles $T', T'' \in \mathcal{T}_i$ obtained in the procedure that constructs sets $\mathcal{T}_j$, for $j \in [m]$. Recall that in the procedure, triangle $T = \triangle vp'p''$, where v is the only vertex of T that is not in P and $p', p'' \in P$ are its other two vertices. Moreover, $T' = \triangle p_\ell p'v'$ and $T'' = \triangle p_\ell p''v''$, where p_ℓ is the anchor point on the line ℓ that is parallel to the base p', p'', whereas v' and v'' are the intersection points of ℓ and the sides vp' and vp'', respectively. (See Figure 5.)

[7] If $\epsilon \in (1/2^j, 1/2^{j-2})$ for some odd j, to ϵ-test $\mathcal{P}$ it is enough to ϵ'-test $\mathcal{P}$ with $\epsilon' = 1/2^j$ since $\epsilon' < \epsilon$.

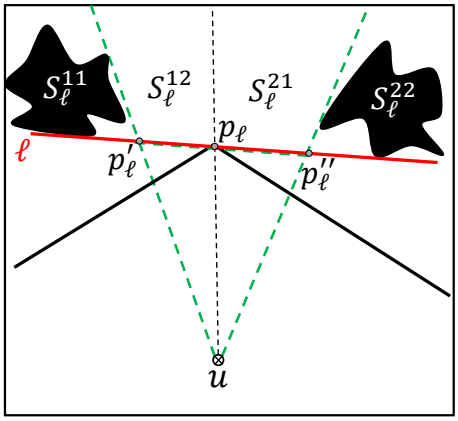


Figure 7 A pattern of type 1.

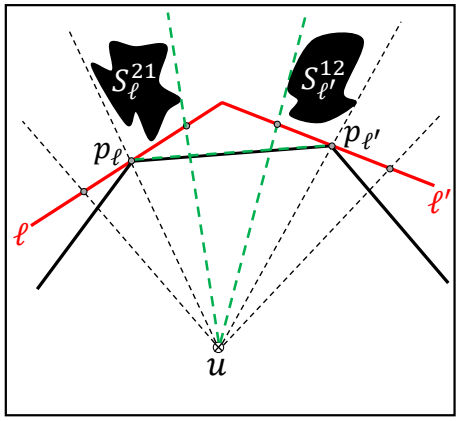


Figure 8 A pattern of type 2.

▶ **Claim 3.13.** For the triangles $T \in \mathcal{T}_{i-1}$ and $T', T'' \in \mathcal{T}_i$, defined above,

$$A(T') + A(T'') \leq \frac{A(T)}{4}.$$

Proof. Let a and c be the lengths of the sides $p'p''$ and $v'v''$, respectively. Let h_a and h_c be the heights of triangles T and $\triangle vv'v''$, respectively. See Figure 5. Then

$$A(T')+A(T'') = A(T)-A(\triangle vv'v'')-A(\triangle p_\ell p'p'') = \frac{ah_a}{2}-\frac{ch_c}{2}-\frac{a(h_a-h_c)}{2} = \frac{(a-c)h_c}{2}.$$

Since triangles T and $\triangle vv'v''$ are similar, $\frac{h_c}{h_a} = \frac{c}{a}$. Thus, $\frac{A(T')+A(T'')}{A(T)} = \frac{(a-c)h_c}{2} \cdot \frac{2}{ah_a} = (1-\frac{c}{a})\frac{h_c}{h_a} = (1-\frac{c}{a})\frac{c}{a} \leq \frac{1}{4}$, as claimed. The last inequality holds since $(1-x)x$ is maximized when $x = 1/2$. ◀

By Claim 3.13, $\sum_{T\in\mathcal{T}_i} A(T) \leq \frac{1}{4}\sum_{T'\in\mathcal{T}_{i-1}} A(T')$ for all $i \in [m]$. The total area of all triangles in T_0 is at most 1. Thus, the total area of all triangles in $\mathcal{T}_m$ is at most $\frac{1}{4^m} = \frac{\epsilon}{2}$, completing the proof of Lemma 3.12. ◀

Next, we find an upper bound on $|L|$. Recall that $m = \log(2/\epsilon)/2$. The set L consists of the lines that define the sides of the bounding rectangle R and one line for each triangle in $\mathcal{T}_i$ for all $i \in \{0, 1, \ldots, m-1\}$. Therefore, $|L| = 4 + \sum_{i=0}^{m-1} |\mathcal{T}_i| \leq 4 + \sum_{i=0}^{m-1} 4 \cdot 2^i = 4 \cdot 2^m \leq \frac{5.7}{\sqrt{\epsilon}}$.

▶ **Lemma 3.14.** *Let V be the set of vertices of the polygon formed by all lines in L. Then the central point u is in Hull(V).*

▶ **Lemma 3.15.** *The white area of Hull(P) is at least 0.14ϵ.*

Proof. There are at most $5.7 \cdot \epsilon^{-1/2}$ lines in L. Each of them sweeps a black area $0.1 \cdot \epsilon^{3/2}$. Thus, $A(\cup_{\ell\in L} S_\ell) \leq 0.1 \cdot \epsilon^{3/2} \cdot 5.7 \cdot \epsilon^{-1/2} \leq 0.57\epsilon$. By Lemma 3.12, the area of all triangles in $\mathcal{T}_m$ is at most 0.5ϵ. By (1), the violet area is at most 0.04ϵ. We obtain a convex figure if we recolor all black and violet points outside of Hull(V), all white points inside Hull(P), and color each triangle in $\mathcal{T}_m$ according to the majority of its area. This recolors area at most $(0.57 + 0.04 + 0.5/2) \cdot \epsilon = 0.86\epsilon$ outside of Hull(P). Since F is ϵ-far from convexity, the white area of Hull(P) is at least 0.14ϵ. ◀

Now we show that with probability at least 0.68, the tester samples a witness triple. For each line $\ell \in L^*$, let p_ℓ denote the anchor point of S_ℓ on ℓ. Denote the two sets, into which the ray $r(u, p_\ell)$ divides S_ℓ, by S_ℓ^1 and S_ℓ^2 (points of S_ℓ^1 come first in the clockwise order). Let p'_ℓ and p''_ℓ denote the anchor points on ℓ of the sets S_ℓ^1 and S_ℓ^2, respectively. Let the ray $r(u, p'_\ell)$ (resp., $r(u, p''_\ell)$) divide the set S_ℓ^1 (resp., S_ℓ^2) into sets S_ℓ^{11} and S_ℓ^{12} (resp., S_ℓ^{21} and S_ℓ^{22}) such that S_ℓ^{11} (resp., S_ℓ^{21}) is the leftmost subset of S_ℓ^1 (resp., S_ℓ^2).

Recall what patterns are from Definition 3.6. For each $\ell \in L^*$, the rays $r(u, p'_\ell), r(u, p''_\ell)$ and the line ℓ, together with sets $B_1 = S_\ell^{11}$ and $B_2 = S_\ell^{22}$, form a pattern. We say that such a pattern is of *type 1*. For every two adjacent lines ℓ and ℓ' from L, the rays $r(u, p'_\ell), r(u, p'_{\ell'})$ and the line $\ell(p_\ell, p_{\ell'})$, together with sets $B_1 = S_\ell^{21}$ and $B_2 = S_{\ell'}^{12}$, also form a pattern. We say that such a pattern is of *type 2*. Patterns of types 1 and 2 alternate. See Figures 6-8.

By Lemma 3.14, the point u is inside Hull(V). Note that u can be inside Hull(P) or in one of the triangles of $\mathcal{T}_m$. If the former is the case, then every type 1 and type 2 pattern is well defined, their W regions entirely cover Hull(P) and are disjoint (see Figure 6). If the latter is the case, consider the triangle of $\mathcal{T}_m$ in which u is located. Consider the sweeping lines that define the sides of this triangle. Type 2 pattern with respect to these lines may not be well defined but all other type1 and type 2 patterns are well defined. Moreover, their W regions are disjoint and they entirely cover Hull(P). These two properties of the patterns are the only properties that we need in the further analysis.

Index all patterns of types 1 and 2 by natural numbers $1, 2, \ldots$ All of them are white-light patterns, since there are no white-heavy patterns left after the recoloring phase. For a pattern i, let E_W^i denote the event that a white point is sampled from its W region, let a_i be the white area of W, and let E_B^i denote the event that a point is sampled both from its B_1 and from its B_2. By Claim 3.7, $\Pr[E_B^i] \geq 0.39 \cdot \epsilon^{1/3}$. Moreover, $\Pr[E_W^i] = 1 - e^{-a_i s}$. Recall that, by Definition 3.6, $a_i < 0.025\epsilon^{4/3}$ and, thus, $a_i s < 1.5$. We use the fact that $1 - e^{-x} \geq 0.5x$ for all $x \in (0, 1.5)$. We obtain that $\Pr[E_W^i] \geq 0.5 a_i s$. Therefore, the probability that we sample a witness triple from pattern i is $\Pr[E_B^i \cap E_W^i] \geq 0.39 \cdot \epsilon^{1/3} \cdot 0.5 a_i s = 9.75 a_i/\epsilon$. By Lemma 3.15, $\sum a_i \geq 0.14\epsilon$. By standard arguments, the probability that a witness triple is sampled from at least one pattern is $\Pr[\bigcup(E_B^i \cap E_W^i)] \geq 1 - e^{-\sum_i 9.75 a_i/\epsilon} \geq 1 - e^{-0.14 \cdot 9.75} > 0.68$, as desired. This completes the proof of Proposition 3.9. ◀

4 Lower Bound for Uniform Testing of Convexity

▶ **Theorem 4.1.** *Every 1-sided error uniform ϵ-tester for convexity needs $\Omega(\epsilon^{-4/3})$ samples.*

Proof. By the Poissonization Lemma (Lemma 2.2), it is sufficient to prove the lower bound for Poisson algorithms. Observe that a 1-sided error tester can reject only if the samples it obtained are not consistent with any convex figure. For each sufficiently small ϵ, we will construct a set C_ϵ in $U = [0, 1]^2$ that is ϵ-far from convex. We will show that there exists a constant c_0 such that every Poisson-s tester with $s = c_0 \cdot \epsilon^{-4/3}$ fails to detect a violation of convexity with probability at least $1/2$, for every constructed set C_ϵ.

First, we construct the hard sets C_ϵ. Let $k = \lceil \frac{1}{5} \cdot \epsilon^{-1/2} \rceil$. Let G be a convex regular $2k$-gon inside $[0, 1]^2$ with the side length $\frac{1}{2}\sin(\frac{\pi}{2k})$. Number all vertices of G from 1 to $2k$ in the clockwise order (see Figure 9). Let G' and G'' be the two regular k-gons obtained by connecting the vertices with odd and even numbers, respectively. Let C_ϵ be the set of points in $G' \cup G''$. That is, all points in $G' \cup G''$ are black on the figure, and all remaining points in $[0, 1]^2$ are white.

▶ **Lemma 4.2.** *The figure (U, C_ϵ) is ϵ-far from convexity for all sufficiently small ϵ.*

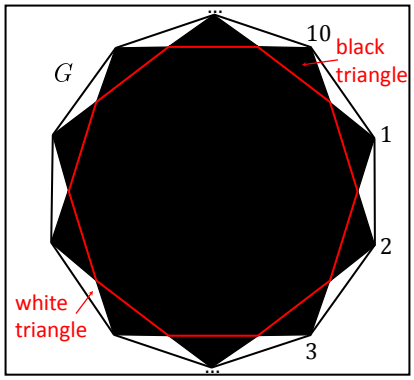


Figure 9 An illustration of G for $k = 5$.

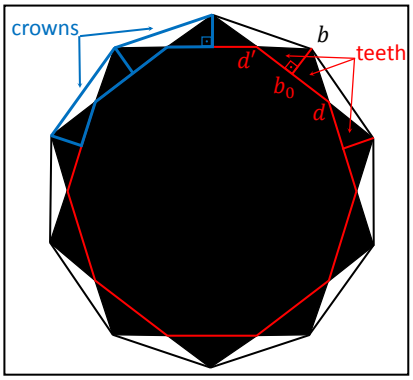


Figure 10 Teeth and crowns.

Proof. The region $G\backslash(G' \cup G'')$ consists of triangles in which all points are white. Call any such triangle *white*. The symmetric difference of G' and G'' consists of triangles in which all points are black. Call any such triangle *black*. Let T be a black triangle and b be its vertex such that it is also a vertex of G (see Figure 10). Let d and d' denote the other two vertices of T. Let b_0 be the point on the side dd' such that bb_0 is the height of T. Call triangles $\triangle bb_0d$ and $\triangle bb_0d'$ *teeth*. A *crown* consists of two teeth that intersect in exactly one point and the white triangle between them. The the following claim is proved in the full version.

▶ **Claim 4.3.** Let A_T and A_W be the areas of a tooth and a white triangle, respectively. Then, for sufficiently large k, we have $1/(5k^3) \leq A_T \leq A_W \leq 1/k^3$. Moreover, area at least $\frac{A_T}{8}$ of each of the $2k$ disjoint crowns must be changed to make C_ϵ convex.

There are $2k$ disjoint crowns. Recall that $k = \lceil \frac{1}{5} \cdot \epsilon^{-1/2} \rceil$. Thus, by Claim 4.3, to make C_ϵ convex, area at least $\frac{A_T}{8} \cdot 2k \geq 1/(20k^2) \geq \epsilon$ needs to be modified. That is, the figure (U, C_ϵ) is ϵ-far from convexity. ◀

Now consider how an algorithm can detect a violation of convexity in the hard figures we constructed. First, it is sufficient to change all the points in the white triangles to make such a figure convex. Therefore, a violation can be detected only if a point from a white triangle is in the sample. For any white triangle, it is sufficient to change the points in one of the two black triangles adjacent to it to ensure that the points from the white triangle are not in the convex hull of black points. Therefore, it is necessary to sample a point from both adjacent black triangles. Thus, the probability of detecting a violation of convexity is bounded from above by the probability of detecting a *red-flag triple*, defined next.

▶ **Definition 4.4** (A red-flag triple)**.** A triple of points (w, b_1, b_2) is a *red-flag triple* if w belongs to a white triangles and b_1 and b_2 belong to two different adjacent black triangles.

▶ **Lemma 4.5.** *Let c_0 be an appropriate constant. For all sufficiently small ϵ, a Poisson-s algorithm with $s = c_0 \cdot \epsilon^{-4/3}$ detects a red-flag triple in the figure (U, C_ϵ) with probability at most 1/2.*

Proof. We define the following random variables for the Poisson-s algorithm: Y counts the total number of sampled red-flag triples, Y_W counts the number of sampled red-flag triples that involve a point w from a white triangle W, variable X_W counts the number of samples in a white triangle W, and X_{B_1} and X_{B_2} count the number of samples in the two black triangles adjacent to W, respectively. To prove the lemma, it is sufficient to show that $\Pr[Y \geq 1] \leq 1/2$.

By the Poissonization Lemma (Lemma 2.2), X_W is a Poisson random variable with expectation $A_W \cdot s$, where A_W is the area of a white triangle. Similarly, $\mathbb{E}[X_{B_1}] = \mathbb{E}[X_{B_2}] = 2A_T \cdot s$, where A_T is the area of a tooth and hence half the area of a black triangle. The random variables X_W, X_{B_1}, X_{B_2} are independent because they are sample counts for disjoint areas. Since $Y_W = X_W \cdot X_{B_1} \cdot X_{B_2}$, we get that $\mathbb{E}[Y_W] = \mathbb{E}[X_W] \cdot \mathbb{E}[X_{B_1}] \cdot \mathbb{E}[X_{B_2}] = 4A_W \cdot (A_T)^2 \cdot s^3 \leq 4(A_W)^3 \cdot s^3 \leq \frac{4}{k^9} \cdot s^3$. The inequalities above use Claim 4.3 and hold for sufficiently large k (i.e., sufficiently small ϵ).

Since there are $2k$ crowns, with identical distributions of samples inside them,

$$\mathbb{E}[Y] = 2k \cdot \mathbb{E}[Y_W] \leq 8\frac{1}{k^8} \cdot s^3 \leq 8 \cdot 5^8 \epsilon^4 \cdot c_0^3 \epsilon^{-4} \leq 1/2,$$

assuming c_0 is sufficiently small. By Markov's inequality, the probability of detecting a red-flag is at most $\Pr[Y \geq 1] \leq \mathbb{E}[Y] \leq 1/2$. ◀

Theorem 4.1 follows from Lemmas 4.2 and 4.5. Thus, 1-sided error uniform ϵ-tester for convexity needs $s = \Omega(\epsilon^{-4/3})$ samples. ◀

5 Conclusion

We showed that in 2 dimensions, testing convexity of figures with uniform samples can be done faster than learning convex figures under the uniform distribution. It is an interesting open question whether this is also true in higher dimensions. We showed that the running time of our tester cannot be improved if 1-sided error is required. The question is open for 2-sided error testers.

In subsequent work [4], we designed an adaptive, $O(1/\epsilon)$ time tester for testing convexity of visual images in the pixel model. The tester and its analysis, with small modifications, also apply to testing convexity of figures.

References

1 A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Inf. Process. Lett.*, 9(5):216–219, 1979. doi:10.1016/0020-0190(79)90072-3.

2 Imre Barany. Extremal problems for convex lattice polytopes: a survey. *Contemporary Mathematics*, 2000.

3 Imre Bárány and Zoltán Füredi. On the shape of the convex hull of random points. *Probability theory and related fields*, 77(2):231–240, 1988.

4 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The role of adaptivity in image testing. Manuscript, 2015.

5 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *STOC*, pages 164–173, 2014. doi:10.1145/2591796.2591887.

6 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, 2014*, pages 309–320, 2014.

7 Bernard Chazelle and C. Seshadhri. Online geometric reconstruction. *J. ACM*, 58(4):14, 2011. doi:10.1145/1989727.1989728.

8 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *Algorithms – ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 266–277, 2001. doi:10.1007/3-540-44676-1_22.

9 Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In *Algorithms – ESA 2000, 8th Annual European Symposium, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, pages 155–166, 2000. doi:10.1007/3-540-45253-2_15.

10 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1987. doi:10.1007/978-3-642-61568-9.

11 Ronen Eldan. A polynomial number of random points does not determine the volume of a convex body. *Discrete and Computational Geometry*, 46(1):29–47, 2011. doi:10.1007/s00454-011-9328-x.

12 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

13 Sariel Har-Peled. On the expected complexity of random convex hulls. *CoRR*, abs/1111.5340, 2011. URL: http://arxiv.org/abs/1111.5340.

14 Igor Kleiner, Daniel Keren, Ilan Newman, and Oren Ben-Zwi. Applying property testing to an image partitioning problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):256–265, 2011. doi:10.1109/TPAMI.2010.165.

15 Simon Korman, Daniel Reichman, and Gilad Tsur. Tight approximation of image matching. *CoRR*, abs/1111.1713, 2011.

16 Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. Fast-match: Fast affine template matching. In *CVPR*, pages 2331–2338, 2013. doi:10.1109/CVPR.2013.302.

17 Nimrod Megiddo. Partitioning with two lines in the plane. *J. Algorithms*, 6(3):430–433, 1985. doi:10.1016/0196-6774(85)90011-2.

18 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5):1158–1184, 2003. URL: http://epubs.siam.org/sam-bin/dbq/article/41402.

19 Luis Rademacher and Santosh Vempala. Testing geometric convexity. In *FSTTCS*, pages 469–480, 2004.

20 Sofya Raskhodnikova. Approximate testing of visual properties. In *RANDOM-APPROX*, pages 370–381, 2003. doi:10.1007/978-3-540-45198-3_31.

21 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM J. Comput.*, 39(3):813–842, 2009.

22 Sofya Raskhodnikova and Grigory Yaroslavtsev. Learning pseudo-Boolean k-DNF and submodular functions. In *SODA*, pages 1356–1368, 2013.

23 H Raynaud. Sur l'enveloppe convexe des nuages de points aleatoires dans r n. i. *Journal of Applied Probability*, 7(1):35–48, 1970.

24 Alfréd Rényi and Rolf Sulanke. Über die konvexe hülle von n zufällig gewählten punkten. *Probability Theory and Related Fields*, 2(1):75–84, 1963.

25 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

26 Bernd Schmeltz. Learning convex sets under uniform distribution. In *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, pages 204–213, 1992.

27 C. Seshadhri and Jan Vondrák. Is submodularity testable? *Algorithmica*, 69(1):1–25, 2014. doi:10.1007/s00453-012-9719-2.

28 Wojciech Szpankowski. *Average Case Analysis of Algorithms on Sequences*. John Wiley & Sons, Inc., New York, 2001.

29 Gilad Tsur and Dana Ron. Testing properties of sparse images. In *FOCS*, pages 468–477, 2010. doi:10.1109/FOCS.2010.52.

30 John W Tukey. Mathematics and the picturing of data. In *Proceedings of the international congress of mathematicians*, volume 2, pages 523–531, 1975.

Separating a Voronoi Diagram via Local Search

Vijay V. S. P. Bhattiprolu[1] **and Sariel Har-Peled**[*2]

1 **School of Computer Science, Carnegie Mellon University, Pittsburgh, USA**
vpb@cs.cmu.edu
2 **Department of Computer Science, University of Illinois, Urbana, USA**
sariel@illinois.edu

Abstract

Given a set P of n points in $\mathbb{R}^d$, we show how to insert a set Z of $O\left(n^{1-1/d}\right)$ additional points, such that P can be broken into two sets P_1 and P_2, of roughly equal size, such that in the Voronoi diagram $\mathcal{V}(P \cup Z)$, the cells of P_1 do not touch the cells of P_2; that is, Z separates P_1 from P_2 in the Voronoi diagram (and also in the dual Delaunay triangulation). In addition, given such a partition (P_1, P_2) of P, we present an approximation algorithm to compute a minimum size separator realizing this partition. We also present a simple local search algorithm that is a PTAS for approximating the optimal Voronoi partition.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.1.2 Algorithms, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Separators, Local search, Approximation, Voronoi diagrams, Delaunay triangulation, Meshing, Geometric hitting set

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.18

1 Introduction

Many algorithms work by partitioning the input into a small number of pieces, of roughly equal size, with little interaction between the different pieces, and then recurse on these pieces. One natural way to compute such partitions for graphs is via the usage of separators. A (vertex) separator of a graph $G = (V, E)$, informally, is a "small" set $Z \subseteq V$ whose removal breaks the graph into two (or more) subgraphs, each of which is of size at most n/c, where c is some constant strictly larger than one. As a concrete example, any tree with n vertices has a single vertex, which can be computed in linear time, such that its removal breaks the tree into subtrees, each with at most $n/2$ vertices.

Separators in planar graphs. Ungar [34] showed that a planar graph with n vertices contains a separator of size $O\left(\sqrt{n \log n}\right)$. This was later improved by Lipton and Tarjan [24] to $O(\sqrt{n})$, and they also provided an algorithm to compute the separator in linear time. Specifically, there exists a separator of size $O(\sqrt{n})$ such that its removal partitions the graph into two disjoint subgraphs each containing at most $2n/3$ vertices (each of these subgraphs is not necessarily connected).

There has been a substantial amount of work on planar separators in the last four decades, and they are widely used in data-structures and algorithms for planar graphs, including (i) shortest paths [15], (ii) distance oracles [33], (iii) max flow [13], and (iv) approximation

* Work on this paper by the second author was partially supported by NSF AF awards CCF-0915984, CCF-1421231, and CCF-1217462.


© Vijay V. S. P. Bhattiprolu and Sariel Har-Peled;
licensed under Creative Commons License CC-BY

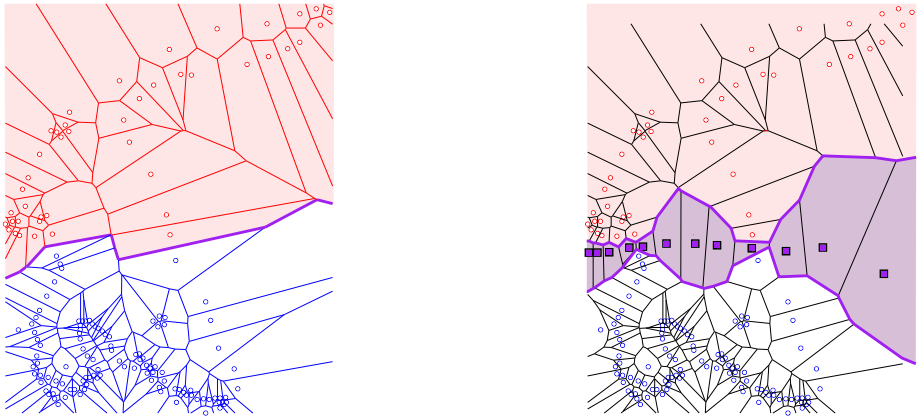


Figure 1 On the left a Voronoi partition, and on the right, a separator realizing it.

algorithms for TSP [22]. This list is a far cry from being exhaustive, and is a somewhat arbitrary selection of some recent results on the topic.

Geometric separators. Any planar graph can be realized as a set of interior disjoint disks, where a pair of disks touch each other, if and only if the corresponding vertices have an edge between them. This is the *circle packing theorem* [31], also known as the Koebe-Andreev-Thurston theorem [23].

Surprisingly, the existence of a planar separator is an easy consequence of the circle packing theorem. This was proved by Miller *et al.* [26] (see also [19]). Among other things, Miller *et al.* showed that given a set of n balls in $\mathbb{R}^d$, such that no point is covered more than k times, the intersection graph of the balls has a separator of size $O\left(k^{1/d}n^{1-1/d}\right)$. This implies that the k-nearest neighbor graph of a set of points in $\mathbb{R}^d$, has a small separator [26, 19]. Various extensions of this technique were described by Smith and Wormald [32].

Other separators. Small separators are known to exist for many other families of graphs. These include graphs (i) with bounded tree width [8], (ii) with bounded genus [16], (iii) that are minor free [1], and (iv) grids. Furthermore, graphs with hereditary sublinear separators have *polynomial expansion* [11], and vice versa – graphs with polynomial expansion have sublinear separators [29].

Voronoi separators. In this paper, we are interested in geometric separation in a Voronoi diagram [3]. Specifically, given a set P of points in $\mathbb{R}^d$, we are interested in inserting a small set of new points Z, such that there is a balanced partition of P into two sets P_1, P_2, such that no cell of P_1 touches a cell of P_2 in the Voronoi diagram $\mathcal{V}(P \cup Z)$. Note, that such a set Z also separates P_1 and P_2 in the Delaunay triangulation of $P \cup Z$.

Using Voronoi separators. One of the motivations of Lipton and Tarjan [24, 25] was implementing divide and conquer algorithms on graphs. For example, *generalized nested dissection* – solving a system of linear equations rising out of numerical simulations done over planar meshes. Thus, Voronoi separators (which in the dual are *Delaunay separators*) provide a way to breakup Delaunay meshes. Unlike their planar graph counterpart, Voronoi separators also exist in higher dimensions.

Specifically, some meshing algorithms rely on computing a Delaunay triangulation of geometric models to get good triangulations that describe solid bodies. Such meshes in turn are fed into numerical solvers to simulate various physical processes. To get good triangulations, one performs a Delaunay refinement that involves inserting new points into the triangulations, to guarantee that the resulting elements are well behaved. Since the underlying geometric models can be quite complicated and these refinement processes can be computationally intensive, it is natural to try and break up the data in a balanced way, and Voronoi separators provide one way to do so. In particular, small Voronoi separators provide a way to break up a point set in such a way that there is limited interaction between two pieces of the data.

Geometric hitting set. Given a set of objects in $\mathbb{R}^d$, the problem of finding a small number of points that stab all the objects is an instance of geometric hitting set. There is quite a bit of research on this problem. In particular, the problem is NP-Hard for almost any natural instance, but a polynomial time $(1+\varepsilon)$-approximation algorithm is known for the case of balls in $\mathbb{R}^d$ [9], where one is allowed to place the stabbing points anywhere. The discrete variant of this problem, where there is a set of allowable locations to place the stabbing points, seems to be significantly harder and only weaker results are known [20]. See Mustafa *et al.* [27] for a QPTAS for the case of disks and points, and Har-Peled and Quanrud [21] for PTAS for shallow fat objects and matching hardness results.

One of the more interesting versions of the geometric hitting set problem, is the art gallery problem, where one is given a simple polygon in the plane, and one has to select a set of points (inside or on the boundary of the polygon) that "see" the whole polygon. While much research has gone into variants of this problem [30], nothing is known as far as an approximation algorithm (for the general problem). The difficulty arises from the underlying set system being infinite, see [12] for some efforts in better understanding this problem.

Geometric local search. Relatively little is known regarding local search methods for geometric approximation problems. Arya *et al.* [2] gave a local search method for approximating k-median clustering by a constant factor, and this was recently simplified by Gupta and Tangwongsan [17]. Mustafa and Ray [28] gave a local search algorithm for the discrete hitting set problem over pseudo disks and r-admissible regions in the plane, which yields a PTAS. Chan and Har-Peled [10] gave a local search PTAS for the independent set problem over fat objects, and for pseudodisks in the plane. Both works use separators in proving the quality of approximation.

More recently, Bandyapadhyay and Varadarajan [4] have employed some of the results in this paper to give a bi-criteria local search PTAS for variants of the k-means problem.

1.1 Our results

In this paper we give algorithms for the following:

(A) Computing a small Voronoi separator. Given a set P of n points in $\mathbb{R}^d$, we show how to compute, in expected linear time, a balanced Voronoi separator of size $O\left(n^{1-1/d}\right)$. This is described in Section 3. The existence of such a separator was not known before, and our proof is relatively simple and elegant.

(B) Exact algorithm for computing the smallest Voronoi separator realizing a given partition. In Section 4, given a partition (P_1, P_2) of a point set P in $\mathbb{R}^d$, we describe an algorithm that computes a minimum size Voronoi separator realizing this separation. The running

time of the algorithm is $n^{O(d^2\mathsf{b})}$, where b is the cardinality of the optimal separating sets.

(C) Constant approximation algorithm for the smallest Voronoi separator realizing a given partition. In Section 5, we describe how to compute a constant factor approximation to the size of an optimal Voronoi separator for a given partition of a set in $\mathbb{R}^d$. This is the natural extension of the greedy algorithm for geometric hitting set of balls, except that in this case, the set of balls is infinite and is encoded implicitly, which somewhat complicates things.

(D) A PTAS for the smallest Voronoi separator realizing a given partition. We present a polynomial time approximation scheme to compute a Voronoi separator, realizing a given partition, whose size is a $(1+\varepsilon)$-approximation to the size of an optimal Voronoi separator for a given partition of a set in $\mathbb{R}^d$. The running time is $n^{O(1/\varepsilon^d)}$. See the full version of the paper for the full details [7], which are omitted in this version.
Interestingly, the new algorithm provides a PTAS for the geometric hitting set problem (for balls), that unlike previous approaches that worked top-down [9, 14], works more in a bottom-up approach. Note, that since our set of balls that needs to be pierced is infinite, and is defined implicitly, it is not obvious a priori how to use the previous algorithms in this case.

Sketch of algorithm. The new algorithm works by first computing a "dirty" constant approximation hitting set using a greedy approach (this is relatively standard). Somewhat oversimplifying, the algorithm next clusters this large hitting set into tight clusters of size $k = O(1/\varepsilon^d)$ each. It then replaces each such cluster of the weak hitting set, by the optimal hitting set that can pierce the same set of balls, computed by using the exact algorithm – which is "fast" since the number of piercing points is at most $O(1/\varepsilon^d)$. In the end of this process the resulting set of points is the desired hitting set. Namely, the new approximation algorithm reduces the given geometric hitting set instance, into $O(m/k)$ smaller instances where m is the size of the overall optimal hitting set and each of the smaller instances has an optimal hitting set of size $O(k)$.
For the analysis of this algorithm, we need a strengthened version of the separator theorem. See Theorem 23 for details.

(E) Local search PTAS for Voronoi partition and continuous geometric hitting set problems. An interesting consequence of the new bottom-up PTAS, is that it leads to a simple local search algorithm for geometric hitting set problems for fat objects. Specifically, in Section 7, we show that the algorithm starts with any hitting set (of the given objects) and continues to make local improvements via exchanges of size at most $O\big(1/\varepsilon^d\big)$, until no such improvement is possible, yielding a PTAS. The analysis of the local search algorithm is subtle and involves simultaneously clustering the locally optimal solution, and the optimal solution, and matching these clusters to each other.

Relation to known results. The separator result is similar in spirit (but not in details) to the work of Miller *et al.* [26] and Chan [9] on a separator for a k-ply set of balls – the main difference being that Voronoi cells behave differently than balls do. The bottom-up PTAS approach seems to be new, and should be applicable to other problems. Having said that, it seems like the top-down approaches [9, 14] potentially can be modified to work in this case, but the low level details seem to be significantly more complicated, and the difficulty in making them work was the main motivation for developing the new approach. The basic idea of using separators in analyzing local search algorithms appear in the work of Mustafa and Ray [28] and Chan and Har-Peled [10].

2 Preliminaries

For a point set $P \subseteq \mathbb{R}^d$, the *Voronoi diagram* of P, denoted by $\mathcal{V}(P)$ is the partition of space into convex cells, where the *Voronoi cell* of $\mathsf{p} \in P$ is

$$\mathcal{C}_P(\mathsf{p}) = \left\{ \mathsf{q} \in \mathbb{R}^d \;\middle|\; \|\mathsf{q} - \mathsf{p}\| \leq \mathsf{d}(\mathsf{q}, P) \right\},$$

where $\mathsf{d}(\mathsf{q}, P) = \min_{\mathsf{s} \in P} \|\mathsf{q} - \mathsf{s}\|$ is the distance of q to the set P. Voronoi diagrams are a staple topic in Computational Geometry, see [6], and we include the definitions here for the sake of completeness.

In the plane, the Voronoi diagram has linear descriptive complexity. Here, the complexity refers to the length of encoding a set as a semialgebraic set [5]. Specifically, the diagram can be broken into linear number of cells (i.e., triangles in this case), where each cell can be described by a constant number of algebraic inequalities.

For a point set P, and points $\mathsf{p}, \mathsf{q} \in P$, the geometric loci of all points in $\mathbb{R}^d$ that have both p and q as nearest neighbor, is the *bisector* of p and q – it is denoted by $\beta_{\mathsf{p},\mathsf{q}} = \left\{ \mathsf{s} \in \mathbb{R}^d \;\middle|\; \|\mathsf{s} - \mathsf{p}\| = \|\mathsf{s} - \mathsf{q}\| = \mathsf{d}(\mathsf{s}, P) \right\}$. A point $\mathsf{s} \in \beta_{\mathsf{p},\mathsf{q}}$ is the center of a ball whose interior does not contain any point of P and that has p and q on its boundary. The set of all such balls induced by $\beta_{\mathsf{p},\mathsf{q}}$ is the *pencil* of p and q, denoted by $\mathrm{pencil}(\mathsf{p}, \mathsf{q})$.

▶ **Definition 1.** Let P be a set of points in $\mathbb{R}^d$, and P_1 and P_2 be two disjoint subsets of P. The sets P_1 and P_2 are *Voronoi separated* in P if for all $\mathsf{p}_1 \in P_1$ and $\mathsf{p}_2 \in P_2$, we have that their Voronoi cells are disjoint; that is, $\mathcal{C}_P(\mathsf{p}_1) \cap \mathcal{C}_P(\mathsf{p}_2) = \emptyset$.

▶ **Definition 2.** For a set P, a *partition* of P is a pair of sets (P_1, P_2), such that $P_1 \subseteq P$, and $P_2 = P \setminus P_1$. A set Z is a *Voronoi separator* for a partition (P_1, P_2) of $P \subseteq \mathbb{R}^d$, if P_1 and P_2 are Voronoi separated in $P \cup Z$; that is, the Voronoi cells of P_1 in $\mathcal{V}(P \cup Z)$ do not intersect the Voronoi cells of P_2. We will refer to the points of the separator Z as *guards*.

See Figure 1 for an example of the above definitions.

▶ **Definition 3.** For a ball b, its *cover number* is the minimum number of (closed) balls of half the radius that are needed to cover it. The *doubling constant* of a metric space is the maximum cover number over all possible balls. Let c_{dbl}^d be the doubling constant for $\mathbb{R}^d$.

The constant c_{dbl}^d is exponential in d, and $c_{\mathrm{dbl}}^d \leq \left\lceil 2\sqrt{d} \right\rceil^d$ – indeed, cover a ball (say, of unit radius) by a grid with sidelength $1/\sqrt{d}$, and observe that each grid cell has diameter 1, and as such can be covered by a ball of radius $1/2$.

▶ **Definition 4.** For a closed set $\mathsf{X} \subseteq \mathbb{R}^d$, and a point $\mathsf{p} \in \mathbb{R}^d$, the *projection* of p into X is the closest point in X to p. We denote the projected point by $\mathsf{nn}(\mathsf{p}, \mathsf{X})$.

3 Computing a small Voronoi separator

3.1 Preliminaries and how to guard a sphere

Given a set P of n points in $\mathbb{R}^d$, we show how to compute a balanced Voronoi separator for P of size $O\left(n^{1-1/d}\right)$.

▶ **Definition 5.** A set $\mathsf{Y} \subseteq \mathsf{X} \subseteq \mathbb{R}^d$ is *ℓ-dense* in X, if for any point $\mathsf{p} \in \mathsf{X}$, there exists a point $\mathsf{s} \in \mathsf{Y}$, such that $\|\mathsf{p} - \mathsf{s}\| \leq \ell$.

▶ **Lemma 6** (proof in [7]). *Consider an arbitrary sphere $\mathbb{S}$, and a point $\mathsf{p} \in \mathbb{R}^d \setminus \mathbb{S}$. Then one can compute, in constant time, a set of points $Q \subseteq \mathbb{S}$, such that the Voronoi cell $\mathcal{C}_{Q \cup \{\mathsf{p}\}}(\mathsf{p})$ does not intersect $\mathbb{S}$, and $|Q| = O(1)$. We denote the set Q by* $\mathrm{blockerSet}(\mathsf{p}, \mathbb{S})$.

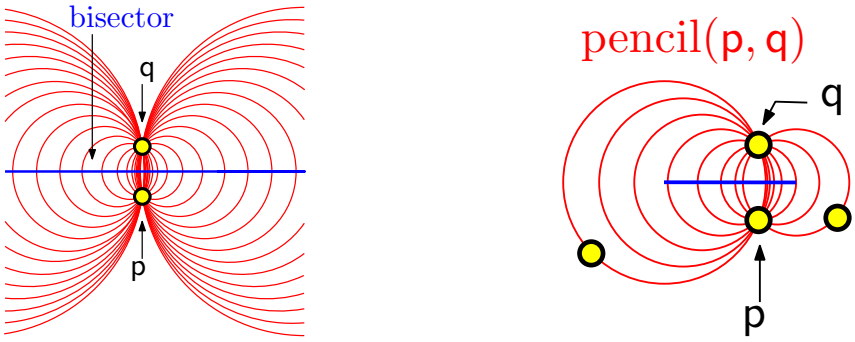


Figure 2 (a) The unbounded bisector induced by p and q. (b) The pencil of p and q.

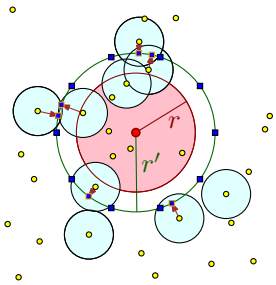


Figure 3 A slightly inaccurate depiction of how the algorithm works.

3.2 The algorithm computing the Voronoi separator

The input is a set P of n points in $\mathbb{R}^d$. The algorithms works as follows:

(A) Let $c_d = c^d_{\mathrm{dbl}} + 1$, see Definition 3. Let $\mathrm{ball}(\psi, r)$ be the smallest (closed) ball that contains n/c_d points of P where $\psi \in \mathbb{R}^d$.

(B) Pick a number r' uniformly at random from the range $[r, 2r]$.

(C) Let $b' = \mathrm{ball}(\psi, r')$.

(D) Let $P_1 = P \cap b'$ and $P_2 = P \setminus b'$.

(E) Let $\ell = r'/n^{1/d}$. Compute an ℓ-dense set Z, of size $O\Big((r'/\ell)^{d-1}\Big) = O\big(n^{1-1/d}\big)$, on the sphere $\mathbb{S} = \partial b'$ using the algorithm of Lemma 7 described below.

(F) If a point $\mathsf{p} \in P$ is in distance smaller than ℓ from $\mathbb{S}$, we insert $\mathrm{blockerSet}(\mathsf{p}, \mathbb{S})$ into the separating set Z, see Lemma 6.

We claim that the resulting set Z is the desired separator.

Efficient implementation. One can find a 2-approximation (in the radius) to the smallest ball containing n/c_d points in linear time, see [18]. This would slightly deteriorate the constants used above, but we ignore this minor technicality for the sake of simplicity of exposition. If the resulting separator is too large (i.e., larger than $\Omega\big(n^{1-1/d}\big)$ see below for details), we rerun the algorithm.

3.2.1 Computing a dense set

The following is well known, and we include it only for the sake of completeness.

▶ **Lemma 7** (proof in [7]). *Given a sphere $\mathbb{S}$ of radius r' in $\mathbb{R}^d$, and given a number $\ell > 0$, one can compute a ℓ-dense set X on $\mathbb{S}$ of size $O\left((r'/\ell)^{d-1}\right)$. This set can be computed in $O(|\mathsf{X}|)$ time.*

3.3 Correctness

▶ **Lemma 8** (proof in [7]). *We have $|P_1| \geq n/c_d$ and $|P_2| \geq n/c_d$.*

▶ **Lemma 9** (proof in [7]). *The sets P_1 and P_2 are Voronoi separated in $\mathcal{V}(P \cup Z)$.*

▶ **Lemma 10** (proof in [7]). *Let $Y = |Z|$. We have that $\mathbf{E}[Y] \leq c_{\mathrm{sep}} n^{1-1/d}$, where c_{sep} is some constant.*

3.4 The result

▶ **Theorem 11.** *Let P be a set of n points in $\mathbb{R}^d$. One can compute, in expected linear time, a sphere $\mathbb{S}$, and a set $Z \subseteq \mathbb{S}$, such that*

(i) $|Z| = O\left(n^{1-1/d}\right)$,
(ii) *$\mathbb{S}$ contains $\geq cn$ points of P inside it,*
(iii) *there are $\geq cn$ points of P outside $\mathbb{S}$, and*
(iv) *Z is a Voronoi separator of the points of P inside $\mathbb{S}$ from the points of P outside $\mathbb{S}$.*

Here $c > 0$ is a constant that depends only on the dimension d.

Proof. Clearly, each round of the algorithm takes $O(n)$ time. By Markov's inequality the resulting separator set Z is of size at most $2c_{\mathrm{sep}} n^{1-1/d}$, with probability at least $1/2$, see Lemma 10. As such, if the separator is larger than this threshold, then we rerun the algorithm. Clearly, in expectation, after a constant number of iterations the algorithm would succeed, and terminates. (It is not hard to derandomize this algorithm and get a linear running time by defining an appropriate number of concentric balls around $\mathrm{ball}(\psi, r)$ and using an averaging argument.) ◀

4 Exact algorithm for computing optimal separation of a partition

Given a set P of n points in $\mathbb{R}^d$, and a partition (P_1, P_2) of P, we are interested in computing the smallest Voronoi separating set realizing this partition.

4.1 Preliminaries and problem statement

▶ **Definition 12.** For a set $P \in \mathbb{R}^d$ and a pair of disjoint subsets (P_1, P_2), the set of *bad pairs* is $\mathcal{BP}(P, P_1, P_2) = \left\{ (\mathsf{p}_1, \mathsf{p}_2) \in P_1 \times P_2 \;\middle|\; \mathcal{C}_P(\mathsf{p}_1) \cap \mathcal{C}_P(\mathsf{p}_2) \neq \emptyset \right\}$.

For a Voronoi diagram $\mathcal{V}(P)$, we can assume that all its faces (of various dimensions) are all triangulated (say, using bottom-vertex triangulation). This does not change the asymptotic complexity of the Voronoi diagram. For $k = 0, 1, \ldots, d$, such a k dimensional Voronoi simplex is a *k-feature*. Such a k-feature f, is induced by $d - k + 1$ sites, denoted by $\mathrm{sites}(\mathsf{f})$; that is, any point $\mathsf{p} \in \mathsf{f}$ has an equal distance to all the points of $\mathrm{sites}(\mathsf{f})$ and these are the nearest neighbor of p in P. Thus, a vertex v of the Voronoi diagram is a 0-feature, and $|\mathrm{sites}(v)| = d + 1$ (assuming general position, which we do). The *span* of a feature f, is the set of points in $\mathbb{R}^d$ that are equidistant to every site in $\mathrm{sites}(\mathsf{f})$; it is denoted by $\mathrm{span}(\mathsf{f})$ and is the k flat that contains f (it is the *affine span* of f). A *k-halfflat* is the intersection of a halfspace with a k-flat.

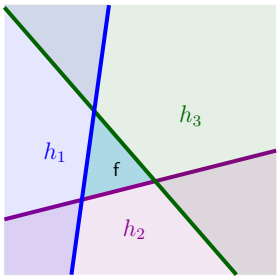


Figure 4 A 2-feature f and its induced 2-halfflats h_1, h_2, h_3.

Consider any k-feature f. The complement set $\text{span}(\mathsf{f}) \setminus \mathsf{f}$ can be covered by $k+1$ k-halfflats of $\text{span}(\mathsf{f})$. Specifically, each of these halfflats is an open k-halfflat of $\text{span}(\mathsf{f})$, whose boundary contains a $(k-1)$-dimensional face of the boundary of f. This set of halfflats of f, is the *shell* of f, and is denoted by $\text{shell}(\mathsf{f})$, see Figure 4.

Once the Voronoi diagram is computed, it is easy to extract the "bad features". Specifically, the set of *bad features* is

$$\mathcal{F}_{\text{bad}}(P, P_1, P_2) = \left\{ \mathsf{f} \in \text{features}(\mathcal{V}(P)) \;\middle|\; \text{sites}(\mathsf{f}) \cap P_1 \neq \emptyset \;\text{ and }\; \text{sites}(\mathsf{f}) \cap P_2 \neq \emptyset \right\}.$$

Clearly, given a Voronoi diagram the set of bad features can be computed in linear time in the size of the diagram.

Given a k-feature f, it is the convex-hull of $k+1$ points; that is, $\mathsf{f} = \mathcal{CH}(\mathsf{X})$, where $\mathsf{X} = \{\mathsf{q}_1, \ldots, \mathsf{q}_{k+1}\}$. We are interested in finding the closest point in a feature to an arbitrary point p. This is a constant size problem for a fixed d, and can be solved in constant time. We denote this closest point by $\mathsf{nn}(\mathsf{p}, \mathsf{f}) = \arg\min_{\mathsf{q} \in \mathsf{f}} \mathsf{d}(\mathsf{p}, \mathsf{q})$. For the feature f, and any point p, we denote by $\text{pencil}_{\mathsf{f}}(\mathsf{p})$ the ball $\text{ball}(\mathsf{p}, \mathsf{d}(\mathsf{p}, \text{sites}(\mathsf{f})))$ (if it is uniquely defined). Furthermore, for an arbitrary set S of points, $\text{pencil}_{\mathsf{f}}(S)$ denote $\left\{ \text{ball}(\mathsf{p}, \mathsf{d}(\mathsf{p}, \text{sites}(\mathsf{f}))) \;\middle|\; \mathsf{p} \in S \right\}$. In particular, for any point $\mathsf{p} \in \mathsf{f}$, consider $\text{ball}(\mathsf{p}, \mathsf{d}(\mathsf{p}, P))$ – it contains the points of $\text{sites}(\mathsf{f})$ on its boundary. The set of all such balls is the *pencil* of f, denoted by

$$\text{pencil}(\mathsf{f}) = \left\{ \text{ball}(\mathsf{p}, \mathsf{d}(\mathsf{p}, \text{sites}(\mathsf{f}))) \;\middle|\; \mathsf{p} \in \mathsf{f} \right\}. \tag{1}$$

The *trail* of f is the union of all these balls; that is, $\text{trail}(\mathsf{f}) = \bigcup_{\mathsf{p} \in \mathsf{f}} \text{ball}(\mathsf{p}, \mathsf{d}(\mathsf{p}, P))$. Finally, let $\text{mb}(\mathsf{f})$ denote the smallest ball in the pencil of a feature f. Clearly, the center of $\text{mb}(\mathsf{f})$ is the point $\mathsf{nn}(\mathsf{p}, \mathsf{f})$, where p is some arbitrary point of $\text{sites}(\mathsf{f})$. As such, $\text{mb}(\mathsf{f})$ can be computed in constant time.

▶ **Lemma 13** (proof in [7]). *Let p be any point and let f be any k-feature. The point p induces a halfflat of $\text{span}(\mathsf{f})$ denoted by $\mathcal{H}(\mathsf{p}, \mathsf{f})$, such that $\text{pencil}(\mathcal{H}(\mathsf{p}, \mathsf{f}))$ is the set of all balls in $\text{pencil}(\text{span}(\mathsf{f}))$ that contain p.*

We are now ready to restate our problem in a more familiar language.

▶ **Lemma 14** (Restatement of the problem {proof in [7]}). *Given a set P of n points in $\mathbb{R}^d$, and a pair of disjoint subsets (P_1, P_2), finding a minimum size Voronoi separator realizing the separation of (P_1, P_2), is equivalent to finding a minimum size hitting set of points Z, such that Z stabs (the interior) of all the balls in the set*

$$\mathcal{B} = \mathcal{B}(P, P_1, P_2) = \bigcup_{\mathsf{f} \in \mathcal{F}_{\text{bad}}(P, P_1, P_2)} \text{pencil}(\mathsf{f}). \tag{2}$$

4.2 Exact algorithm in $\mathbb{R}^d$

Given a set P of n points in $\mathbb{R}^d$, a pair of disjoint subsets (P_1, P_2) of P and an upper bound b on the number of guards, we show how one can compute a minimum size Voronoi separator realizing their separation in $n^{O(d^2\mathsf{b})}$ time. Our approach is to construct a small number of polynomial inequalities that are necessary and sufficient conditions for separation, and then use cylindrical algebraic decomposition to find a feasible solution.

▶ **Lemma 15** (proof in [7]). *For a set of guards Z, a feature f is completely removed from $\mathcal{V} = \mathcal{V}(P \cup Z)$, if and only if the induced halfflats of the guards, on* $\operatorname{span}(\mathsf{f})$*, cover f.*

▶ **Observation 16.** *Given a set $\mathcal{H}$ of at least $k+1$ halfflats in a k-flat, if $\mathcal{H}$ covers the k-flat, then there exists a subset of $k+1$ halfflats of $\mathcal{H}$, that covers the k-flat. This is a direct consequence of Helly's theorem*[1]*.*

Low dimensional example. To get a better understanding of the problem at hand, the reader may imagine the subproblem of removing a bad 2-feature f (i.e. f is a triangle) from the Voronoi diagram. We know that a set Z of n guards removes f from the Voronoi diagram, if and only if the n 2-halfflats induced by Z cover the triangle f. If we add the feature-induced halfflats induced by f to the above set of halfflats, then the problem of covering the triangle f, reduces to the problem of covering the entire plane with this new set S of $n+3$ halfflats. Then from Observation 16, we have that f is removed from the Voronoi diagram if and only if there are three halfplanes of S, that cover the entire plane (there are $O(n^3)$ such triplets). Lemma 20 below shows how to convert the condition that any three 2-halfflats cover the plane, into a polynomial inequality of degree four in the coordinates of the guards.

4.2.1 Constructing the Conditions

▶ **Lemma 17** (proof in [7]). *Let f be a k-feature, and $\mathcal{H}$ be a set of k-halfflats on* $\operatorname{span}(\mathsf{f})$*. Then, $\mathcal{H}$ covers f $\iff$ there exists a subset $\mathcal{G} \subseteq \mathcal{H}' = \mathcal{H} \cup \operatorname{shell}(\mathsf{f})$ of size $k+1$ that covers* $\operatorname{span}(\mathsf{f})$*.*

▶ **Observation 18.** *Consider a set $\mathcal{H}$ of $k+1$ k-halfflats all contained in some k-flat F. We are interested in checking that $\mathcal{H}$ covers F. Fortunately, this can be done by computing the $k+1$ vertices induced by $\mathcal{H}$ on F, and verifying that each one of them is covered by the other halfflat of $\mathcal{H}$. Formally, $\mathcal{H}$ covers F $\iff$ for every halfflat $h \in \mathcal{H}$, we have that $\bigcap_{i \in \mathcal{H} \setminus \{h\}} \partial i \in h$.*

For a set P of $d+1$ points in $\mathbb{R}^d$ in general position, let $\operatorname{ball}_{\text{in}}(P)$ be the unique (circumscribing) ball having all the points of P on its boundary.

▶ **Lemma 19** (proof in [7]). *Consider a k-feature f, let $\mathcal{H}$ be a set of halfflats on* $\operatorname{span}(\mathsf{f})$*, and let Z be a set of guards inducing the halfflats of $\mathcal{H}$. Assume that $|\operatorname{sites}(\mathsf{f})| + |Z| = d+2$. Let $h \in \mathcal{H}$ be a k-halfflat induced by a guard g, and let $\mathsf{p} = \bigcap_{i \in \mathcal{H} \setminus \{h\}} \partial i$. Then $\mathsf{p} \in h$ $\iff$* $\mathsf{g} \in \operatorname{ball}_{\text{in}}\Big(\operatorname{sites}(\mathsf{f}) \cup \Big(Z \setminus \{\mathsf{g}\}\Big)\Big)$.

[1] Indeed, the intersection of the complement of the halfflats of $\mathcal{H}$ is empty. As such, by Helly's theorem there exists $k+1$ of them that have an empty intersection, and the union of their complement (which are the original halfflats) cover the k-flat.

A set Z of m guards in $\mathbb{R}^d$ can be interpreted as a vector in $\mathbb{R}^{dm}$ encoding the locations of the guards. One can then reduce the requirement that Z provides the desired separation into a logical formula over the coordinates of this vector.

▶ **Lemma 20** (proof in [7]). *Let* f *be a bad* k*-feature of* $\mathcal{F}_{\text{bad}}(P, P_1, P_2)$*, and let* m *be a parameter. One can compute a boolean sentence* $\mathcal{A}_{\mathsf{f}}(Z)$ *consisting of* $m^{O(k)}$ *polynomial inequalities (of degree* $\leq d+2$*), over* dm *variables, such that* $\mathcal{A}_{\mathsf{f}}(Z)$ *is true* $\iff$ *the set of* m *guards* Z *(induced by the solution of this formula) destroys* f *completely when inserted. Formally, for every* $\mathsf{f}' \in \mathcal{F}_{\text{bad}}(P \cup Z, P_1, P_2)$*, we have* $\mathsf{f} \cap \mathsf{f}' = \emptyset$*.*

4.2.2 The Result

▶ **Theorem 21.** *Let* P *be a set of* n *points in* $\mathbb{R}^d$*. Let* (P_1, P_2) *be some disjoint partition of* P *and* b *be a parameter, such that there exists a Voronoi separator for* (P_1, P_2) *of at most* b *points. A minimum size Voronoi separator can be computed in* $n^{O(d^2\mathsf{b})}$ *time, if such a separator exists of size at most* b*.*

Proof. Let g be a parameter to be fixed shortly. Let Z be a set of g guards in $\mathbb{R}^d$. By Lemma 20, the condition $\mathcal{A}(Z)$ that every bad feature is removed from the Voronoi diagram of $P \cup Z$, can be written as

$$\mathcal{A}(Z) = \bigwedge_{\mathsf{f} \in \mathcal{F}_{\text{bad}}(P, P_1, P_2)} \mathcal{A}_{\mathsf{f}}(Z).$$

Namely, Z is a Voronoi separator for (P_1, P_2). The formula $\mathcal{A}(Z)$ contains $n^{O(d)}$ degree-$(d+2)$ polynomial inequalities comprising of at most $d\mathsf{g}$ variables. By [5, Theorem 13.12], one can compute, in $n^{O(d^2\mathsf{g})}$ time, a solution as well as the sign of each polynomial in $\mathcal{P}$, for every possible sign decomposition that $\mathcal{P}$ can attain. Now for each attainable sign decomposition, we simply check if $\mathcal{A}(Z)$ is true. This can be done in $n^{O(d)}$ time. The algorithm computes a Voronoi separator for (P_1, P_2) in P of size g, in $n^{O(d^2\mathsf{g})}$ time, if such a separator exists.

Now, the algorithms tries $\mathsf{g} = 1, \ldots, \mathsf{b}$ and stops as soon as a feasible solution is found. ◀

5 Constant factor approximation

Given a set P of n points in $\mathbb{R}^d$, and a partition (P_1, P_2) of P, we show how one can compute in $n^{O(d)}$ time, a Voronoi separator Z for (P_1, P_2), whose size is a constant factor approximation to the size of an optimal Voronoi separator realizing such a partition.

▶ **Theorem 22** (proof in [7]). *Let* P *be a set of* n *points in* $\mathbb{R}^d$*, and* (P_1, P_2) *be a given pair of disjoint subsets of* P*. One can compute a Voronoi separator* Z *that realizes this partition. The algorithm runs in* $O(n^2 \log n)$ *time for* $d = 2$*, and in* $O\left(n^{\lceil d/2 \rceil + 1}\right)$ *time, for* $d > 2$*. The algorithm provides a constant factor approximation to the smallest Voronoi separator realizing this partition.*

6 Stronger Separator for PTAS

We are interested in a $(1+\varepsilon)-$approximation to an optimal Voronoi separator of a given partition. As implied by Lemma 14, this boils down to a hitting set problem over balls. The challenge here is that the set of balls is an infinite set, see Eq. (2). We need the following improved separator theorem, whose proof can be found in the full version [7].

▶ **Theorem 23** (proof in [7]). *Let* X *be a set of points in* $\mathbb{R}^d$*, and* $k > 0$ *be an integer. One can compute, in* $O(|\mathsf{X}|)$ *expected time, a set* Z *of* $O\left(k^{1-1/d}\right)$ *points and a sphere* $\mathbb{S}$ *containing* $\Theta(k)$ *points of* X *inside it, such that for any set* $\mathcal{B}$ *of balls stabbed by* X*, we have that every ball of* $\mathcal{B}$ *that intersects* $\mathbb{S}$ *is stabbed by a point of* Z*.*

7 Local Search PTAS for geometric hitting set

Here, we present a simple local search $(1+\varepsilon)$-approximation algorithm (PTAS) for hitting set problems of balls (or fat objects) in $\mathbb{R}^d$. The set of balls is either specified explicitly, or as in the case of the Voronoi partition, implicitly.

7.1 The algorithm LocalHitBalls

7.1.1 Preliminaries

Let $\mathcal{B}$ be a set of balls (possibly infinite) in $\mathbb{R}^d$ that is represented (potentially implicitly) by an input of size n, and let $\nu(\mathcal{B})$ be the size of its minimum hitting set. We assume that $\mathcal{B}$ satisfies the following properties:

(A) **Initial solution**: One can compute a hitting set X_0 of $\mathcal{B}$ in $n^{O(1)}$ time, such that the size of X_0 is a constant factor approximation to the optimal.

(B) **Local exchange**: Let X and Y be point sets, such that
- **(i)** X is a hitting set of $\mathcal{B}$ (i.e., it is the current solution),
- **(ii)** $|\mathsf{X}| \leq n^{O(1)}$ (i.e., it is not too large),
- **(iii)** $\mathsf{Y} \subseteq \mathsf{X}$ (i.e., Y is the subset to be replaced),
- **(iv)** $|\mathsf{Y}| \leq \ell$, where ℓ is any integer.

Then one can compute in $n^{O(\ell)}$ time, the smallest set Y', such that $(\mathsf{X} \setminus \mathsf{Y}) \cup \mathsf{Y}'$ is a hitting set of $\mathcal{B}$.

7.1.2 Algorithm

The input is a set $\mathcal{B}$ of balls in $\mathbb{R}^d$ satisfying the properties above. The algorithm works as follows:

(A) Compute an initial hitting set X_0 of $\mathcal{B}$ (see (P1)), and set $\mathsf{X} \leftarrow \mathsf{X}_0$.

(B) While there is a beneficial exchange of size $\leq \ell = O\left(1/\varepsilon^d\right)$ in X, carry it out using (P2). Specifically, verify for every subset $\mathsf{Y} \subseteq \mathsf{X}$ of size at most ℓ, if it can be replaced by a strictly smaller set Y' such that $(\mathsf{X} \setminus \mathsf{Y}) \cup \mathsf{Y}'$ remains a hitting set of $\mathcal{B}$. If so, set $\mathsf{X} \leftarrow (\mathsf{X} \setminus \mathsf{Y}) \cup \mathsf{Y}'$, and repeat.

(C) If no such set exists, then return X as the hitting set.

Details. For the geometric hitting set problem where $\mathcal{B}$ is a set of n balls in $\mathbb{R}^d$, (P1) follows by a simple greedy algorithm hitting the smallest ball (in the spirit of Theorem 22) – see also [9]. As for (P2), one can check for a smaller hitting set of size at most ℓ by computing the arrangement $\mathcal{A}(\mathcal{B})$, and directly enumerating all possible hitting sets of size at most ℓ.

The more interesting case is when the set $\mathcal{B}$ is defined implicitly by an instance of the minimum Voronoi separation problem (i.e., we have a set P of n points in $\mathbb{R}^d$, and a desired Voronoi partition (P_1, P_2) of P). Then (P1) follows from Theorem 22. Furthermore, (P2) follows from the algorithm of Theorem 21 through computing the minimum hitting set Y' of $\mathcal{B}(P \cup (\mathsf{X} \setminus \mathsf{Y}), P_1, P_2)$, in $n^{O(\ell)}$ time, where $|\mathsf{Y}'| \leq \ell$.

ClusterLocalOpt($\mathcal{B}$, $\mathcal{L}$, $\mathcal{O}$):
  // $\mathcal{B}$: given set of balls need hitting.
  // $\mathcal{L} \subseteq \mathbb{R}^d$: local optimum set hitting $\mathcal{B}$ computed by LocalHitBalls.
  // $\mathcal{O} \subseteq \mathbb{R}^d$: optimal hitting set.
 $k := O\big(1/\varepsilon^d\big)$, $L_1 := \mathcal{L}$, $O_1 := \mathcal{O}$, $Z_1 := \emptyset$, $\mathcal{B}_1 := \mathcal{B}$, and $i := 1$.
 while $L_i \cup Z_i \neq \emptyset$ **do**
  Apply Theorem 23 to $L_i \cup Z_i$ with the parameter $\min(k, |L_i \cup Z_i|)$.
  b_i, T_i: ball and the separator set returned, respectively.
  $f_i := b_i \setminus \bigcup_{j=1}^{i-1} b_j$ be the region of space newly covered by b_i.
  $\mathcal{L}_i := \mathcal{L} \cap f_i$ and $\mathcal{O}_i = \mathcal{O} \cap f_i$.
  $Z_{i+1} := (Z_i \setminus b_i) \cup T_i$.
  $L_{i+1} := L_i \setminus \mathcal{L}_i$.
  $i := i + 1$
 $I := i$.

Figure 5 Clustering the local optimal solution.

7.2 Quality of approximation

The bound on the quality of approximation follows by a clustering argument similar to our bottom-up PTAS algorithm (see full version [7]). The clustering is described in Figure 5. This simultaneously breaks up the local optimal solution $\mathcal{L}$ and the optimal solution $\mathcal{O}$ into small clusters, and we will use a local exchange argument to bound their corresponding sizes. The following lemma testifies that this clustering algorithm provides a "smooth" way to move from $\mathcal{L}$ to $\mathcal{O}$, through relatively small steps.

▶ **Lemma 24.** *Any ball of $\mathcal{B}$ is stabbed by $\mathsf{X}_{i+1} = L_{i+1} \cup Z_{i+1} \cup \bigcup_{j=1}^{i} \mathcal{O}_j$, for all i, where $L_{i+1} = \bigcup_{j=i+1}^{I} \mathcal{L}_i$.*

Proof. The claim holds clearly for $i = 0$, as $L_1 = \mathcal{L}$, where $\mathcal{L}$ is the locally optimal solution. Now, for the sake of contradiction, consider the smallest i for which the claim fails, and let $b \in \mathcal{B}$ be the ball that is not being stabbed. We have that b is stabbed by $\mathsf{X}_i = \left(\bigcup_{j=i}^{I} \mathcal{L}_i\right) \cup Z_i \cup \left(\bigcup_{j=1}^{i-1} \mathcal{O}_j\right)$ but not by X_{i+1}.

It can not be that b is stabbed by a point of $\bigcup_{j=1}^{i-1} \mathcal{O}_j$, as such a point is also present in X_{i+1}. As such, b must be stabbed by one of the points of $L_i \cup Z_i$, and then this stabbing point must be inside b_i – indeed, the points removed from L_i and Z_i as we compute L_{i+1} and Z_{i+1}, respectively, are the ones inside b_i. Now, if b intersects ∂b_i, then $T_i \subseteq Z_{i+1}$ stabs b by Theorem 23, a contradiction.

So it must be that $b \subseteq b_i$. But then consider the region $F_i = \cup_{j=1}^{i} f_j = \cup_{j=1}^{i} b_j$. It must be that $b \subseteq F_i$. This in turn implies that the point of $\mathcal{O}$ stabbing b is in $\mathcal{O} \cap F_i = \cup_{j=1}^{i} \mathcal{O}_i \subseteq \mathsf{X}_{i+1}$. A contradiction. ◀

▶ **Lemma 25.** *Consider any ball $b \in \mathcal{B}$. Let i be the smallest index such that $\mathcal{O}_i$ stabs b. Then b is stabbed by $\mathcal{L}_i \cup T_i \cup (Z_i \cap b_i)$. Furthermore for all i, $o_i \leq \lambda_i + t_i + z_i$ and $o_i = O(k)$, where $o_i = |\mathcal{O}_i|$, $t_i = |T_i|$, $\lambda_i = |\mathcal{L}_i|$, and $z_i = |Z_i \cap b_i|$.*

Proof. If b intersects ∂b_i, then it is stabbed by T_i by Theorem 23. Otherwise, $b \subseteq F_i = \cup_{j=1}^{i} f_j = \cup_{j=1}^{i} b_j$. In particular, this implies that no later point of $\mathcal{O}_{i+1} \cup \ldots \cup \mathcal{O}_t$ can stab b. That is, only $\mathcal{O}_i$ stabs b. By Lemma 24 both X_i and X_{i+1} stab b, where $\mathsf{X}_i =$

$(\mathcal{L} \setminus F_{i-1}) \cup Z_i \cup \bigcup_{j=1}^{i-1} \mathcal{O}_j$. Namely, b is stabbed by a point of $(\mathcal{L} \setminus F_{i-1}) \cup Z_i$ that is contained inside b_i. Such a point is either in $Z_i \cap b_i$, or in $(\mathcal{L} \setminus F_{i-1}) \cap b_i = \mathcal{L}_i$, as claimed.

The second part follows by observing that otherwise $\mathcal{O}_i$ can be replaced by $\mathcal{L}_i \cup T_i \cup (Z_i \cap b_i)$ in the optimal solution.

The third claim follows by observing that by the algorithm design $\lambda_i + z_i = O(k)$, and $t_i = O\big(k^{1-1/d}\big)$. As such, $o_i \leq \lambda_i + t_i + z_i = O(k)$. ◀

▶ **Lemma 26.** *Consider any ball $b \in \mathcal{B}$ that is stabbed by $\mathcal{L}_i$ but it is not stabbed by $\mathcal{L} \setminus \mathcal{L}_i$. Then the ball b is stabbed by $\mathcal{O}_i \cup T_i \cup (Z_i \cap b_i)$. Additionally, using the notations of Lemma 25, we have $\lambda_i \leq o_i + t_i + z_i$.*

Proof. If b intersects ∂b_i, then it is stabbed by T_i by Theorem 23. So we assume for now on that b is not stabbed by T_i. But then, $b \subseteq b_i \subseteq F_i = \cup_{j=1}^{i} f_j = \cup_{j=1}^{i} b_j$.

Now, by Lemma 24 both X_i and X_{i+1} stab b, where $\mathsf{X}_{i+1} = \bigcup_{j=i+1}^{I} \mathcal{L}_j \cup Z_{i+1} \cup \bigcup_{j=1}^{i} \mathcal{O}_j$. Specifically, b is stabbed by a point of $Z_{i+1} \cup \bigcup_{j=1}^{i} \mathcal{O}_j$ that is contained inside b_i. Such a point is either in $Z_i \cap b_i$ and then we are done, or alternatively, it can be in $b_i \cap \bigcup_{j=1}^{i} \mathcal{O}_j$.

Now, if $b \subseteq f_i$ then $\mathcal{O}_i$ must stab b, and we are done. Otherwise, let $k < i$ be the maximum index such that b_i intersects ∂b_k. Observe that as b intersects f_i, it can not be that it intersects the balls $b_{k+1}, \ldots, b_{i-1}$. In particular, Theorem 23 implies that there is a point of T_k that stabs b, as b is being stabbed by $L_k \cup Z_k$. This point of T_k is added to Z_{k+1}, and it is not being removed till Z_{i+1}. As such, this point is in Z_i, and it is also in b_i, thus implying the claim.

As for the second part, by Lemma 25, $o_i + t_i + z_i \leq \alpha = O(k)$. As such, setting $\ell = \Omega(1/\varepsilon^d)$ to be sufficiently large, we have that $\ell > 2\alpha$, and the local search algorithm would consider the local exchange of $\mathcal{L}_i$ with $\mathcal{O}_i \cup T_i \cup (Z_i \cap b_i)$. As this is an exchange not taken, it must be that it is not beneficial, implying the inequality. ◀

▶ **Lemma 27.** *We have that* (i) $\sum_{i=1}^{I} t_i \leq (\varepsilon/4)\,|\mathcal{O}|$, *and* (ii) $\sum_{i=1}^{I} z_i \leq (\varepsilon/4)\,|\mathcal{O}|$.

Proof. Observe that $k = O(1/\varepsilon^d)$ and $t_i = |T_i| = O(k^{1-1/d}) = O\big(1/\varepsilon^{d-1}\big) \leq c\varepsilon k$, where c can be made arbitrarily small by making k sufficient large. In particular, in every iteration, the algorithm removes $\geq k$ points from $L_i \cup Z_i$, and replaces them by t_i points. Starting with a solution of size $|\mathcal{L}| \leq |\mathsf{X}_0| \leq c'\,|\mathcal{O}|$, where c' is some constant, this can happen at most $I \leq c'\,|\mathcal{O}|\,/(k - c\varepsilon k) = O(\varepsilon^d\,|\mathcal{O}|)$. As such, we have $\sum_{i=1}^{I} t_i = O\big(\varepsilon^d\,|\mathcal{O}|\,c\varepsilon k\big) = O(c\varepsilon\,|\mathcal{O}|)$, as $k = O(1/\varepsilon^d)$. The claim now follows by setting k to be sufficiently large.

The second claim follows by observing that t_i counts the numbers of points added to Z_{i+1}, while z_i counts the number of points removed from it. As Z_I is empty, it must be that $\sum_i t_i = \sum_i z_i$. ◀

▶ **Lemma 28.** *We have that $|\mathcal{L}| \leq (1+\varepsilon)\,|\mathcal{O}|$.*

Proof. We have by Lemma 26 and Lemma 27 that $|\mathcal{L}| = \sum_{i=1}^{I} \lambda_i \leq \sum_{i=1}^{I} (o_i + t_i + z_i) = |\mathcal{O}| + \sum_{i=1}^{I} t_i + \sum_{i=1}^{I} z_i \leq (1 + \varepsilon/2)\,|\mathcal{O}|$. ◀

7.2.1 The Result

▶ **Theorem 29.** *Let $\mathcal{B}$ be a set of balls in $\mathbb{R}^d$ satisfying properties (P1) and (P2). Then* ***LocalHitBalls*** *computes, in $n^{O(1/\varepsilon^d)}$ time, a hitting set of $\mathcal{B}$, whose size is a $(1+\varepsilon)$-approximation to a minimum size hitting set of $\mathcal{B}$.*

Proof. Lemma 28 implies the bound on the quality of approximation.

As for the runtime, observe that the size of local solution reduces by at least one after each local improvement, and from (P1), the initial local solution has size $n^{O(1)}$. Thus there can be at most $n^{O(1)}$ local improvement steps before the algorithm stops. Furthermore, every local solution has at most $n^{O(\ell)}$ subsets of size ℓ that are checked for local improvement. By (P2), such a local improvement can be checked in $n^{O(\ell)}$ time. Thus **LocalHitBalls** runs in $n^{O\left(1/\varepsilon^d\right)}$ time. ◀

8 Conclusions

We presented a new separator result that provides a new way to perform geometric divide and conquer for Voronoi diagrams (or Delaunay triangulations). We use this to derive a new PTAS for the Voronoi partition problem, making progress on a geometric hitting set problem where the ranges to be hit are defined implicitly, and their number is infinite. Significantly, the resulting local search algorithm is relatively simple, and should have other applications [4].

There are many interesting open problems for further research. In particular, the new PTAS might be more practical for the piercing balls problem than previous algorithms, and it might be worthwhile to further investigate this direction. Additionally, the proof technique for the local search algorithm might be applicable to other separator based geometric problems.

References

1 N. Alon, P. Seymour, and R. Thomas. A separator theorem for nonplanar graphs. In *J. Amer. Math. Soc.*, volume 3, pages 801–808, 1990. doi:10.1090/S0894-0347-1990-1065053-0.

2 V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.

3 F. Aurenhammer, R. Klein, and D.-T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013. doi:10.1142/8685.

4 S. Bandyapadhyay and K. R. Varadarajan. On variants of k-means clustering. *CoRR*, abs/1512.02985, 2015. Also in SoCG 16. URL: http://arxiv.org/abs/1512.02985.

5 S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2006. doi:10.1007/3-540-33099-2.

6 M. de Berg, O. Cheong, M. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Santa Clara, CA, USA, 3rd edition, 2008. URL: http://www.cs.uu.nl/geobook/.

7 V. V. S. P. Bhattiprolu and S. Har-Peled. Separating a voronoi diagram via local search. *CoRR*, abs/1401.0174, 2014. URL: http://arxiv.org/abs/1401.0174.

8 J. Böttcher, K. P. Pruessmann, A. Taraz, and A. Würfl. Bandwidth, expansion, treewidth, separators and universality for bounded-degree graphs. *Eur. J. Comb.*, 31(5):1217–1227, July 2010. doi:10.1016/j.ejc.2009.10.010.

9 T. M. Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithms*, 46(2):178–189, 2003. doi:10.1016/S0196-6774(02)00294-8.

10 T. M. Chan and S. Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete Comput. Geom.*, 48:373–392, 2012. doi:10.1007/s00454-012-9417-5.

11 Z. Dvorak and S. Norin. Strongly sublinear separators and polynomial expansion. *ArXiv e-prints*, April 2015. URL: http://arxiv.org/abs/1504.04821, arXiv:1504.04821.

12 A. Efrat and S. Har-Peled. Guarding galleries and terrains. *Inform. Process. Lett.*, 100(6):238–245, 2006. doi:10.1016/j.ipl.2006.05.014.

13 D. Eisenstat and P. N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *Proc. 45th Annu. ACM Sympos. Theory Comput.* (STOC), pages 735–744, 2013. doi:10.1145/2488608.2488702.

14 T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.*, 34(6):1302–1323, 2005. doi:10.1137/S0097539702402676.

15 J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Sys. Sci.*, 72(5):868–889, 2006. doi:10.1016/j.jcss.2005.05.007.

16 J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984. doi:10.1016/0196-6774(84)90019-1.

17 A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *ArXiv e-prints*, September 2008. URL: http://arxiv.org/abs/0809.2554, arXiv:0809.2554.

18 S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. Amer. Math. Soc., Boston, MA, USA, 2011. URL: http://sarielhp.org/book/, doi:10.1090/surv/173.

19 S. Har-Peled. A simple proof of the existence of a planar separator. *ArXiv e-prints*, 2011. URL: http://arxiv.org/abs/1105.0103, arXiv:1105.0103.

20 S. Har-Peled and M. Lee. Weighted geometric set cover problems revisited. *J. Comput. Geom.*, 3(1):65–85, 2012. URL: http://sarielhp.org/papers/08/expand_cover/.

21 S. Har-Peled and K. Quanrud. Approximation algorithms for polynomial-expansion and low-density graphs. In *Proc. 23nd Annu. Euro. Sympos. Alg.* (ESA), volume 9294 of *Lect. Notes in Comp. Sci.*, pages 717–728, 2015. doi:10.1007/978-3-662-48350-3_60.

22 P. N. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM J. Comput.*, 37(6):1926–1952, 2008. doi:10.1137/060649562.

23 P. Koebe. Kontaktprobleme der konformen Abbildung. *Ber. Verh. Sächs. Akademie der Wissenschaften Leipzig, Math.-Phys. Klasse*, 88:141–164, 1936.

24 R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36:177–189, 1979. doi:10.1137/0136016.

25 R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. doi:10.1137/0209046.

26 G. L. Miller, S. H. Teng, W. P. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. Assoc. Comput. Mach.*, 44(1):1–29, 1997. doi:10.1145/256292.256294.

27 N. H. Mustafa, R. Raman, and S. Ray. Settling the APX-hardness status for geometric set cover. In *Proc. 55th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 541–550, 2014. doi:10.1109/FOCS.2014.64.

28 N. H. Mustafa and S. Ray. Improved results on geometric hitting set problems. *Discrete Comput. Geom.*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

29 J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion I. decompositions. *Eur. J. Comb.*, 29(3):760–776, 2008.

30 J. O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, 1987. URL: http://cs.smith.edu/~orourke/books/art.html.

31 J. Pach and P. K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, 1995. URL: http://www.addall.com/Browse/Detail/0471588903.html.

32 W. D. Smith and N. C. Wormald. Geometric separator theorems and applications. In *Proc. 39th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 232–243, 1998. doi:10.1109/SFCS.1998.743449.

33 C. Sommer, E. Verbin, and W. Yu. Distance oracles for sparse graphs. In *Proc. 50th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 703–712, Georgia, USA, 2009. doi:10.1109/FOCS.2009.27.

34 P. Ungar. A theorem on planar graphs. *J. London Math. Soc.*, 26:256–262, 1951.

On Visibility Representations of Non-Planar Graphs*

Therese Biedl[†1], Giuseppe Liotta[‡2], and Fabrizio Montecchiani[§3]

1 **David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada**
biedl@uwaterloo.ca

2 **Dipartimento di Ingegneria, Università degli Studi di Perugia, Perugia, Italy**
giuseppe.liotta@unipg.it

3 **Dipartimento di Ingegneria, Università degli Studi di Perugia, Perugia, Italy**
fabrizio.montecchiani@unipg.it

Abstract

A rectangle visibility representation (RVR) of a graph consists of an assignment of axis-aligned rectangles to vertices such that for every edge there exists a horizontal or vertical line of sight between the rectangles assigned to its endpoints. Testing whether a graph has an RVR is known to be NP-hard. In this paper, we study the problem of finding an RVR under the assumption that an embedding in the plane of the input graph is fixed and we are looking for an RVR that reflects this embedding. We show that in this case the problem can be solved in polynomial time for general embedded graphs and in linear time for 1-plane graphs (i.e., embedded graphs having at most one crossing per edge). The linear time algorithm uses a precise list of forbidden configurations, which extends the set known for straight-line drawings of 1-plane graphs. These forbidden configurations can be tested for in linear time, and so in linear time we can test whether a 1-plane graph has an RVR and either compute such a representation or report a negative witness. Finally, we discuss some extensions of our study to the case when the embedding is not fixed but the RVR can have at most one crossing per edge.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Visibility Representations, 1-Planarity, Recognition Algorithm, Forbidden Configuration

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.19

1 Introduction

A visibility representation is an appealing method of displaying graphs. It consists of assigning axis-aligned rectangles to vertices and horizontal or vertical line segments to edges in such a way that line segments of edges begin and end at the rectangles representing their endpoints and intersect no rectangles in-between. Equivalently, each edge corresponds to a horizontal or

* Because of space limitations some proofs are only sketched or omitted; complete proofs can be found in the full version of the paper [2].

† Research of TB supported by NSERC.

‡ Research of GL supported in part by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001.

§ Research of GL and FM supported in part by the MIUR project AMANDA "Algorithmics for MAssive and Networked DAta", prot. 2012C4E3KT_001. Research undertaken while FM was visiting the University of Waterloo.

© Therese Biedl, Giuseppe Liotta, and Fabrizio Montecchiani;
licensed under Creative Commons License CC-BY

vertical line-of-sight between its endpoints; hence the name "visibility". Edge segments may cross each other, but any such crossing occurs at a right angle. These right-angle crossings as well as the absence of bends leads to a high readability of the visualization. See Fig. 1.

Visibility representations were introduced to the Computational Geometry community in 1985, when Wismath [26] showed that every planar graph has a bar-visibility representation, i.e., where vertices are represented as horizontal bars and edges correspond to vertical visibilities. (The same result was discovered independently multiple times [10, 17, 19, 23, 24].)

Such bar-visibility representations can exist only for planar graphs, so for representing non-planar graphs one must use edge segments in both directions. (We use the term *rectangle visibility representation* or *RVR* to distinguish such drawings from bar-visibility representations.) Clearly not any graph G can have an RVR either; for example G must have thickness 2 (i.e., be the union of two planar graphs), and, as was shown in [4], it must have at most $6n - 20$ edges. But neither of these conditions is sufficient. In fact, Shermer showed in 1996 that it is NP-hard to test whether a given graph has an RVR [20].

In this paper, we study the problem of testing whether a graph has a rectangle visibility representation if some aspects of it are fixed a priori. In previous work by Streinu and Whitesides [21], it was shown that testing whether a graph has an RVR is polynomial if the following additional information is given: we know for each edge whether it is drawn horizontal or vertical, we know the rotational scheme (i.e., the fixed order of edges around each vertex), and we know which vertices form the outer face of the visibility representation. Streinu and Whitesides give necessary and sufficient conditions for when such restrictions can be realized, and show that these can be tested in $O(n^2)$ time.

In this paper, we study a slightly different scenario. Like Streinu and Whitesides, we assume that the rotational scheme and the outer face is fixed. However, we do not require to know the directions of the edges; instead we must know a priori which edges cross each other, and the order in which these crossings occur along each edge (if there is more than one crossing on an edge). In other words, the graph has a *fixed embedding* (see also Section 2). A reduction to the topology-shape-metrics approach then allows to test in $O(n^{1.5} \log n)$ time whether a graph with n vertices has a rectangle visibility representation; see Section 2.2.

As our main result, we then consider the case when any edge contains at most one crossing (the so-called *1-planar graphs*). Previous attempts to create visibility representations for 1-planar graphs had relaxed the restriction that edges must not cross non-incident vertices, and defined a *bar k-visibility drawing* to be a bar-visibility drawing where each line of sight can intersect at most k bars [8]. In other words, each edge can cross at most k vertices. These crossings are arguably less intuitive than crossings between edges. Brandenburg *et al.* [5] and independently Evans *et al.* [11] prove that 1-planar graphs have a bar 1-visibility drawing (if we are allowed to change the embedding).

In contrast to these results, we consider in Section 3 embedding-preserving rectangle visibility representations of embedded 1-planar graphs, also called 1-plane graphs. It is easy to see that not all 1-plane graphs have such an RVR (see also Section 4). The main contribution of our paper is to show that for a 1-plane graph it is possible to test in linear time whether it has an RVR. The approach is entirely different from the topology-shape-metrics above and reveals much insight into the structure required for an RVR to exist. Specifically, we consider three configurations (called B-configuration, W-configuration, and the newly defined T-configuration) and show that a 1-plane graph has an RVR if and only if none of these three configurations occurs. This mirrors in a pleasing way the result by Thomassen [25], which shows that a 1-plane graph has a straight-line drawing if and only if it contains no B-configuration or W-configuration. Our result also provides an answer (in a special case)

to Shermer's question of characterizing when graphs have RVRs [20]. Also, we prove that testing whether a 1-plane graph contains any of the three forbidden configurations can be done in linear time, and in the absence of them we can compute in linear time an RVR which fits into a grid of quadratic size. It may be worth recalling that embedding-preserving straight-line drawings of 1-plane graphs may require exponential area [13]. In Section 4 we show both negative and positive results in the variable embedding setting. Finally, in Section 5 we conclude with a discussion and open problems.

2 Definitions and Basic Results

We only consider *simple graphs*, i.e., graphs with neither loops nor multiple edges. A *drawing* Γ of a graph G maps each vertex to a point of the plane and each edge to a Jordan arc between its two endpoints. We only consider *simple drawings*, i.e., drawings such that the arcs representing two edges have at most one point in common, which is either a common endpoint or a common interior point where the two arcs properly cross. A drawing is *planar* if no two arcs cross. A drawing divides the plane into topologically connected regions, called *faces*. The unbounded region is called the *outer face*. For a planar drawing the boundary of a face consists of vertices and edges, while for a non-planar drawing the boundary of a face may contain vertices, crossings, and edges (or parts of edges). An *inner* face/edge/vertex is a face/edge/vertex that is not (part of) the outer face. A *rotational scheme* of a graph G consists of assigning at each vertex a cyclic order of the edges incident to that vertex. A drawing is said to respect a rotational scheme if scanning around the vertex in clockwise order encounters the edges in the prescribed order. We say that a graph has a *fixed embedding* if the graph comes with a rotational scheme, a fixed order of crossings along each edge, and with one face indicated to be the outer face. (One can show that fixing the rotational scheme and the order of crossings fixes all faces.) Given a graph G with a fixed embedding, the *planarization* G_p of G is the graph obtained by replacing each crossing of G with a *dummy-vertex*. The vertices of G in G_p are called *original vertices* to avoid confusion. The planarization G_p is a *plane* graph, i.e., a planar graph with a fixed embedding without crossing.

A *visibility representation* (also called *rectangle visibility representation* or RVR for short) of a graph G is an assignment of disjoint axis-aligned rectangles to the vertices in such a way that for every edge there exists a horizontal or vertical line of sight between its endpoints (not all such line of sights give rise to an edge). We follow a commonly adopted model where the lines of sight are *thick*, i.e., have non-zero area (see e.g. [16, 21, 23, 26]). Thus we may assume that each edge is routed as a (horizontal or vertical) line segment, attaching at each end at a point that is not a corner of the rectangle of that vertex. In what follows, when this leads to no confusion, we shall use the same term *edge* to indicate both an edge of a graph, the line of sight between two rectangles, and the line segment representing the edge, and we use the same term *vertex* for both the vertex of a graph and the rectangle that represents it. We observe that if we have an RVR, then we can naturally extract a drawing from it as follows. Place a point for each vertex v inside the rectangle representing v and connect it to all the attachment points of incident edges of v inside the rectangle; this adds no crossing. We say that an RVR Γ of a graph G *respects* the embedding of G if applying this operation to Γ results in the same embedding.

2.1 Topology-shape-metrics

The *topology-shape-metrics* approach is a well-known method introduced by Tamassia [22] to create orthogonal drawings of graphs, i.e., drawings where all vertices are points and edges

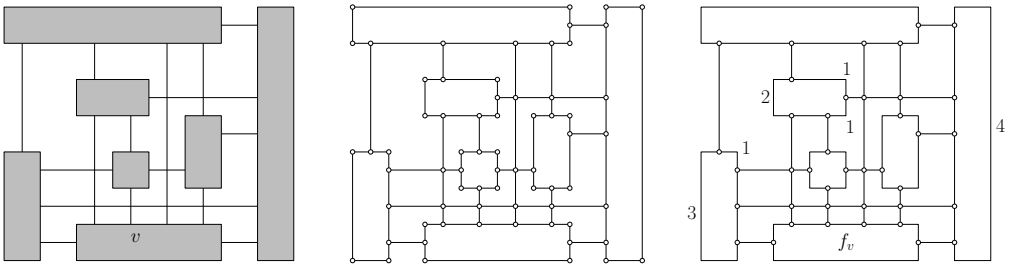


Figure 1 A visibility representation of a graph, the orthogonal representation obtained from it, and the orthogonal representation used to compute it. We show only selected non-zero bend-counts.

are curves with horizontal and vertical segments. In this section, we briefly review how any RVR can be expressed as an orthogonal-shape-metric; this is quite straightforward but has to our knowledge not been used before. We will use such metrics twice in our paper: once to determine whether a given embedded graph has an RVR, and once to store an RVR obtained for 1-plane graphs so that they can be manipulated efficiently.

Consider an RVR $\mathcal{R}$ and interpret it as a PSLG, i.e., a planar straight-line graph in the computational geometry sense, by replacing corners, attachment points and crossings by nodes as in Fig. 1. We get a planar graph that has special properties:

▶ **Definition 1.** An *orthogonal representation* is a plane graph H with maximum degree 4, an assignment of an angle $\alpha(v, f) \in \{90°, 180°, 270°\}$ to any vertex-face-incidence (between vertex v and face f) of H, and a bend-count $b(e, f) \in \{0, +1, -1, +2, -2, \dots\}$ to any edge-face-incidence (between edge e and face f) of H, such that the following is satisfied:

- at every vertex v, the angles at v sum to 360°,
- at every edge e, the two bend-counts sum to 0,
- for every inner face f with k vertices, the angles and bend-counts (multiplied by 90°) at f sum to $(k-2)180°$.
- the angles and bend-counts (multiplied by 90°) at the outer face sum to $(k+2)180°$, where k is the number of vertices on the outer face.

▶ **Observation 2.** *If G has an RVR Γ, then construct a plane graph H by replacing every crossing, every edge-vertex attachment point and every corner of a vertex-rectangle of Γ with a node in H. The resulting graph then has an orthogonal representation where all bend-counts are 0.*

Tamassia showed [22] that for any orthogonal representation, one can reconstruct a planar orthogonal drawing of the graph that respects these angles and bend-counts. Tamassia also showed how to compute for a given plane graph H some angle-assignment and bend-counts such that the result is an orthogonal representation. This is done via a flow-approach; see [22] or [9] for details. This approach allows also to specify upper and lower bounds on angles and edge-counts.

2.2 Testing embedded graphs for RVRs

Now presume that we are given a graph G with a fixed embedding. In this section, we will show how to test whether G has an RVR that respects its embedding using the topology-shape-metrics approach. Note that the embedding tells us nearly the entire graph H of the orthogonal representation of a (putative) RVR of G. The only thing unknown is the location

of the corners of each rectangle of a vertex. We can now interpret these corners as bends (rather than as nodes) in the topology. Hence define a graph H as follows (see also Fig. 1):

- Start with the planarization G_p of G.
- For any original vertex v of G, define a cycle C_v with $\deg(v)$ nodes in H. Attach the (parts of) edges incident to v to the nodes of cycle C_v in the specified order of the planar embedding, in such a way that the interior of the cycle forms a face. In other words, form a cycle that can represent the boundary of the rectangle of v.
- Impose an upper and lower bound of 0 onto the bend-counts of all (parts of) edges of G. In other words, we do not allow any bends for the original edges.
- Impose a lower bound of 0 onto each bend-count $b(e, f_v)$ where e is an edge of some cycle C_v and f_v is the face formed by cycle C_v. In other words, cycle C_v may have bends, but all such bends must form 90° angles towards f_v.
- If a is a node of C_v, then impose an upper and lower bound of 180° onto the angle $\alpha(a, f_v)$, where f_v as before is the face inside C_v. In other words, at attachment points of edges face f_v must not have a corner.

▶ **Theorem 3.** *Let G be a with a fixed embedding and n vertices and crossings. Then we can test in $O(n^{1.5} \log n)$ time whether G has an RVR that respects this embedding.*

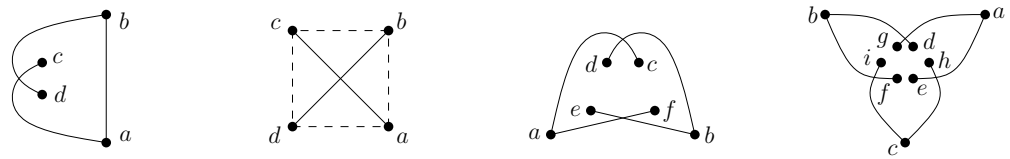
Proof. Build the graph H as described above. It should be obvious that if we can find an orthogonal representation of H that respects these constraints, then it gives rise to an RVR of G. Indeed, for any vertex v the cycle C_v must form a rectangle: It cannot have a bend at any attachment-point, and any bend that it has provides an angle of 90°, hence by the condition it has exactly four such bends. At any dummy-vertex the four angles must be 90° since its degree is four. All edges of H that came from edges of G are drawn straight, and continue straight at any crossing, so give rise to a line of sight. We then use Tamassia's flow-approach to test whether an orthogonal representation of H exists that satisfies all constraints. This takes $O(n^{1.5} \log n)$ time [7]. ◀

3 Embedded 1-planar graphs

While the topology-shape-metrics approach gives a polynomial way to test whether a graph has an RVR, its black-box approach is somewhat unsatisfactory. In particular, in case of a negative answer, we obtain no good insights as to which parts of the graph prevented the existence of a representation. In this section, we therefore turn our attention to 1-planar graphs. We chose this graph class for two reasons. One is that they are known to have thickness 2 and at most $4n - 8$ edges [18], and so they are good candidates for always having an RVR. (We will, however, see that this is not the case.) Secondly, 1-planar graphs have been studied widely in the graph theory and graph drawing communities (see e.g. [1, 5, 11, 13, 18, 25]), and visibility representations of 1-planar graphs hence should be of interest.

3.1 Background

A graph is *1-planar* if it has a drawing with at most one crossing per edge. A *1-plane* graph is a 1-planar graph with a fixed embedding. Recall that this means that we have a fixed order of edges at each vertex, it is fixed which pairs of edges cross, and we know what the faces are and which one of them is the outer face. A 1-plane graph G is called *triangulated 1-plane* if all faces are triangles. In a 1-plane graph, no two crossings can be adjacent and so a triangle is composed of three vertices or two vertices and a crossing point.



(a) B **(b)** Kite **(c)** W **(d)** T

Figure 2 Possible crossing configurations in a 1-plane graph G. (a) A B-configuration. (b) A (subgraph) of a kite. The dashed edges either do not exist or define faces. (c) A W-configuration. (d) A T-configuration.

Let $e = (a, c)$ and $e' = (b, d)$ be two edges of G that cross at a point p. We say that e, e' induce a *B-configuration* if there exists an edge between their endpoints (say edge (a, b)) such that (in the induced drawing of $\{a, b, c, d\}$) the triangle $\{a, b, p\}$ contains vertices c and d inside (see also Fig. 2a).

If the crossing is not in a B-configuration, then there are two further possibilities for edge (a, b). The triangle $(a, b), (b, p), (p, a)$ may be an interior face, or it could have other vertices inside. If, for all pairs in $\{a, c\} \times \{b, d\}$, there either is no edge between the vertices or the triangle that it forms is a face, then we could add the edges that did not exist previously, routing them along the crossing, and then obtain a *kite*, i.e., a crossing with a 4-cycle among its endpoints such that removing the crossing turns the 4-cycle into a face.

Let (a, c) and (b, d) be two edges of G that cross at a point p, and let (a, f) and (b, e) be two further edges of G that cross at a point q. The four edges induce a *W-configuration* if vertices c, d, e, f lie inside the closed region delimited by the edge segments (a, p), (b, p), (a, q), and (b, q) (see also Fig. 2c).

We introduce a fourth configuration, called the *trillium configuration*, or *T-configuration* for short, which is illustrated in Fig. 2d. Namely, let (a, c) and (b, d) be a pair of edges of G that cross at a point p. Let (a, e) and (c, h) be a second pair of edges that cross at a point q. Let (b, f) and (c, i) be a third pair of edges that cross at a point t. The six edges induce a T-configuration if vertices d, e, f, g, h, i lie inside the closed region delimited by the edge segments $a - p - b - t - c - q - a$. Vertices a, b, c are called the *outer vertices* of the T-configuration, while the remaining six vertices are the *inner vertices*.

3.2 Main result and outline

In this section we prove the following characterization of 1-plane graphs that admit an RVR:

▶ **Theorem 4.** *A 1-plane graph G admits an RVR if and only if it contains no B-configuration, no W-configuration, and no T-configuration.*

Notably, Theorem 4 extends the characterization for straight-line drawability of 1-plane graphs given by Thomassen [24], adding the T-configuration to the set of obstructions. We prove necessity in Section 3.3 and sufficiency in Section 3.4. The latter comes with an efficient algorithm to construct an RVR Γ of a drawable graph G. Since testing for forbidden configurations can also be done efficiently (Section 3.5), we get:

▶ **Theorem 5.** *Let G be a 1-plane graph with n vertices. There exists an $O(n)$-time algorithm to test whether G admits an RVR. Also, in the positive case the algorithm computes an RVR of G that has $O(n^2)$ area.*

3.3 Proof of necessary condition

To prove the necessity of Theorem 4, we show a slightly stronger statement:

▶ **Lemma 6.** *Let G be a graph that admits an RVR Γ and whose outer face is a cycle $\mathcal{C}$. If while walking along $\mathcal{C}$ we encounter k vertices and c crossing points, then $k \geq 3$ and $c \leq 2k-4$.*

Proof. Convert Γ into an orthogonal representation (Observation 2) and consider the angles at the outer face. Since there are k vertices on the outer face, we have $2k$ edge-attachment points, and each contributes 90°. Each of the c crossings contributes 90°. If b is the number of rectangle-corners on the outer face, then $b \leq 4k$, and each contributes 270°. In total the outer face has $2k+c+b$ nodes and the sum of angles must be $180°(2k+c+b+2)$. But we also know that the angles sum to $90°(2k+c)+270°b = 180°(2k+c+b+2)+90°(b-2k-c-4)$. So $b-2k-c-4=0$, hence $4k \geq b = 2k+c+4$, hence $c \leq 2k-4$. This implies $k \geq 2$, and in fact $k \geq 3$ is required, else $c=0$ and the outer face would consist of a double edge. ◀

Now assume that graph G a B-configuration (W-configuration, T-configuration), and consider the subgraph G' induced by that configuration. Then the outer face of this subgraph is a cycle with $k=2$ and $c=1$ ($k=2$ and $c=2$, $k=3$ and $c=3$). So G' (and with it, G) cannot have an RVR and the necessity holds.

3.4 Proof of sufficient condition

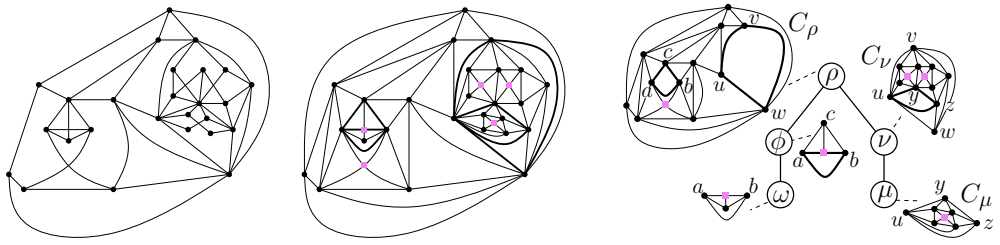
We show that the absence of any B-/W-/T-configuration suffices to compute an RVR. Let G be a 1-plane graph with n vertices and with no B-/W-/T-configuration as a subgraph, see also Fig. 3a for an example. We describe a drawing algorithm, `1PRVDrawer`, which computes an RVR Γ of G. We assume that G is 3-connected (Sections 3.4.1–3.4.6); an extension to the general case is proved in the full-length version [2].

3.4.1 Triangulating and planarizing

The first step of algorithm `1PRVDrawer` is to triangulate G. This is a well-known operation for 3-connected 1-plane graphs, see [1, 13]. However, these algorithms all modify the given 1-planar embedding, even if there is no B-configuration or W-configuration. In fact, this may be required since every crossing must form a kite in a triangulation of G, and so if an edge between endpoints of a crossing already existed, but did not form a face, then it must be re-routed. We do not want to change the embedding, and hence cannot triangulate the 1-plane graph per se. Instead, we combine the triangulation step with a planarization step to ensure that the result has no multiple edges, yet at all crossings (respectively the corresponding dummy-vertices) we have kites.

We construct the triangulated planar graph G^+ with the following steps:

- Let G_1 be the planarization of G. Note that dummy-vertices have degree 4 (and we will maintain this throughout later modifications).
- For each dummy-vertex z of G_1, let $v_0, \ldots, v_3$ be the four neighbors in clockwise order. For $i = 0, \ldots, 3$, consider the face at z between the edges (z, v_i) and (z, v_{i+1}) (addition modulo 4). If this face is the outer face, or an inner face that is not a triangle, then add the edge (v_i, v_{i+1}). Put differently, we add the edges that are needed to turn z into a kite. (In what follows, we will use the previously defined terms "kite", "B-configuration", "W-configuration" and "T-configuration" for the planarized version as well, i.e., for the



(a) G **(b)** G^+ **(c)** $\mathcal{T}$

Figure 3 (a) A 1-plane graph G. (b) The triangulated planar graph G^+ obtained from G. Dummy-vertices are pink squares. (c) The 4-block tree $\mathcal{T}$ of G^+.

same situations with crossings replaced by dummy-vertices.) Call the resulting graph G_2. The following result (proved in [2]) is crucial for our correctness:

▶ **Lemma 7.** *If G is simple and 3-connected and has no B-configuration, W-configuration or T-configuration, then G_2 has no B-configuration, W-configuration, or T-configuration and no multiple edges.*

In the plane graph G_2 each dummy-vertex is surrounded by a kite by construction. Hence all faces of G_2 that are not triangles contain only original vertices. Triangulate each such face arbitrarily; this does not create a B-configuration or multiple edges since in a planar graph this would imply a cutting pair.

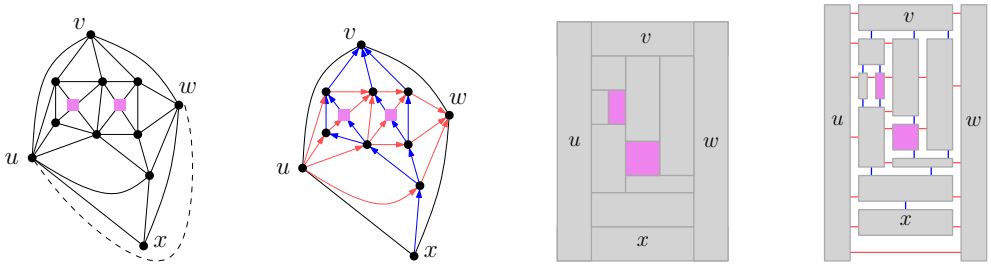
The resulting graph G^+ is the plane triangulated graph that we use to obtain the RVR. By construction it has the following crucial properties: It is simple, has no B-configuration, W-configuration or T-configuration, every dummy-vertex has degree 4, and removing from G^+ all the edges inserted in the above procedure, and replacing each dummy-vertex with a crossing point, we obtain the original 1-plane graph G.

3.4.2 The 4-block tree and an outline

Algorithm `1PRVDrawer` uses a decomposition of G^+ into its 4-connected components, i.e., a 4-connected triangulated subgraph of G^+. Let the *4-block tree* $\mathcal{T}$ of G^+ be a tree defined as follows (see also [14] and Fig. 3c). Every 4-connected component C_ν of G^+ is represented by a node ν in $\mathcal{T}$. There is an edge between two nodes ν and μ in $\mathcal{T}$, if there is a *separating triangle* (i.e., a triangle with vertices both inside and outside) that belongs to both C_ν and C_μ. We root $\mathcal{T}$ at the node ρ with the 4-connected component that contains the outer face. Then for any parent ν and child μ, the separating triangle common to C_ν and C_μ is an inner face in C_ν and the outer face of C_μ. It is known that $\mathcal{T}$ can be computed in $O(n)$ time [14].

We now give an overview of the remaining steps of algorithm `1PRVDrawer`. It first visits $\mathcal{T}$ top-down and determines for each 4-connected component C_ν a special edge, called *surround-edge*. It also computes an RVR γ_ν of C_ν using a so-called rectangular dual representation. Next, the algorithm visits $\mathcal{T}$ bottom-up. Let ν be a node of $\mathcal{T}$ with $h \geq 1$ children $\mu_1, \ldots, \mu_h$. Denote by G_ν the graph whose 4-block tree is the subtree of $\mathcal{T}$ rooted at ν. The already computed visibility representations $\Gamma_{\mu_1}, \ldots, \Gamma_{\mu_h}$ of $G_{\mu_1}, \ldots, G_{\mu_h}$ are suitably merged into γ_ν and an RVR Γ_ν of G_ν is obtained.

At the root we obtain an RVR of $G_\rho = G^+$. In order to turn this in an RVR of G, we must un-planarize, i.e., replace every dummy-vertex by a crossing. For this to be feasible, we need each dummy-vertex to be drawn in a special way: its rectangle needs to have one incident



(a) C_ν^- **(b)** transv.pair **(c)** $\mathcal{R}$ **(d)** $\mathcal{R}_\nu$

Figure 4 (a) The graph C_ν^- obtained from C_ν in Fig. 3c. (b) A transversal pair of bipolar orientations for C_ν^-. (c) The corresponding rectangular dual representation. (d) Retracting the rectangles and adding the surround-edge.

edge on each side. We call this the *4-sides-condition*. Even then converting dummy-vertices into crossings is non-trivial, since the four edges on the four sides may not be suitably aligned. To achieve such an alignment, we use the so-called "zig-zag-bend-elimination-slide" [3], with which we can transform parts of the drawing until edges are aligned and hence each dummy-vertex can be turned into a crossing point. Since this introduces no other crossings we obtain an RVR, and deleting the added edges gives the desired RVR of G.

3.4.3 Surround-edges and drawing 4-connected components

Consider a 4-connected component C_ν. We want to find an RVR of C_ν that satisfies the 4-sides-condition. To be able to merge later, it must satisfy one more property: We pick a so-called *surround-edge* e on the outer face beforehand, and we want an RVR such that all edges incident to an endpoint of e are drawn horizontally.

▶ **Observation 8.** *If G^+ has no B-configuration, W-configuration or T-configuration, then there exists an edge e on the outer face $\{u, v, w\}$ of C_ν such that the vertex facing e from inside C_ν is an original vertex. Here, the* vertex facing e from the inside *is the third vertex on the face f_e of G^+ that is incident to e and inside $\{u, v, w\}$.*

If more than one edge of the outer face of C_ν satisfies the condition of Observation 8, then we chose the surround-edge arbitrarily with one exception: If ν has a parent π, and the surround-edge of C_π is one of the outer-face edges of C_ν, then we use the same edge also as surround-edge of C_ν. Note that whether an edge can be used as surround-edge depends on its incident face in G^+ (not C_π), and so the surround-edge for C_π, if it exists in C_ν, also qualifies as surround-edge for C_ν. For example, for graph C_ϕ in Fig. 3c, edge (a, b) must be chosen as surround-edge, because both (b, c) and (a, c) have a dummy-vertex as third vertex on the face. Since edge (a, b) also belongs to C_ω, it becomes the surround-edge of C_ω as well.

▶ **Lemma 9.** *C_ν has an RVR such that every inner vertex of degree 4 satisfies the 4-sides-condition and all edges incident to an endpoint of the surround-edge of C_ν are drawn horizontally.*

Proof Sketch. If C_ν is K_4, then such an RVR is easily obtained (see also Fig. 7b), so we assume that C_ν has at least 5 vertices. Remove the surround-edge from C_ν to obtain graph C_ν^-. Since C_ν is 4-connected, C_ν^- is internally 4-connected. Such a graph is known to have a *transversal pair of bipolar orientations*, which directs and colors each inner edge either

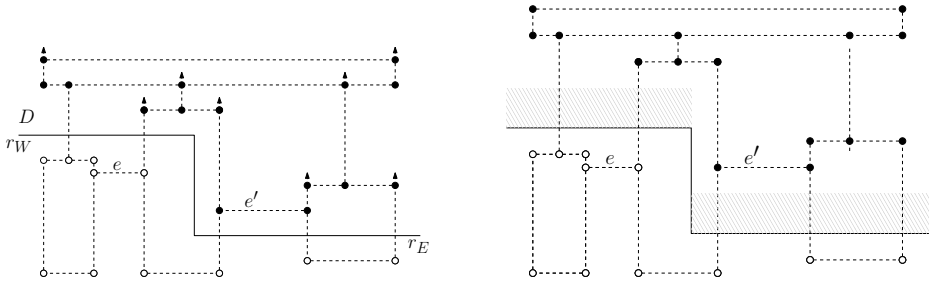


Figure 5 A zig-zag-slide (adapted from [3]). Black points move upward while white points remain stationary. With this amount of sliding, edges e and e' become aligned.

red or blue[1] such that the following holds: (i) For each inner vertex v, the clockwise order of edges around v contains outgoing blue edges, then outgoing red edges, then incoming blue edges, then incoming red edges, and none of these sets is empty. (ii) If v_N, v_E, v_S, v_W are the four vertices on the outer face in clockwise order, then all inner edges incident to $v_N/v_E/v_S/v_W$ are blue incoming/red incoming/blue outgoing/red outgoing. See Fig. 4b. Such a transversal pair of bipolar orientations always exists and can be computed in linear time [12, 15]. Their proof allows us to specify v_E and v_W to be the ends of the surround-edge, and to color all outer face edges red. In the same papers it was also shown that from this coloring and orientation, we can obtain a *rectangular dual representation*, i.e., an assignment of interior-disjoint rectangles to vertices such that the union of these rectangles is a rectangle without holes. For each edge (v, w), the rectangles of v and w share a positive-length part of their boundaries. Moreover, if the edge is directed $v \to w$, then the rectangle of v is left (below) w if and only if the edge is red (blue). See Fig. 4c.

We can now obtain an RVR of C'_ν easily: use the rectangular dual representation of C'_ν, and retract the rectangles a bit. The 4-sides-condition holds since rectangles of inner vertices have neighbors on all sides. Since outer face edges are red and the surround-edge was (v_W, v_E), all incident edges of v_W and v_E are horizontal. Finally we add the surround-edge after lengthening the rectangles of v_W, v_E and obtain the desired RVR. See Fig. 4d. ◀

3.4.4 Zig-zag-slides

Both for the merging step and for un-doing the planarization, we need a method of modifying the drawing such that some items are moved while others are stationary. This can be achieved with a zig-zag-bend-elimination-slide (or zig-zag-slide for short) [3]. Assume that Γ is an RVR and we have a dividing curve D as follows: D consists of some vertical line segment s that intersects no horizontal element of the drawing (i.e., neither a horizontal edge nor a horizontal boundary of a rectangle). At the top end of s, attach a leftward horizontal ray r_W. At the bottom end of s, attach a rightward horizontal ray r_E. No conditions are being put onto r_W and r_E. Define the *region above* D be all points that are in the x-range of r_W and strictly above r_W, and all points in the x-range of r_E and on or above r_E.

Now slide all points in the region above D upwards by some amount $\delta > 0$. This maintains a visibility representation, after lengthening vertical edges and rectangle borders, since no horizontal segments cross s. See Fig. 5. Similar define zig-zag-slides for the three other shapes of D achieved by flipping the picture horizontally and/or rotating 90°.

[1] In all figures of the paper red edges appear in lighter gray than blue edges when printed b/w.

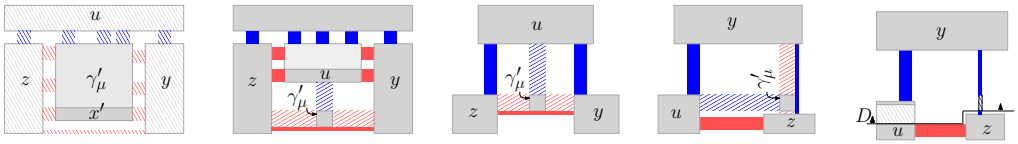


Figure 6 (a) Illustration of γ_μ and γ'_μ. (b) Illustration for the proof of Lemma 10. (c–e) Illustration for the proof of Lemma 11. (c) u has two vertical edges. (d) y has two vertical edges, u has a higher top than z. (e) Slide so that u has a higher top than z.

3.4.5 Merging components

Let μ be a child of ν in $\mathcal{T}$, and recall that we already obtained drawings γ_ν of C_ν and (recursively) γ_μ of the graph G_μ consisting of C_μ and the components at its descendants. The common vertices $\{u, y, z\}$ of G_μ and C_ν form the outer face of G_μ. See Fig. 3c. The outer face of C_μ^-, consists of $\{u, y, z\}$ as well as a fourth vertex, say x'; after possible renaming of $\{u, y, z\}$ we may assume that the order around the outer face of C_μ^- is u, y, x', z. In particular, drawing γ_μ contains (up to rotation) node u on the top, node y on the right, the surround-edge (y, z) and node x' at the bottom, and node z on the left. Let γ'_μ be the drawing obtained from γ_μ by deleting u, y, z. It suffices to merge γ'_μ, since u, y, z and the edges between them are represented in γ_ν already. See Fig. 6a.

Triangle $\{u, y, z\}$ is an inner face of C_ν. The challenge is now to find a region inside where this face is drawn in γ_ν into which γ'_μ will "fit". Put differently, we want to find a region within γ_ν inside the face $\{u, y, z\}$ that is (after possible rotation) below u, to the left of y and to the right of z. We call such a region, a *feasible region* for γ_μ in γ_ν. As we have no control over how triangle $\{u, y, z\}$ is drawn in γ_ν, the existence of a feasible region is non-trivial, and may in fact require modifying γ_ν slightly. We start with a special case that will be needed later:

▶ **Lemma 10.** *Assume that the surround-edge of C_ν also belongs to C_μ. Then there exists a feasible region for γ_μ in γ_ν.*

Proof. If the surround-edge e of C_ν belongs to C_μ, then by our choice of surround-edge we also used e as the surround-edge for C_μ. That is, (y, z) is the surround-edge for both C_μ and C_ν. In γ_ν, (y, z) was drawn (after possible rotation) bottommost, with u above it. A feasible region is now found in the small strip above the drawing of edge (y, z). See Fig. 6b. ◀

▶ **Lemma 11.** *A feasible region for γ_μ in γ_ν always exists, possibly after modifying γ_ν without changing angles or incidences.*

Proof. We already argued this for the case where $\{u, y, z\}$ includes the surround-edge e of C_ν, so assume that this is not the case. Then face $\{u, y, z\}$ of C_ν is also a face of $C'_\nu = C_\nu - e$. The drawing of C'_ν was obtained via the transversal pair of bipolar orientations of C'_ν, with red/blue edges corresponding to horizontal/vertical edges. It is well-known that any inner face in a transversal pair of bipolar orientations has at least one red and at least one blue edge. So $\{u, y, z\}$ is drawn with at least one horizontal and at least one vertical edge, say two edges are vertical and one is horizontal (the other case is similar). Now consider which vertex of $\{u, y, z\}$ is the one where both incident edges of triangle $\{u, y, z\}$ are vertical.

If vertex u has two incident vertical edges (u, y) and (u, z), then (y, z) is drawn horizontally. We can then find a feasible region within the thick line of sight of edge (y, z). See Fig 6c.

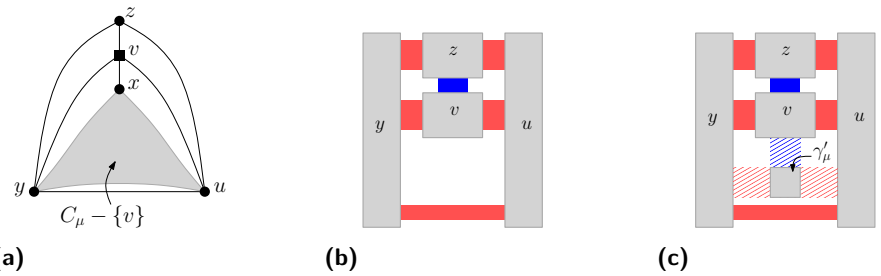


Figure 7 Illustration for the proof of Observation 12.

Now assume that y has two incident vertical edges (y, u) and (y, z) (the case of vertex z is symmetric). For ease of description, assume that y is above u and z, and u is to the left of z. If the top side of u is strictly above the top side of z, then it is easy to find a feasible region within the thick line of sight of edge (y, z) into which we can merge γ'_ν after rotating it; see Fig. 6d. If the top side of u is below (or at the same height) of the top side of z, then we modify the drawing to create a suitable region using a zig-zag-slide. The dividing curve D is defined as follows. Insert a vertical edge just left of z, ranging from just above (u, z) (hence below the top of u since we have thick lines of visibility) to just above the top of z. Expand it with a horizontal ray leftward from the lower end and a horizontal ray rightward from the upper end. By sliding upward sufficiently far, we hence achieve that the top of u is above the one of z, which allows us to find a feasible region as previously. See Fig 6e. ◀

By Lemma 11, for every child μ of ν we can find a feasible region into which we merge (after suitable scaling) the drawing $\gamma_\mu - \{u, y, z\}$. This gives the RVR of $C_\nu \cup G_\mu$. It remains to prove the 4-sides-condition for an inner vertex v of $C_\nu \cup G_\mu$. This holds clearly if v was an inner vertex of either C_ν or G_μ. If it is neither, then one can argue that v is involved in two separating triangles and with our choice of surround-edge the merging ensures the 4-sides-condition. See Fig. 7 and [2] for details. So we have:

▶ **Observation 12.** *In the RVR of $C_\nu \cup G_\mu$, the 4-sides-condition holds for every dummy-vertex that is an inner vertex of $C_\nu \cup G_\mu$.*

3.4.6 Undoing the planarization

Once all merging is completed, the drawing Γ_ρ at the root gives an RVR of G^+. To turn this into an RVR of G, we must undo the planarization, hence remove the dummy-vertices and replace them with a crossing. Since dummy-vertices are surrounded by a kite, none of them is on the outer face of G^+. So any dummy-vertex z is an inner vertex of G^+ and its rectangle R_z satisfies the 4-sides-condition. Let e_W, e_N, e_E and e_S be the four edges incident at the west/north/east and south side of R_z. If e_W and e_E have the same y-coordinate, and e_N and e_S have the same x-coordinate, then we can simply remove R_z, extend the edges, and obtain the desired crossing.

So assume that e_W and e_E have different y-coordinates, with (say) the y-coordinate y_W of e_W larger than the y-coordinate y_E of e_E. Construct a dividing curve D by starting with a vertical line segment inside R_z, ranging from y-coordinate $y_W + \varepsilon$ to y-coordinate $y_E - \varepsilon$, where $\varepsilon > 0$ is chosen small enough that the segment is inside R_z. Attach a leftward horizontal ray at the top and a rightward horizontal ray at the bottom. Now apply a zig-zag-slide of length $y_W - y_E$. This moves e_E upward while keeping e_W stationary, and hence aligns the

two edges. See also Fig. 5, where this is illustrated for $e_W = e$ and $e_E = e'$. With another zig-zag-slide (with the dividing curve using a horizontal line segment and vertical rays), we similarly can align e_N and e_S, if needed. After this, remove the dummy-vertex to obtain the crossing. Repeating this at all dummy-vertices, and finally deleting all added edges, gives the RVR of G, as no crossings are created except where dummy-vertices were removed. This ends the description of algorithm `1PRVDrawer`. We remark that with minor changes `1PRVDrawer` can also handle general (not 3-connected) 1-plane graphs, see [2].

▶ **Lemma 13.** *Let G be a 1-plane graph G and with no B-/W-/T-configuration as a subgraph. Then algorithm `1PRVDrawer` computes an RVR of G.*

Lemma 13 shows the sufficieny of Theorem 4.

3.5 Area, time complexity and testing

Algorithm `1PRVDrawer` does not obviously have linear run-time, since we repeatedly change the entire drawing, especially when applying zig-zag-slides. Also, coordinates may get very small when merging, resulting (after re-scaling to be integral) in a very large area. However, both of these become non-issues if we store RVRs implicitly, using the orthogonal representations of Observation 2. Using this approach, no zig-zag-slide needs to be performed, since these only change edge lengths (but not angles or adjacencies) and hence give rise to exactly the same orthogonal representation. Therefore, all that is required to merge a subgraph at a separating triangle $\{u, y, z\}$ is to find, for each of $w \in \{u, y, z\}$, the appropriate segment along the boundary of the rectangle of w in C_μ, and to merge here the list of adjacencies and angles that w has in the PSLG of C_ν. This takes constant time for w, hence constant time for merging at one separating triangle, and hence linear time overall. All other parts of the algorithm (such as finding separating triangles, computing 4-connected components, finding a transversal pair of bipolar orientations, and finding the rectangular dual representation) take linear time as well. The final conversion of the PSLG into an RVR also takes linear time, and gives linear integral coordinates. Here "linear" measures the size of the PSLG, which in our case is proportional to the number of vertices, edges, and crossings. For 1-planar graphs this is $O(n)$ since 1-planar graphs have at most $4n - 8$ edges [18]. Thus the final RVR has linear coordinates and the area is $O(n^2)$. We conclude:

▶ **Lemma 14.** *Let G be a 1-plane graph G with n vertices and with no B-/W-/T-configuration as a subgraph. Then there exists an $O(n)$-time algorithm that computes an RVR Γ of G that has $O(n^2)$ area.*

We finally sketch how to test whether a given 1-plane graph G contains any B-configuration, any W-configuration, or any T-configuration. Hong *et al.* [13] show an algorithm to detect and report every possible B-configuration and W-configuration in $O(n)$ time, where n is the number of vertices of G. Thus, in what follows we describe how to detect and report a T-configuration, assuming that G contains no B-configuration and no W-configuration. Let G^+ be the plane triangulated graph obtained from G by applying the planarization and triangulation step described in Section 3.4. Recall that G^+ contains a T-configuration if and only if G does. Suppose G^+ contains a T-configuration t with outer vertices a, b, c and inner vertices d, e, f, g, h, i, as in Fig. 2d. Since we added a kite around each crossing, vertices a, b, c induce a triangle in G^+. Thus to detect a T-configuration in G^+, we first list all triangles of G^+. The triangle is a T-configuration if and only the three faces on the inside have dummy-vertices as their third vertex, which can be tested in $O(1)$ time per triangle. The only difficulty is hence to list all triangles efficiently. If G is 3-connected, then G^+ is

simple and this can be done in $O(n)$ time (see e.g. [6]). The case when G is not 3-connected is described in [2]. This, together with Lemma 14, implies Theorem 5.

4 Variable Embedding Setting

Alam *et al.* [1] proved that if G is 3-connected, then it is possible to reroute its edges so as to remove all B-configurations and W-configurations but at most one and hence obtain a straight-line 1-planar drawing. We can show that this is not true for visibility representations. All the proofs for this section can be found in [2].

▶ **Theorem 15.** *Let $k \geq 1$ be an integer. There exists a 3-connected 1-planar graph G_k with $n = 6k$ vertices with at least $k - 1$ T-configurations in every possible 1-planar embedding.*

The following stronger corollary can also be proved.

▶ **Corollary 16.** *Let $k \geq 1$ be an integer value. There exists a 3-connected 1-planar graph G_k with $n = 6k$ vertices that does not admit any 1-planar embedding that can be represented as an RVR unless we delete at least $k = n/6$ edges.*

On the positive side, we can mirror the results of Alam *et al.* for visibility representations by increasing the connectivity one more.

▶ **Theorem 17.** *Every 4-connected 1-planar graph G admits a 1-planar embedding that can be represented as an RVR, except for at most one edge.*

▶ **Corollary 18.** *Every optimal 1-plane graph (i.e., a 1-plane graph with $4n - 8$ edges) admits an RVR, except for one edge.*

5 Conclusions and Open Problems

In this paper, we studied rectangle visibility representations of non-planar graphs with a fixed embedding. We showed that we can test in polynomial time whether a graph has such a representation. Of special interest to us were 1-planar graphs; here we can give a linear-time algorithm to test the existence of visibility representations if the embedding is fixed. Moreover, in case of a negative answer the algorithm provides a witness in form of either a B-configuration, W-configuration, or T-configuration. We also briefly studied 1-planar graphs without fixed embeddings.

The most pressing open problem is whether we can restrict the drawings less and still test for the existence of visibility representations? Most importantly, if we fix the rotational scheme and outer face, but *not* order in which crossing occur (and perhaps not even which edges cross), is it possible to test whether a visibility representation respecting the rotational scheme and outer face exists? The NP-hardness proof of Shermer [20] does not hold if the rotational scheme is fixed, since it uses a reduction from linear-arboricity-2, and fixing the rotational scheme would severely restrict the possible ways of splitting a graph into two linear forests. The orthogonal representation approach utterly fails if the order of crossings is not fixed, since it requires planarization as a first step.

Secondly, can we characterize for more graph classes exactly when they have a rectangle visibility representation? Notice that Lemma 6 does not use 1-planarity, and hence gives a necessary condition for any subgraph of a graph G (with a fixed embedding) to have an RVR. A second necessary condition stems from that as soon as edges may have 2 or more crossings, the edge-parts between the crossings may be non-trivial and have cycles; clearly

the graph formed by these crossings must be bipartite if G has a visibility representation. Are these two conditions sufficient, and if not, can we find necessary and sufficient conditions, at least for some restricted graph classes such as 2-planar and fan-planar graphs?

References

1 Md. Jawaherul Alam, Franz J. Brandenburg, and Stephen G. Kobourov. Straight-line grid drawings of 3-connected 1-planar graphs. In *GD 2013*, volume 8242 of *LNCS*, pages 83–94. Springer, 2013.

2 Therese Biedl, Giuseppe Liotta, and Fabrizio Montecchiani. On visibility representations of non-planar graphs. *CoRR*, abs/1511.08592, 2015. URL: `http://arxiv.org/abs/1511.08592`.

3 Therese Biedl, Anna Lubiw, Mark Petrick, and Michael J. Spriggs. Morphing orthogonal planar graph drawings. *ACM Trans. Algorithms*, 9(4):29, 2013. `doi:10.1145/2500118`.

4 Prosenjit Bose, Alice M. Dean, Joan P. Hutchinson, and Thomas C. Shermer. On rectangle visibility graphs. In *GD 1996*, volume 1190 of *LNCS*, pages 25–44. Springer, 1997.

5 Franz J. Brandenburg. 1-Visibility representations of 1-planar graphs. *J. Graph Algorithms Appl.*, 18(3):421–438, 2014.

6 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

7 Sabine Cornelsen and Andreas Karrenbauer. Accelerated bend minimization. *J. Graph Algorithms Appl.*, 16(3):635–650, 2012.

8 Alice M. Dean, William S. Evans, Ellen Gethner, Joshua D. Laison, Mohammad Ali Safari, and William T. Trotter. Bar k-visibility graphs. *J. Graph Algorithms Appl.*, 11(1):45–59, 2007.

9 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice-Hall, 1999.

10 Pierre Duchet, Yahya Ould Hamidoune, Michel Las Vergnas, and Henry Meyniel. Representing a planar graph by vertical lines joining different levels. *Discrete Math.*, 46(3):319–321, 1983.

11 William S. Evans, Michael Kaufmann, William Lenhart, Tamara Mchedlidze, and Stephen K. Wismath. Bar 1-visibility graphs vs. other nearly planar graphs. *J. Graph Algorithms Appl.*, 18(5):721–739, 2014.

12 Éric Fusy. Transversal structures on triangulations: A combinatorial study and straight-line drawings. *Discrete Math.*, 309(7):1870–1894, 2009.

13 Seok-Hee Hong, Peter Eades, Giuseppe Liotta, and Sheung-Hung Poon. Fáry's theorem for 1-planar graphs. In *COCOON 2012*, volume 7434 of *LNCS*, pages 335–346. Springer, 2012.

14 Goos Kant. A more compact visibility representation. *Int. J. Comput. Geometry Appl.*, 7(3):197–210, 1997.

15 Goos Kant and Xin He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theor. Comput. Sci.*, 172(1-2):175–193, 1997.

16 Goos Kant, Giuseppe Liotta, Roberto Tamassia, and Ioannis G. Tollis. Area requirement of visibility representations of trees. *Inf. Process. Lett.*, 62(2):81–88, 1997.

17 Ralph H. J. M. Otten and J. G. Van Wijk. Graph representations in interactive layout design. In *IEEE ISCSS*, pages 914–918. IEEE, 1978.

18 János Pach and Gáza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

19 Pierre Rosenstiehl and Robert Endre Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discr. & Comput. Geom.*, 1:343–353, 1986.

20 Thomas C. Shermer. On rectangle visibility graphs. III. External visibility and complexity. In *CCCG 1996*, pages 234–239. Carleton University Press, 1996.

21 Ileana Streinu and Sue Whitesides. Rectangle visibility graphs: Characterization, construction, and compaction. In *STACS 2003*, volume 2607 of *LNCS*, pages 26–37. Springer, 2003.

22 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM J. Computing*, 16(3):421–444, 1987.

23 Roberto Tamassia and Ioannis G. Tollis. A unified approach to visibility representations of planar graphs. *Discr. & Comput. Geom.*, 1(1):321–341, 1986.

24 Carsten Thomassen. Plane representations of graphs. In *Progress in Graph Theory*, pages 43–69. AP, 1984.

25 Carsten Thomassen. Rectilinear drawings of graphs. *J. Graph Theory*, 12(3):335–341, 1988.

26 Stephen K. Wismath. Characterizing bar line-of-sight graphs. In *SoCG 1985*, pages 147–152. ACM, 1985.

Delaunay Triangulations on Orientable Surfaces of Low Genus*

Mikhail Bogdanov[1], Monique Teillaud[2], and Gert Vegter[3]

1 **Inria, Centre de recherche Sophia Antipolis – Méditerranée, France**
2 **Inria, Centre de recherche Nancy – Grand Est. CNRS, Loria, Université de Lorraine, France**
3 **Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, The Netherlands**

Abstract

Earlier work on Delaunay triangulation of point sets on the 2D flat torus, which is locally isometric to the Euclidean plane, was based on lifting the point set to a locally isometric 9-sheeted covering space of the torus [28, 20, 12, 11]. Under mild conditions the Delaunay triangulation of the lifted point set, consisting of 9 copies of the input set, projects to the Delaunay triangulation of the input set.

We improve and generalize this work. First we present a new construction based on an 8-sheeted covering space, which shows that eight copies suffice for the standard flat torus. Then we generalize this construction to the context of compact orientable surfaces of higher genus, which are locally isometric to the hyperbolic plane. We investigate more thoroughly the Bolza surface, homeomorphic to a sphere with two handles, both because it is the hyperbolic surface with lowest genus, and because triangulations on the Bolza surface have applications in various fields such as neuromathematics and cosmological models [32, 3, 15]. While the general properties (existence results of appropriate covering spaces) show similarities with the results for the flat case, explicit constructions and their proofs are much more complex, even in the case of the apparently simple Bolza surface. One of the main reasons is the fact that two hyperbolic translations do not commute in general.

To the best of our knowledge, the results in this paper are the first ones of this kind. The interest of our contribution lies not only in the results, but most of all in the construction of covering spaces itself and the study of their properties.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases covering spaces, hyperbolic surfaces, finitely presented groups, Fuchsian groups, systole

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.20

1 Introduction

Delaunay triangulations of the flat cubic torus [39], i.e., the flat torus for which the fundamental domain is a cube, are used in many fields of science, ranging from the infinitely small, with "nuclear pasta" [33] to the infinitely large, with cosmology [35, 36, 40, 24] (more references can be found in [12, 13, 14, 10]). They are needed either because the objects studied are periodic in space, or because they allow for modeling periodic boundary conditions, e.g.,

* Work partially supported by the Inria OrbiCG associate team `http://www.loria.fr/~teillaud/collab/OrbiCG`


© Mikhail Bogdanov, Monique Teillaud and Gert Vegter;
licensed under Creative Commons License CC-BY

in fluid mechanics. Periodic Delaunay triangulations in the hyperbolic plane are also used in diverse fields, such as physics, solid modeling, cosmological models, neuromathematics [32, 31, 3, 15]; they can also be seen as triangulations on hyperbolic surfaces.

A closed orientable flat (resp. hyperbolic) surface M is isometric to a quotient $\mathbb{E}^2/\Gamma$ (resp. $\mathbb{H}^2/\Gamma$), where Γ is a discrete group of isometries acting on the Euclidean plane $\mathbb{E}^2$ (resp. hyperbolic plane $\mathbb{H}^2$), with properties made precise later in the paper.

As in [14], we stick to the standard definition of a *triangulation* [30, Section 9.1, condition (3)] of a topological space $\mathbb{Y}$ as a geometric simplicial complex K such that $K = \bigcup_{\sigma \in K} \sigma$ is homeomorphic to $\mathbb{Y}$. A triangulation defined by a point set $\mathcal{P}$ is a triangulation whose set of vertices is $\mathcal{P}$. The Delaunay triangulation of a point set $\mathcal{P}$ in $\mathbb{E}^2$ (resp. $\mathbb{H}^2$) is a triangulation of the convex hull (resp. hyperbolic convex hull) of $\mathcal{P}$ that can be defined uniquely even in degenerate cases [19].

The Delaunay triangulation of points on a 2D flat torus was usually obtained from a triangulation of the convex hull of nine periodic copies of the points in the plane $\mathbb{E}^2$, laid in a 3x3 pattern [28, 22] (or 27 copies in a 3x3x3 pattern in the 3D case [20]). A more recent algorithm uses covering spaces instead, and computes the Delaunay triangulation directly in the torus whenever possible [12], thus providing all adjacency relations between simplices. It led to CGAL packages [27, 11], which have already been used in various fields.[1] In that work, the Delaunay triangulation of a surface M is defined as follows:

▶ **Definition 1.** The *Delaunay triangulation* $DT_M(\mathcal{P})$ of M defined by a finite point set $\mathcal{P}$ in $\mathbb{E}^2$ (resp. $\mathbb{H}^2$) is the projection onto M of the Delaunay triangulation $\mathrm{DT}_{\mathbb{E}}(\Gamma\mathcal{P})$ (resp. $\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P})$) of the infinite point set $\Gamma\mathcal{P}$, if it is a triangulation.

Note that the fact that Γ is discrete guarantees that the triangulation of $\Gamma\mathcal{P}$ in $\mathbb{E}^2$ (resp. $\mathbb{H}^2$) is locally finite [14]. We elaborate a bit on the projection here and check that it is well-defined. Let the projection be denoted as $\pi : \mathbb{E}^2$(resp. $\mathbb{H}^2$) $\to M$. By definition, for any given point $p \in \mathcal{P}$, all elements of the orbit $\Gamma p = \{g(p) \mid g \in \Gamma\}$ project by π to the same point $\pi(p)$ of M. We can assume that no two points of $\mathcal{P}$ lie on the same Γ-orbit: otherwise we can always choose a proper subset $\mathcal{P}' \subset \mathcal{P}$, so that $\Gamma\mathcal{P}' = \Gamma\mathcal{P}$. The Delaunay triangulation $\mathrm{DT}_{\mathbb{E}}(\Gamma\mathcal{P})$ (resp. $\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P})$) is a partition of $\mathbb{E}^2$ (resp. $\mathbb{H}^2$) by vertices, edges, and triangular faces. Each $g \in \Gamma$ is an isometry, so the empty disk circumscribing a Delaunay triangle $(p, q, r), p, q, r \in \mathbb{E}^2$(resp. $\mathbb{H}^2$) is transformed by g into the disk circumscribing triangle $(g(p), g(q), g(r))$, which is also empty. If the set $\Gamma\mathcal{P}$ contains degeneracies, the Delaunay triangulation of a polygon formed by cocircular points $P = (p_0, p_1, \ldots, p_k = p_0), k \geq 4$ is not uniquely defined. For any $g \in \Gamma$, the polygon $g(P) = (g(p_0), g(p_1), \ldots, g(p_k) = g(p_0))$ is also formed by cocircular points. Once a triangulation of P is chosen, the polygon $g(P)$ can be triangulated by the image by g of the triangulation of P. Then, these two triangulations project under π to the same set of triangles on M. Since π is surjective, the projection $\pi(\mathrm{DT}_{\mathbb{E}}(\Gamma\mathcal{P}))$ (resp. $\pi(\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P}))$) forms a partition of M by vertices, edges, and triangular faces.[2] This discussion allows us to ignore degeneracies in the sequel.

The algorithm follows the classic incremental algorithm in $\mathbb{E}^d$ [7]: for each new point p, the conflicting simplices (i.e., simplices whose circumscribing balls contain p) are deleted, and the region that they formed is then triangulated with new simplices with apex p. That

[1] See `http://www.cgal.org/projects.html` for some identified users.

[2] In the case of the flat torus, this can automatically be achieved by means of a symbolic perturbation scheme [19] that ensures that the triangulation in the universal covering space is uniquely defined and invariant under the action of the group.

algorithm relies on the fact that the union of the set of conflicting simplices is always a topological ball, which subsumes that the graph of edges of the triangulation has no cycle of length one *(1-cycles)* or two *(2-cycles)*. However, the graph of edges of the projection of the Delaunay triangulation on a manifold can have non-trivial cycles: for instance, in the extreme case when the input consists of only one point p, then the set Γp is the orbit of p under Γ, so all Delaunay vertices project to the same point, i.e., Delaunay edges project to a small set of 1-cycles on the manifold, all with the same vertex. The existence of such 1- or 2-cycles can be checked using a geometric criterion. The algorithm computes in some finite-sheeted covering manifold (i.e., roughly speaking, using "copies" of the input points) satisfying the criterion: a 9- (resp 27-) sheeted covering manifold is sufficient for the 2D (resp. 3D) flat torus. When sufficiently many points have been inserted so that the criterion is satisfied on the initial manifold, the computation continues in this manifold, i.e., without using more copies of points.

To the best of our knowledge, there are no known similar results on hyperbolic surfaces. Intuition is challenged there, in particular because the group Γ is non-Abelian in general.

In this paper, we first propose a construction of a family of 2^k sheeted-covering spaces, $k > 0$, of the flat cubic torus (Section 2). We show how this construction allows us to improve the computation of a Delaunay triangulation of that torus by reducing the number of sheets of the covering spaces.

For any closed hyperbolic surface M, we prove in Section 3 the existence of a finite-sheeted covering space $\widetilde{M}$ of M so that the projection of the Delaunay triangulation in $\mathbb{H}^2$ onto $\widetilde{M}$ has no 1- or 2-cycle (Proposition 10). We also present a geometric criterion to checks that 1- or 2-cycles don't exist (Proposition 9). When such a covering space $\widetilde{M}$ can be constructed, then the algorithm previously proposed for closed Euclidean manifolds [14] naturally extends to a set of points $\mathcal{P}$ on the hyperbolic surface M.

The Bolza surface $\mathcal{M}$ is the simplest possible surface after the 2D Euclidean torus: it is homeomorphic to a double torus. The main contribution of this paper (Section 4) is a construction of some finite-sheeted covering spaces of $\mathcal{M}$, which allows us to concretely exhibit a 4-sheeted covering space in which the projection of a Delaunay triangulation never has a 1-cycle, and a 128-sheeted covering space in which it has no 2-cycle. These results mostly are of theoretical interest for the incremental algorithm above, due to the large number of points that would need to be considered for the computations; we propose a more practical algorithm in the last section.

2 Some finitely-sheeted covering spaces of the flat torus

Let a, b denote two orthogonal translations of the same length. We denote as $\mathcal{G}_1$ the group $< a, b >$ generated by a and b. Since $\mathcal{G}_1$ is Abelian, any $x \in \mathcal{G}_1$ can be uniquely written as $x = a^\alpha b^\beta$, where $\alpha, \beta \in \mathbb{Z}$. The *length* $\lambda_1(x)$ of x can thus be defined as the sum $|\alpha| + |\beta|$. We are going to study some covering spaces of the flat torus $\mathbb{T} = \mathbb{T}^2_1 = \mathbb{E}^2/\mathcal{G}_1$.

The minimal distance by which $\mathcal{G}_1$ moves a point of $\mathbb{E}^2$ is $\delta(\mathcal{G}_1) = \|a\| = \|b\|$, where $\|\cdot\|$ denotes the Euclidean norm. Any closed square of edge length $l = \delta(\mathcal{G}_1)$ whose edges are parallel to a and b is a fundamental domain of $\mathcal{G}_1$.

In the sequel, the inverse x^{-1} of an element x of $\mathcal{G}_1$ will alternatively be denoted by $\overline{x}$. All subgroups are normal, since $\mathcal{G}_1$ is Abelian. Let us consider the subgroup $\mathcal{G}_2$ consisting of elements of $\mathcal{G}_1$ of even length. It is easy to see that $\mathcal{G}_2 = \langle \overline{a}b, ab \rangle$.

▶ **Lemma 2.** *$\mathcal{G}_2$ is a subgroup of index 2 in $\mathcal{G}_1$.*

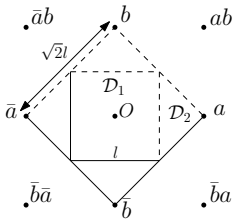


Figure 1 The original domains $\mathcal{D}_1$ and $\mathcal{D}_2$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively.

Proof. Let $\varphi : \mathcal{G}_1 \to \mathbb{Z}_2$ be the group homomorphism defined by $\varphi(x) = \lambda_1(x) \mod 2$. The subgroup $\mathcal{G}_2$ is the kernel of φ. According to the First Isomorphism Theorem (see, e.g., [1]) $\ker\varphi$ is a normal subgroup of $\mathcal{G}_1$, and $\mathcal{G}_1/\ker\varphi \cong \varphi(\mathcal{G}_1)$. Therefore, $\mathcal{G}_1/\mathcal{G}_2 \cong \mathbb{Z}_2$. ◀

For $k > 1$, if $\mathcal{G}_{2^{k-1}}$ is a subgroup of index 2^{k-1} of $\mathcal{G}_1$ generated by two elements, i.e., $\mathcal{G}_{2^{k-1}} = \langle g, h\rangle\,,\ g, h \in \mathcal{G}_1$, we construct the subgroup $\mathcal{G}_{2^k}$ of $\mathcal{G}_{2^{k-1}}$ as follows: $\mathcal{G}_{2^k} = \langle \overline{g}h, gh\rangle\,.$ $\mathcal{G}_{2^k}$ consists of elements of even length of $\mathcal{G}_{2^{k-1}}$. The proof of the following lemma is similar to that of Lemma 2.

▶ **Lemma 3.** *For any $k > 0$, $\mathcal{G}_{2^k}$ is a subgroup of index 2^k in $\mathcal{G}_1$.*

By construction, for any $k \geq 0$, the generators of $\mathcal{G}_{2^k}$ are two translations whose vectors a_k and b_k are orthogonal and of equal length $\delta(\mathcal{G}_{2^k}) = \sqrt{2}^k l$. The orbits of $\mathcal{G}_{2^k}$ are isomorphic to $\mathbb{Z}^2$. The half-open square $\left\{t_a a_k + t_b b_k \mid t_a, t_b \in [-\frac{1}{2}, \frac{1}{2})\right\}$ contains exactly one representative of each orbit. We call it the *original domain* of $\mathcal{G}_{2^k}$ and denote it as $\mathcal{D}_{2^k}$. See Figure 1; for simplicity, x denotes both an element x of $\mathcal{G}_1$ and the image xO of the origin O by x. The closure of the original domain $\mathcal{D}_{2^k}$ is a fundamental domain of $\mathcal{G}_{2^k}$; it is also the cell of O in the Voronoi diagram of its orbit $\mathcal{G}_{2^k}O$.

For any $k > 0$, $\mathbb{T}^2_{2^k} = \mathbb{E}^2/\mathcal{G}_{2^k}$ is a flat torus, with the corresponding projection map $\pi_{2^k} : \mathbb{E}^2 \to \mathbb{T}^2_{2^k}$. The torus $\mathbb{T}^2_{2^k}$ is a covering space of $\mathbb{T}^2$ together with the covering map $\rho_k = \pi_1 \circ \pi_{2^k}^{-1}$. Since the index of $\mathcal{G}_{2^k}$ in $\mathcal{G}_1$ is 2^k, we get the following result:

▶ **Proposition 4.** *For any $k > 0$, $\mathbb{T}^2_{2^k}$ is a 2^k-sheeted covering space of $\mathbb{T}^2$.*

Delaunay triangulations via 2^k-sheeted covering spaces, $k > 0$

As shown in [12] the Delaunay triangulation $DT_{\mathbb{T}}(\mathcal{P})$ as defined in introduction (Def. 1) does not exist for each set of points $\mathcal{P} \in \mathbb{E}^2$, because $\pi(DT(\mathcal{GP}))$ can have 1- or 2-cycles. However, there are always some covering spaces of the torus in which the Delaunay triangulation exists and can be computed.

Let $\Delta(\mathcal{S})$ denote the diameter of the largest disk in $\mathbb{E}^2$ that does not contain any point of a set $\mathcal{S}$ in its interior. Note that for any $p \in \mathbb{E}^2, \Delta(\mathcal{G}_{2^k}p) = \sqrt{2}^{k+1} l$.

▶ **Proposition 5** ([12])**.** *If $\Delta(\mathcal{GP}) < \frac{\delta(\mathcal{G})}{2}$, then for any finite $\mathcal{P}' \supseteq \mathcal{P}$, $\pi(DT(\mathcal{GP}'))$ is a triangulation of $\mathbb{T}^2$.*

▶ **Proposition 6.** *If $\Delta(\mathcal{G}_1\mathcal{P}) < \frac{1}{2}\delta(\mathcal{G}_{2^k})$, then $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{P} \cup \mathcal{G}_{2^k}\mathcal{T}))$ is a triangulation of $\mathbb{T}^2_{2^k}$ for any finite point set $\mathcal{T}$ in $\mathbb{E}^2$. In particular, then $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{P}'))$ is a triangulation of $\mathbb{T}^2_{2^k}$ for any finite $\mathcal{P}' \supseteq \mathcal{P}$, $\mathcal{P}' \in \mathbb{E}^2$.*

Proof. Let $\mathcal{P}_{2^k}$ denote the set $\mathcal{G}_1\mathcal{P} \cap \mathcal{D}_{2^k}$. By Proposition 5, if $\Delta(\mathcal{G}_{2^k}\mathcal{P}_{2^k}) < \frac{1}{2}\delta(\mathcal{G}_{2^k})$, then $\pi_{2^k}(DT(\mathcal{G}_{2^k}\mathcal{P}_{2^k} \cup \mathcal{G}_{2^k}\mathcal{T}))$ is a triangulation. Let us note that $\mathcal{G}_{2^k}\mathcal{P}_{2^k} = \mathcal{G}_1\mathcal{P}$.

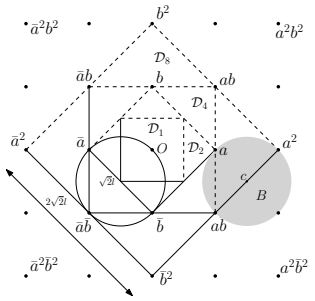


Figure 2 The original domain $\mathcal{D}_8$. Maximum empty disk with diameter $\Delta(\mathcal{G}_1 O)$.

For $\mathcal{P}' \supseteq \mathcal{P}$, $\mathcal{G}_1\mathcal{P}' = \mathcal{G}_1\mathcal{P} \cup \mathcal{G}_1(\mathcal{P}' \setminus \mathcal{P}) = \mathcal{G}_1\mathcal{P} \cup \mathcal{G}_{2^k}(\mathcal{G}_1(\mathcal{P}' \setminus \mathcal{P}) \cap \mathcal{D}_{2^k})$. It remains to take $\mathcal{T} = \mathcal{G}((\mathcal{P}' \setminus \mathcal{P}) \cap \mathcal{D}_{2^k})$. ◀

By Proposition 6, if the maximum empty disk diameter $\Delta(\mathcal{G}_1\mathcal{P})$ is smaller than $\frac{1}{2}\delta(\mathcal{G}_4) = l$, then $\pi_4(DT(\mathcal{G}_1\mathcal{P} \cup \mathcal{G}_4\mathcal{T})))$ is a simplicial complex for any finite $\mathcal{T}$ in $\mathbb{E}^2$. If it is smaller than $\frac{1}{2}\delta(\mathcal{G}_2) = \frac{\sqrt{2}}{2}l$, then $\pi_2(DT(\mathcal{G}_1\mathcal{P} \cup \mathcal{G}_2\mathcal{T})))$ is a simplicial complex.

Note that, for $k > 3$, $\frac{1}{2}\delta(\mathcal{G}_{2^k}) = \frac{1}{2}\sqrt{2}^k l > \sqrt{2}l = \Delta(\mathcal{G}_1 p)$, for any $p \in \mathcal{P}$. Therefore, by Proposition 6, $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{P}))$ is a simplicial complex. Let us now consider the case $k = 3$.

▶ **Corollary 7.** *For any finite set $\mathcal{P} \subset \mathcal{D}_1$ of $n > 1$ points, $\pi_8(DT(\mathcal{G}_1\mathcal{P}))$ is a simplicial complex.*

Proof. The maximum empty disk diameter $\Delta(\mathcal{G}_1 p)$, for any $p \in \mathcal{P}$, is $\sqrt{2}l$, that is equal to $\frac{1}{2}\delta(\mathcal{G}_8)$ (See Figure 2). We are going to show that $\Delta(\mathcal{G}_1\mathcal{P})$ is strictly less than $\sqrt{2}l$. Let us suppose that B is a disk with diameter $\sqrt{2}l$ centered at some point $c \in \mathbb{E}^2$ and containing no point of $\mathcal{G}_1\mathcal{P}$ in its interior (See Figure 2). B contains $\mathcal{D}_1(c)$, the original domain centered at c. Since $\mathcal{D}_1(c)$ is a half-open square, there is only one point of $\mathcal{D}_1(c)$ on the boundary of B. For p and $q \neq p, q \in \mathcal{P}$, there is a representative of $\mathcal{G}_1 p$ and a representative of $\mathcal{G}_1 q$ in $\mathcal{D}_1(c)$. At least one of these representatives lies in the interior of B. This contradicts that the interior of B is empty. If we add more points, the diameter of the largest empty disk cannot become larger.

By Proposition 6, $\pi_8(DT(\mathcal{G}_1\mathcal{P}))$ is a simplicial complex. ◀

Algorithm. For a finite set of points $\mathcal{P} \subset \mathcal{D}_1$, the algorithm of [12], inspired by the standard incremental algorithm [7], computes $DT_{\mathbb{T}}(\mathcal{G}_1\mathcal{P})$ via a 9-sheeted covering space. As soon as the condition of Proposition 5 is fulfilled upon insertion of a point, it switches to computing in $\mathbb{T}^2$. Assuming that $\mathcal{P}$ contains at least two points, we modify this algorithm and use the covering spaces $\mathbb{T}^2_8, \mathbb{T}^2_4$, and $\mathbb{T}^2_2$ using Proposition 6 and Corollary 7.

- The algorithm starts by precomputing $\pi_8(DT(\mathcal{G}_1\{p, q\}))$, for any $\{p, q\} \subset \mathcal{P}$. Then it adds points one by one in the Delaunay triangulation of the 8-sheeted covering space $\mathbb{T}^2_8$.
- If after insertion of some set of points $\mathcal{S} \subseteq \mathcal{P}$, the maximum empty disk diameter becomes less than l, then $\pi_4(DT(\mathcal{G}_1\mathcal{S}'))$ is guaranteed to be a triangulation for any finite $\mathcal{S}' \supseteq \mathcal{S}$. So, we can discard all redundant periodic copies of simplices of $\pi_8(DT(\mathcal{G}_1\mathcal{S}))$ and switch to incrementally computing $\pi_4(DT(\mathcal{G}_1\mathcal{S}'))$ in the 4-sheeted covering space $\mathbb{T}^2_4$ for $\mathcal{S} \subseteq \mathcal{S}' \subseteq \mathcal{P}$.
- Similarly, if after some more insertions the maximum empty disk diameter becomes less than $\frac{\sqrt{2}}{2}l$, then we switch to incrementally computing triangulations in the 2-sheeted covering space $\mathbb{T}^2_2$.
- If it becomes less than $\frac{1}{2}l$, then we switch to computing $\pi(DT(\mathcal{G}_1\mathcal{P}))$ in $\mathbb{T}^2$.

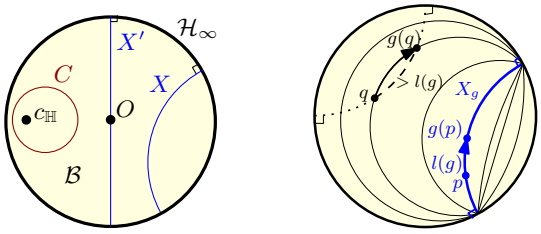


Figure 3 (left) Lines and circles. (right) Translation g of axis X_g.

For any $\mathcal{S} \subset \mathcal{P}$, $\pi_{2^k}(DT(\mathcal{G}_1\mathcal{S}))$ contains 2^k periodic copies of each point of $\mathcal{S}$. Hence, using covering spaces whose number of sheets is as small as possible improves the efficiency of the algorithm, even though the asymptotic complexity is unchanged: it stays randomized worst-case time and space optimal [12].

3 Delaunay triangulations on hyperbolic closed surfaces

3.1 Background

As the *Poincaré disk* is a conformal model of the hyperbolic plane (i.e., it preserves angles), it is widely used in applications [25, 32, 31, 3, 15]. In this model [9] [4, Chapter 19] the hyperbolic plane $\mathbb{H}^2$ is represented as the open unit disk $\mathcal{B}$ of the Euclidean plane $\mathbb{E}^2$. The points on the boundary of $\mathcal{B}$ are the *points at infinity*; their set is denoted as $\mathcal{H}_\infty = \{p \in \mathbb{E}^2 : \|p\| = 1\}$. Hyperbolic lines, or geodesics, are represented either as arcs of Euclidean circles orthogonal to $\mathcal{H}_\infty$ or as diameters of $\mathcal{B}$; a hyperbolic circle is represented as a Euclidean circle contained in $\mathcal{B}$, and its hyperbolic center is the limit point of the pencil of circles that it generates with $\mathcal{H}_\infty$ (Fig. 3(left)).

The group of isometries of $\mathbb{H}^2$ is denoted as $\mathrm{Isom}(\mathbb{H}^2)$, and the group of orientation-preserving isometries is denoted as $\mathrm{Isom}^+(\mathbb{H}^2)$. Identifying $\mathbb{E}^2$ with the complex plane $\mathbb{C}$, each $g \in \mathrm{Isom}^+(\mathbb{H}^2)$ is of the form $g(z) = \frac{\alpha z+\beta}{\overline{\beta} z+\overline{\alpha}}$, with matrix $m_g = \begin{bmatrix} \alpha & \beta \\ \overline{\beta} & \overline{\alpha} \end{bmatrix}$, $\alpha, \beta \in \mathbb{C}, |\alpha|^2 - |\beta|^2 = 1$. There are three types of orientation-preserving isometries of $\mathbb{H}^2$; in this paper we only consider *hyperbolic* isometries, also called *translations*. A translation g fixes no point of $\mathcal{B}$ and fixes two points on $\mathcal{H}_\infty$. The *axis* X_g of g is the hyperbolic line ending at the two fixed points (Fig. 3(right)). Any point is translated by g along a curve at constant distance to the axis. All such curves have the fixed points of g as infinite points. The distance by which g moves all points of X_g is the same, it is called the *translation length* of g, $l(g) = 2 \cdot \mathrm{acosh}\left(\frac{1}{2}\,\mathrm{trace}(m_g)\right)$. If $p \notin X_g$, then the distance by which g moves p is strictly greater than $l(g)$. It is worth noticing that

two translations do not commute in general.

A *Fuchsian group* is a discrete subgroup of $\mathrm{Isom}^+(\mathbb{H}^2)$ [26]. A *hyperbolic surface* is a connected 2-dimensional manifold, such that every point has a neighborhood isometric to a disk of $\mathbb{H}^2$. In this paper all hyperbolic surfaces are *closed* surfaces, i.e., they are compact.

A closed orientable hyperbolic surface M is of the form $\mathbb{H}^2/\Gamma$,
where Γ is a Fuchsian group only containing hyperbolic translations (and the identity).

(See [30, 37, 38, 41].) The projection map $\pi : \mathbb{H}^2 \to M = \mathbb{H}^2/\Gamma$ is a local isometry and a covering projection. The *Dirichlet region* $\mathcal{D}_p(\Gamma)$ for Γ centered at p is defined as the closure

$Cl(V_p(\Gamma p))$ of the open cell $V_p(\Gamma p)$ of p in the Voronoi diagram $\mathrm{VD}_{\mathbb{H}}(\Gamma p)$ of Γp in $\mathbb{H}^2$. From compactness of M follows that $\mathcal{D}_p(\Gamma)$ is a compact hyperbolic convex polygon with finitely many sides. The area of M is finite: $area(M) = 4\pi(genus(M) - 1)$. Each Dirichlet region $\mathcal{D}_p(\Gamma)$ is a *fundamental domain* for the action of Γ on $\mathbb{H}^2$, i.e., *(i)* $\bigcup_{g\in\Gamma} g(\mathcal{D}_p(\Gamma)) = \mathbb{H}^2$, and *(ii)* $\forall g_1, g_2 \in \Gamma, g_1 \neq g_2, g_1(Int(\mathcal{D}_p(\Gamma))) \cap g_2(Int(\mathcal{D}_p(\Gamma))) = \emptyset$.

The *systole* $sys(M)$ is the least length of a non-contractible loop on M. In the sequel, it will play a role similar to the role played by the edge length of the fundamental domain above. There are upper bounds depending on the area and the genus of M [23, 8, 16]. The shortest closed curves on M are simple, they are closed geodesics and they are in one-to-one correspondence with the conjugacy classes of elements of Γ. The length of the closed geodesic γ corresponding to a conjugacy class $[h], h \in \Gamma$, is the translation length $l(h)$ [30]. The axis X_h projects to γ by π; it can be thought of as "winding" X_h on γ infinitely many times. Any gX_h, where $g \in \Gamma$, projects onto the same closed geodesic γ by π; the hyperbolic line gX_h is the axis of the translation ghg^{-1}. Note that the conjugacy classes $[h]$ and $[h^2]$ are different; the closed geodesic corresponding to $[h^2]$ traverses twice the closed geodesic corresponding to $[h]$. Also, the closed geodesics corresponding to $[h]$ and $[h^{-1}]$ are different.

3.2 Projecting $\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P})$ on a closed hyperbolic surface

A triangulation defined by a point set $\mathcal{P}$ in $\mathbb{H}^2$ is a *Delaunay triangulation iff* the circumscribing disk of each triangle does not intersect $\mathcal{H}_\infty$ and is *empty*, i.e., it does not contain any point of $\mathcal{P}$ in its interior.

Let $M = \mathbb{H}^2/\Gamma$ be a closed hyperbolic surface, with projection map π. The images by π of two vertices of a triangle σ of $\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P})$ may coincide; in this case at least one of the edges of $\pi(\sigma)$ is a 1-cycle. Any 1-cycle in the graph of edges of $\pi(DT_{\mathbb{H}}(\Gamma\mathcal{P}))$ is freely homotopic to a simple closed geodesic. A 1-cycle e with a vertex v is a geodesic loop. Note that e is not necessarily a closed geodesic: it may form a non-flat angle at vertex v. We refer the reader to the preprint [6] for the more technical characterization of 2-cycles.

Let us consider disks that do not contain any element of the Γ-orbit of a given point p. The Voronoi diagram of Γp is the partition of $\mathbb{H}^2$ by the images by all elements of Γ of Dirichlet regions $\mathcal{D}_p(\Gamma)$. Largest empty disks are centered at images of the vertices of $\mathcal{D}_p(\Gamma)$ that are furthest to p; δ_p denotes their diameter. We define δ_M as $\delta_M = \sup\{\delta_z \mid z \in \mathcal{D}_O(\Gamma)\}$, where O denotes the origin. δ_M is completely determined by the group Γ. The following inequality holds:

▶ **Proposition 8** (Proof in preprint [6]). $\delta_O \leq \delta_M \leq 2\delta_O$.

For a set of points $\mathcal{P}$, we denote as $\delta_{\mathcal{P}}$ the largest diameter of disks that do not contain any point of $\mathcal{P}$. Clearly, for $p \in \mathcal{P}$, $\delta_{\mathcal{P}} \leq \delta_p \leq \delta_M$.

▶ **Proposition 9** (Proof in [6]). *If $sys(M) > \delta_{\mathcal{P}}$ (resp. $sys(M) > 2\delta_{\mathcal{P}}$) then the graph of edges of $\pi(DT_{\mathbb{H}}(\Gamma\mathcal{P}))$ has no 1-cycle (resp. no 2-cycle).*

Section 4.1 shows a covering of the Bolza surface in which the graph $\pi(DT_{\mathbb{H}}(\Gamma\mathcal{P}))$ has a 2-cycle formed by edges of length $\delta_{\mathcal{P}}$, i.e., $sys(M) = 2\delta_{\mathcal{P}}$, so the criterion is sharp. Let now $\widetilde{\Gamma}$ be a subgroup of finite index k in Γ. The quotient space $\widetilde{M} = \mathbb{H}^2/\widetilde{\Gamma}$ is a closed hyperbolic surface with projection map $\widetilde{\pi} : \mathbb{H}^2 \to \widetilde{M}$. The surface $\widetilde{M}$ is a k-sheeted covering space of M with covering map and local isometry $f : \widetilde{M} \to M$ such that $\pi = f \circ \widetilde{\pi}$. Note that $Area(\widetilde{M}) = k \cdot Area(M)$, $genus(\widetilde{M}) = k(genus(M) - 1) + 1$, $sys(\widetilde{M}) \geq sys(M)$, and $\delta_{\widetilde{M}} = \delta_M$.

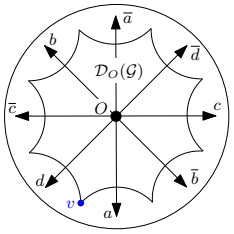


Figure 4 Translations a, b, c, d and their inverses.

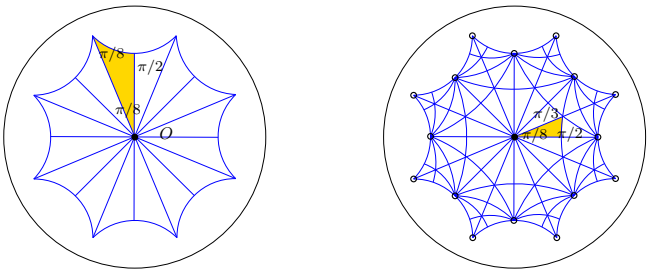


Figure 5 Partition of $\mathcal{D}_O(\mathcal{G})$ into fundamental domains for (left) $T(2,8,8)$ and (right) $T(2,3,8)$ (circles show the elements of the orbit $T(2,3,8)O$ in $\mathcal{D}_O(\mathcal{G})$).

▶ **Proposition 10** (Proof in [6]). *If $sys(\widetilde{M}) > \delta_M$ (resp. $sys(\widetilde{M}) > 2\delta_M$), then for any $\mathcal{P}$, the graph of edges of $\widetilde{\pi}(DT_{\mathbb{H}}(\Gamma\mathcal{P}))$ has no 1-cycle (resp. no 2-cycle). There exists a finite-sheeted covering space $\widetilde{M_1}$ (resp. $\widetilde{M_2}$) of M such that $sys(\widetilde{M_1}) > \delta_M$ (resp. $sys(\widetilde{M_2}) > 2\delta_M$).*

Let us now focus on the specific case of the Bolza surface.

4 The Bolza surface

Consider a regular octagon in $\mathbb{H}^2$ centered at the origin O with angle $\pi/4$ at each vertex, and the four hyperbolic translations a, b, c, d that pairwise identify opposite sides of the octagon (Fig. 4). They generate a Fuchsian group $\mathcal{G}$ for which the octagon is a fundamental domain. The inverse of an element $g \in \mathcal{G}$ is denoted as $\overline{g}$. The group $\mathcal{G}$ can be written as a *finitely presented group*

$$\mathcal{G} = \left\langle a, b, c, d \mid abcd\bar{a}\bar{b}\bar{c}\bar{d} \right\rangle,$$

i.e., the quotient of the group $\langle a, b, c, d\rangle$ generated by a, b, c, d, by the normal closure (i.e., the smallest normal subgroup) of the element $abcd\bar{a}\bar{b}\bar{c}\bar{d}$ in $\langle a, b, c, d\rangle$ [17, Chapter 5.5].

The *Bolza surface* is $\mathcal{M} = \mathbb{H}^2/\mathcal{G}$. It is a closed orientable hyperbolic surface homeomorphic to a torus having two handles, i.e., a double torus.

The group $\mathcal{G}$ shows interesting properties in relation to some triangle groups. For integers $p, q, r \geq 2$, $\frac{1}{p} + \frac{1}{q} + \frac{1}{r} < 1$, the *triangle group* $T(p, q, r)$ is defined as the group generated by reflections in the edges of a triangle $\Delta(p, q, r)$ whose angles are $\frac{\pi}{p}, \frac{\pi}{q}, \frac{\pi}{r}$ [30]. The set of triangles $\{g(\Delta(p, q, r)), g \in T(p, q, r)\}$ partitions the plane $\mathbb{H}^2$. $\mathcal{G}$ is a normal subgroup of index 16 without fixed points of $T(2, 8, 8)$. $\mathcal{G}$ is a normal subgroup of index 96 without fixed points of $T(2, 3, 8)$. These properties of $\mathcal{G}$ are illustrated geometrically on Fig. 5. For $T(2, 3, 8)$ (right): repeated applications of reflections in the edges of $\Delta(2, 3, 8)$ form the fundamental domain of $\mathcal{G}$; the projection map from $\mathbb{H}^2$ to $\mathcal{M}$ induces the partition of the

surface $\mathcal{M}$ into triangles $\Delta(2,3,8)$. $T(2,3,8)$ is actually the largest subgroup of $\mathrm{Isom}(\mathbb{H}^2)$ in which $\mathcal{G}$ is normal, i.e., $T(2,3,8) = \{t \in \mathrm{Isom}(\mathbb{H}^2) \mid t\mathcal{G}t^{-1} = \mathcal{G}\}$.

▶ **Proposition 11.** *If two points p and q of $\mathbb{H}^2$ lie in the same orbit under $T(2,3,8)$, then the Dirichlet regions $\mathcal{D}_p(\mathcal{G})$ and $\mathcal{D}_q(\mathcal{G})$ are isometric.*

As will be seen in the sequel, this is not true otherwise, which is one of the striking differences with the Euclidean case.

Proof. $q = r(p)$ for $r \in T(2,3,8)$. Since $\mathcal{G}$ is normal in $T(2,3,8)$, $r\mathcal{G}r^{-1} = \mathcal{G}$. Then the orbit $\mathcal{G}q = r\mathcal{G}r^{-1}q = r\mathcal{G}p$ is the image of the orbit $\mathcal{G}p$ under r. $\mathrm{VD}_{\mathbb{H}}(\mathcal{G}q)$ is the image of $\mathrm{VD}_{\mathbb{H}}(\mathcal{G}p)$ by r. The result follows since $\mathcal{D}_q(\mathcal{G}) = Cl(V_q(\mathcal{G}q))$. ◀

▶ Notation. For simplicity, in the sequel we denote as g the image $g(O)$ of O by $g \in \mathcal{G}$. We also label each side of $\mathcal{D}_p(\mathcal{G})$ by the translation of $\mathcal{G}$ that maps $\mathcal{D}_p(\mathcal{G})$ to the Dirichlet region adjacent through this side. Let $\rho \in T(2,3,8)$ be the rotation around the origin O by $\pi/4$, and let

$$\mathcal{A} = (a, \overline{b}, c, \overline{d}, \overline{a}, b, \overline{c}, d)$$

be the alphabet formed by the generators of $\mathcal{G}$ and their inverses, considered as an ordered sequence (Fig. 4). Conjugation by ρ acts as a permutation on $\mathcal{A}$.

The Dirichlet regions for $\mathcal{G}$ have been studied in [29]. The rest of this section presents a more intuitive description.

For $x \in \mathcal{A}$, we denote as v_x the vertex of $\mathcal{D}_O(\mathcal{G})$ that is equidistant to O, x, and $\rho^{-1}x\rho$. The eight elements of $\mathcal{A}$ are the elements of the orbit $\mathcal{G}O$ that are closest to O. In the same way, the eight elements $xz, z \in \mathcal{A}$ are closest to x. Any $z \in \mathcal{A}$ can be written as $\rho^k\overline{x}\rho^{-k}$, where $0 \leq k < 8$, so the above $xz, z \in \mathcal{A}$ can be written as $xz = x \cdot \rho^k x \rho^{-k}(O)$. A careful but straightforward observation of the Dirichlet regions of O, x, $x \cdot \rho\overline{x}\rho^{-1}$, etc., shows:

▶ **Observation 12.** *Each vertex v_x is incident to eight isometric Dirichlet regions. See Fig. 6, where $\omega_4(x) = x \cdot \rho\overline{x}\rho^{-1} \cdot \rho^2 x \rho^{-2} \cdot \rho^3 \overline{x} \rho^{-3} = \; y \cdot \rho^{-1}\overline{y}\rho \cdot \rho^{-2} y \rho^2 \cdot \rho^{-3}\overline{y}\rho^3$.*

Without loss of generality, we now examine more closely the vertex v of $\mathcal{D}_O(\mathcal{G})$ that is incident to $\mathcal{D}_a(\mathcal{G})$ and $\mathcal{D}_d(\mathcal{G})$ (see Fig. 4). The eight Dirichlet regions incident to v are centered at the images of O by the following translations, in clockwise order: $F = (\not\vdash, a, ab, abc, abcd = dcba, dcb, dc, d)$ (Fig. 7). The eight Dirichlet regions $\rho^k\mathcal{D}_v(\mathcal{G}), 0 \leq k < 8$, share the common vertex O. The Dirichlet region $\mathcal{D}_v(\mathcal{G})$ is isometric to $\mathcal{D}_O(\mathcal{G})$ since v lies in the $T(2,3,8)$-orbit of O (Fig. 5); $\mathcal{D}_v(\mathcal{G})$ is the regular octagon with vertices in $F(O)$.

For a point $p \in \mathbb{H}^2$, we define $C(p) = \{g \mid g \in \mathcal{G}, \mathcal{D}_p(\mathcal{G}) \cap \mathcal{D}_{g(p)}(\mathcal{G}) \neq \emptyset\}$. Notation C can be remembered as standing for "crown". For conveniency, we will sometimes use the same notation $C(p)$ for the set of points $\{g(p) \mid g \in C(p)\}$ (the context will disambiguate). It is easy to check that for $q = t(p), t \in T(2,3,8)$ we have $C(q) = \{tgt^{-1} \mid g \in C(p)\}$.

Let p be a point in triangle SOQ shown in Fig. 8. Let us consider the Dirichlet regions that are adjacent to $\mathcal{D}_p(\mathcal{G})$.

- If p is in the interior of SOQ, the Dirichlet region $\mathcal{D}_p(\mathcal{G})$ is a polygon with 18 sides. Each vertex of $\mathcal{D}_p(\mathcal{G})$ has degree 3. The sides of $\mathcal{D}_p(\mathcal{G})$ have the following labels in counterclockwise order: $a, a\overline{d}, \overline{b}c\overline{d}, \overline{b}, \overline{b}\overline{a}, c, \overline{d}, \overline{a}, b, \overline{c}\overline{d}, \overline{c}, \overline{c}\overline{b}\overline{a}, d\overline{a}, d, dc, dcb, abc, ab$.
- If p lies in the interior of edge OQ, the sides dc and abc degenerate to a vertex of $\mathcal{D}_p(\mathcal{G})$. On SO, the sides ab and dcb degenerate to a vertex, and on QS b and c degenerate to a vertex. Such a new vertex has degree 4. The Dirichlet region $\mathcal{D}_p(\mathcal{G})$ is a polygon with 14 sides.

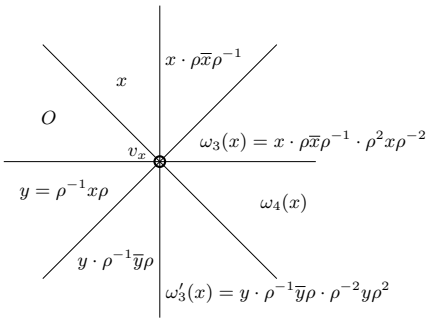


Figure 6 Dirichlet regions around v_x.

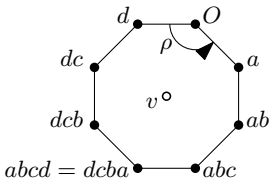


Figure 7 Schematic representation of $\mathcal{D}_v(\mathcal{G})$.

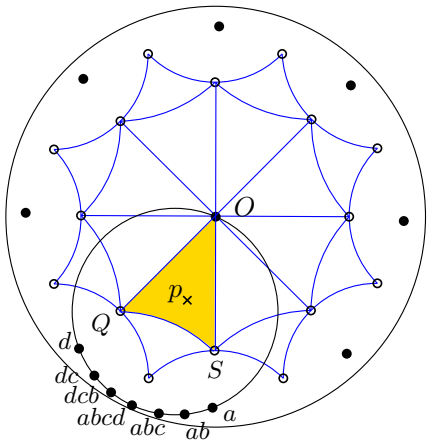


Figure 8 SOQ.

- If $p \in \{O, Q, S\}$, $\mathcal{D}_p(\mathcal{G})$ degenerates to an octagon (Proposition 11). Its adjacent octagons are centered at some points in $C(p)$. In particular: $\mathcal{D}_p(\mathcal{G})$ shares a common vertex with $\mathcal{D}_{abcd(p)}(\mathcal{G})$ *iff* $p = O$.

Fig. 9 shows the Dirichlet regions of three different points q, obtained by computing the Delaunay triangulation in $\mathbb{H}^2$ of sufficiently many points of $\mathcal{G}q$, with the software described in [5].

We define $C^-(O) = C(O) \backslash \{\rho^k(abcd)\rho^{-k}(O) \mid 0 \leq k < 8\}$; it excludes the elements of $C(O)$ corresponding to octagons that meet $\mathcal{D}_O(\mathcal{G})$ opposite a vertex. For $O' = rO, r \in T(2,3,8)$, define $C^-(O') = \{rgr^{-1} \mid g \in C^-(O)\}$. From the description above:

▶ **Proposition 13.** *For any $p \in \mathbb{H}^2$ there exists $O' \in T(2,3,8)O$, such that $C(p) \subset C(O')$. Moreover, if $p \notin T(2,3,8)O$, then $C(p) \subset C^-(O')$.*

▶ **Proposition 14** (Proof in [6])**.** *The largest empty disk diameter satisfies* $4.89 < \delta_{\mathcal{M}} < 6.62$.

There exist closed geodesics that are $< \delta_{\mathcal{M}}$ [2] (see preprint [6]).

4.1 Covering spaces of the Bolza surface

The function `LowIndexSubgroupsFpGroup( Gamma, H, k, excluded )` of GAP [21, 34] provides in principle a way to compute the two covering spaces of Prop. 10. However, its huge memory consumption does not allow us to use it for the Bolza surface. We propose a construction of coverings spaces of $\mathcal{M}$ in which, for any set $\mathcal{P}$, the projection $\pi(\mathrm{DT}_{\mathbb{H}}(\Gamma\mathcal{P}))$ does not contain

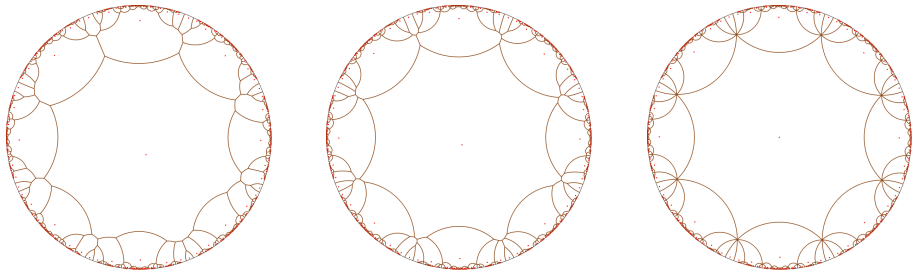


Figure 9 Dirichlet region for $\mathcal{G}$ centered at different points q (rightmost picture: $q = O$).

any 1- or 2-cycle. In a similar spirit as for the flat torus, here we look for subgroups $\widetilde{\mathcal{G}}$ of $\mathcal{G}$ not containing elements corresponding to short loops, and whose associated Dirichlet regions keep some symmetry.

$\mathcal{G}$ is a normal subgroup of the triangle groups $T(2,3,8), T(2,4,8), T(2,8,8)$. They contain the rotation ρ defined earlier, and the reflection τ with respect to the axis X_a of translation a. Obviously, if $\widetilde{\mathcal{G}}$ is a normal subgroup of one of these three triangle groups, then $\widetilde{\mathcal{G}} = \rho\widetilde{\mathcal{G}}\rho^{-1}$ and $\widetilde{\mathcal{G}} = \tau\widetilde{\mathcal{G}}\tau^{-1}$ (which holds for $\widetilde{\mathcal{G}} = \mathcal{G}$).

Let us introduce the subset $N_O(\widetilde{\mathcal{G}}) \subset \widetilde{\mathcal{G}}$ of elements g such that the Dirichlet regions $\mathcal{D}_O(\widetilde{\mathcal{G}})$ and $\mathcal{D}_g(\widetilde{\mathcal{G}})$ are adjacent (i.e., they share a common edge). For $\widetilde{\mathcal{G}} = \mathcal{G}$, $N_O(\mathcal{G}) = \mathcal{A}$. More generally, $N_O(\widetilde{\mathcal{G}})$ generates $\widetilde{\mathcal{G}}$ [26].

▶ **Observation 15.** *The Dirichlet region $\mathcal{D}_O(\widetilde{\mathcal{G}})$ for $\widetilde{\mathcal{G}}$ centered at O is invariant by ρ (resp. τ) if and only if $\widetilde{\mathcal{G}} = \rho\widetilde{\mathcal{G}}\rho^{-1}$ (resp. $\widetilde{\mathcal{G}} = \tau\widetilde{\mathcal{G}}\tau^{-1}$).*

Proof. If $\widetilde{\mathcal{G}} = \rho\widetilde{\mathcal{G}}\rho^{-1}$, then by definition of ρ, $\rho\widetilde{\mathcal{G}}O = \rho\widetilde{\mathcal{G}}\rho^{-1}O = \widetilde{\mathcal{G}}O$, and the Dirichlet region $\mathcal{D}_O(\widetilde{\mathcal{G}})$, as a cell in $\mathrm{VD}_{\mathbb{H}}(\widetilde{\mathcal{G}}O)$, is invariant by ρ (and similarly for τ).

Now assume that $\mathcal{D}_O(\widetilde{\mathcal{G}})$ is invariant by ρ. Then in particular $N_O(\widetilde{\mathcal{G}})(O) = \rho N_O(\widetilde{\mathcal{G}})(O)$, and we can rewrite $N_O(\widetilde{\mathcal{G}})(O)$ as $\rho N_O(\widetilde{\mathcal{G}})\rho^{-1}(O)$ (since $\rho(O) = O$). Thus for each $g \in N_O(\widetilde{\mathcal{G}})$, there is some $h \in N_O(\widetilde{\mathcal{G}})$ such that $g(O) = \rho h\rho^{-1}(O)$, and since $\mathcal{G} = \rho\mathcal{G}\rho^{-1}$, so $\rho h\rho^{-1}$ is in $\mathcal{G}$. Then the translation $\rho h\rho^{-1} \cdot \overline{g}$ is in $\mathcal{G}$, and it fixes O. Since $\mathcal{G}$ has no fixed point, it must be the identity, and $\rho h\rho^{-1} = g$. This is true for each g in $N_O(\widetilde{\mathcal{G}})$, so $N_O(\widetilde{\mathcal{G}}) = \rho N_O(\widetilde{\mathcal{G}})\rho^{-1}$. Since $N_O(\widetilde{\mathcal{G}})$ generates $\widetilde{\mathcal{G}}$, we have $\widetilde{\mathcal{G}} = \rho\widetilde{\mathcal{G}}\rho^{-1}$. The same proof works for τ: the only property of ρ that we used is that $\rho(O) = O$, which is also true for τ. ◀

It follows as a corollary that if $\widetilde{\mathcal{G}}$ is normal in a triangle group, then $\mathcal{D}_O(\widetilde{\mathcal{G}})$ is invariant under ρ and τ. The three triangle groups $T(2,m,8)$ mentioned above can be generated by ρ, τ and some other rotation r_m. If $\mathcal{D}_O(\widetilde{\mathcal{G}})$ is invariant under ρ and τ, then $\widetilde{\mathcal{G}}$ is normal in $T(2,m,8)$ if and only if $\widetilde{\mathcal{G}} = r_m\widetilde{\mathcal{G}}r_m^{-1}$.

Before explicitly constructing subgroups of $\mathcal{G}$, let us derive a condition that will be used later (Prop. 23):

▶ **Proposition 16.** *The condition that $\widetilde{\mathcal{G}}$ contains no element of $C^-(O')$ for any $O' \in T(2,3,8)O$ is necessary and sufficient for the graph of edges of the projection of $DT_{\mathbb{H}}(\mathcal{GP})$ onto $\widetilde{\mathcal{M}} = \mathbb{H}^2/\widetilde{\mathcal{G}}$ to have no 1-cycle, for any $\mathcal{P}$.*

Proof. Let e be an edge of $DT_{\mathbb{H}}(\mathcal{GP})$ such that $\pi(e)$ is a geodesic loop on $\mathcal{M}$, i.e., $e = [p, g(p)]$ for some $p \in \mathbb{H}^2, g \in \mathcal{G}$. Since $e \in DT_{\mathbb{H}}(\mathcal{GP})$, the cells $Cl(V_p(\mathcal{GP}))$ and $Cl(V_{g(p)}(\mathcal{GP}))$ intersect. Since $Cl(V_p(\mathcal{GP})) \subset Cl(V_p(\mathcal{G}p)) = \mathcal{D}_p(\mathcal{G})$ and similarly $Cl(V_{g(p)}(\mathcal{GP})) \subset \mathcal{D}_{g(p)}(\mathcal{G})$, then $\mathcal{D}_p(\mathcal{G}) \cap \mathcal{D}_{g(p)}(\mathcal{G}) \neq \emptyset$, i.e., $g \in C(p)$.

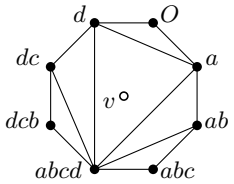


Figure 10 Avoiding edge $[O, abdc]$.

By Proposition 13, $g \in C(p) \subset C(O')$ for some $O' \in T(2,3,8)O$. So if $\widetilde{\mathcal{G}}$ is a subgroup of $\mathcal{G}$ that contains no elements of $C(O')$, then the projection of $DT_{\mathbb{H}}(\mathcal{GP})$ to $\widetilde{\mathcal{M}} = \mathbb{H}^2/\widetilde{\mathcal{G}}$ has no 1-cycle.
Assume that $g \notin C^-(O')$. Then Proposition 13 implies that $p \in T(2,3,8)O$. Without loss of generality assume $p = O$ and $g = abcd$. If $\mathcal{P}$ contains more than one point, then $[O, abcd]$ cannot be a Delaunay edge. If $\mathcal{P} = \{O\}$, then as shown on Fig. 10, we can choose a triangulation of the octagon $\mathcal{D}_v(\mathcal{G})$ that does not contain edge $[O, abcd]$ (see also the treatment of degenerate cases in the introduction section). Then the projection of $DT_{\mathbb{H}}(\mathcal{GP})$ has no 1-cycle. Remark that for any $g \in C^-(O)$, we can choose $DT_{\mathbb{H}}(\mathcal{G}O)$ so that the geodesic segment with endpoints O and g is an edge of $DT_{\mathbb{H}}(\mathcal{G}O)$. ◀

In the sequel, we use the standard notation $[g, h]$ for the *commutator* of two isometries g and h, i.e., $[g, h] = ghg^{-1}h^{-1}$.

We first define the subgroup $\mathcal{G}_2$ of $\mathcal{G}$ generated by words of even size on $\mathcal{A}$ (Fig. 11). $\mathcal{G}_2$ is a subgroup of index 2 in $\mathcal{G}$, hence it is normal in $\mathcal{G}$.

▶ **Proposition 17.** *$\mathcal{G}_2$ is generated by $\{[x, \rho] \mid x \in \mathcal{A}\}$.*

Proof. It is sufficient to prove that any word of size 2 on $\mathcal{A}$ can be generated by this subset. Consider $yz, y, z \in \mathcal{A}$. As mentioned in Section 4, ρ acts as a permutation on $\mathcal{A}$, i.e., $y = \rho^{k_1} x \rho^{-k_1}$ and $z = \rho^{k_2} \overline{x} \rho^{-k_2}$ for some $k_1, k_2, 0 \leq k_1, k_2 < 8$. Then $yz = (\rho^{k_1} x \rho^{-k_1} \cdot \rho^{k_1+1} \overline{x} \rho^{-k_1-1}) \cdot (\rho^{k_1+1} x \rho^{-k_1-1} \cdot \rho^{k_1+2} \overline{x} \rho^{-k_1-2}) \cdot \ldots \cdot (\rho^{k_2-1} x \rho^{-k_2+1} \cdot \rho^{k_2} \overline{x} \rho^{-k_2})$. ◀

The subgroup $\mathcal{G}_2$ can be written in a finitely presented form:

$$\mathcal{G}_2 = \left\langle ab, cd, \overline{a}\overline{b}, \overline{c}\overline{d}, bc, d\overline{a}, \overline{b}\overline{c}, \overline{d}a \mid abcd\overline{a}\overline{b}\overline{c}\overline{d} \right\rangle .$$

As a consequence of Observation 12, $[x, \rho]$, $[\rho^{-1} x \rho, \rho]$, and O are equidistant to v_x, and the Dirichlet region of O for $\mathcal{G}_2$ can be completely described (Fig. 11):

▶ **Observation 18.** *The fundamental region $\mathcal{D}_O(\mathcal{G}_2)$ has 16 vertices that form the circular sequence $(v_x,\ x \mid x \in \mathcal{A})$, in counterclockwise order. The set $N_O(\mathcal{G}_2)$ is the circular sequence $([x, \rho], [x, \rho^{-1}] \mid x \in \mathcal{A})$ in counterclockwise order.*

The Dirichlet region of O is clearly invariant under ρ and τ. Indeed, conjugation by ρ or τ leaves $\mathcal{A}$ invariant, so it preserves the size of a word in $\mathcal{G}$. By Observation 15, $\mathcal{G}_2 = \rho \mathcal{G}_2 \rho^{-1} = \tau \mathcal{G}_2 \tau^{-1}$. We can actually show:

▶ **Proposition 19** (Proof in preprint [6])**.** *$\mathcal{G}_2$ is the only subgroup of index 2 of $\mathcal{G}$ for which the Dirichlet region centered at O is invariant by ρ.*

The generators of $\mathcal{G}_2$ all have the same translation length $l(a) = l(b) = l(c) = l(d)$. As a result, $sys(\mathcal{M}_2) = sys(\mathcal{M})$. However, there are fewer shortest loops on $\widetilde{M}$ than on $\mathcal{M}$, since the elements of $\mathcal{A}$ are not in $\mathcal{G}_2$.

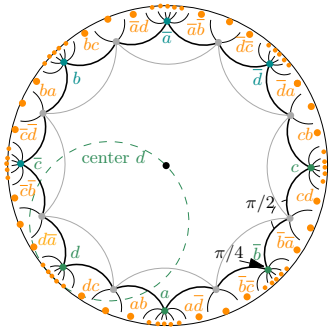


Figure 11 $\mathcal{D}_O(\mathcal{G}_2)$.

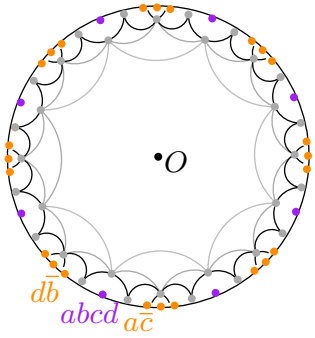


Figure 12 $\mathcal{D}_O(\mathcal{G}_4)$.

As for the flat case, we can continue and define inductively the group $\mathcal{G}_{2^k}$ consisting of elements of even size in the alphabet of generators of $\mathcal{G}_{2^{k-1}}$, for $k \geq 1$. Recall notation $\omega_4(x) = x \cdot \rho\overline{x}\rho^{-1} \cdot \rho^2 x \rho^{-2} \cdot \rho^3 \overline{x} \rho^{-3}$ (Fig. 6).

▶ **Observation 20.** *The domain $\mathcal{D}_O(\mathcal{G}_4)$ has 24 vertices that form the circular sequence $([x,\rho], x, [x,\rho^{-1}] \mid x \in \mathcal{A})$, in counterclockwise order. The set $N_O(\mathcal{G}_4)$ is the circular sequence $(\omega_4(x), x \cdot \rho^2\overline{x}\rho^{-2}, x \cdot \rho^{-2}\overline{x}\rho^2 \mid x \in \mathcal{A})$, in counterclockwise order.*

Proof. Let Π be the polygon with vertices $([x,\rho], x, [x,\rho^{-1}] \mid x \in \mathcal{A})$, in counterclockwise order. Consider a vertex $x \in \mathcal{A}$ of Π. Recall that the elements of the set $\{x \cdot \rho^k \overline{x} \rho^{-k} \mid 0 \leq k < 8\}$ are equidistant from x. Vertices $(\rho \cdot x \cdot \rho^{-1}) \cdot (\rho^2 \cdot \overline{x} \cdot \rho^{-2})$, $(\rho^2 \cdot x \cdot \rho^{-2}) \cdot (\rho^3 \cdot \overline{x} \cdot \rho^{-3})$, and $\rho\overline{x}\rho^{-1} \cdot \overline{x}$ of $\mathcal{D}_O(\mathcal{G}_2)$ are equidistant to O. Thus $[x,\rho] \cdot (\rho \cdot x \cdot \rho^{-1}) \cdot (\rho^2 \cdot \overline{x} \cdot \rho^{-2})$, $[x,\rho] \cdot (\rho^2 \cdot x \cdot \rho^{-2}) \cdot (\rho^3 \cdot \overline{x} \cdot \rho^{-3})$, and $[x,\rho]\rho\overline{x}\rho^{-1} \cdot \overline{x}$ are equidistant from the vertex $[x,\rho]$ of Π, i.e., $x \cdot \rho^2\overline{x}\rho^{-2}$, $\omega_4(x)$, and O are equidistant from the vertex $[x,\rho]$ of Π. The proof is similar for $[x,\rho^{-1}]$. ◀

Using the fact that $N_O(\mathcal{G}_4)$ generates $\mathcal{G}_4$, we get:

▶ **Corollary 21.** *The set $\{\omega_4(x), x \cdot \rho^2\overline{x}\rho^{-2}, x \cdot \rho^{-2}\overline{x}\rho^2 \mid x \in \mathcal{A}\}$ generates $\mathcal{G}_4$.*

In other words, the generators of $\mathcal{G}_4$ are $ac, abcd, d\overline{b}$, and their conjugates by $\rho^k, 1 < k < 8$ (Fig. 12). In a similar way as done for $\mathcal{G}_2$, we can show that $\mathcal{G}_4$ is the only subgroup of index 2 of $\mathcal{G}_2$ for which the Dirichlet region centered at O is invariant by ρ.

▶ **Proposition 22.** *The systole of $\mathcal{M}_4 = \mathbb{H}^2/\mathcal{G}_4$ is equal to $l(abcd)$.*

Proof. $sys(\mathcal{M})$ is equal to the translation length $l(a)$ of the translation a. The translations of $\mathcal{G}$ that have the same translation length as a are conjugates hah^{-1}, where $h \in T(2,3,8)$. As $a \notin \mathcal{G}_4$ and $\mathcal{G}_4$ is normal in $T(2,3,8)$, it does not contain hah^{-1} for any $h \in T(2,3,8)$. Thus $sys(\mathbb{H}^2/\mathcal{G}_4) > l(a)$. The second shortest length of a non-contractible loop corresponds to geodesics of length $l(abcd)$ [2] (see preprint [6]). Since $abcd \in \mathcal{G}_4$ (violet point in Fig. 12), the result follows. ◀

The systole of $\mathcal{M}_4$ is greater than $sys(\mathcal{M})$, but still smaller than $\delta_{\mathcal{M}}$. However, as a consequence of Proposition 16:

▶ **Proposition 23.** *For any $\mathcal{P}$, the projection of $DT_{\mathbb{H}}(\mathcal{GP})$ on $\mathcal{M}_4$ has no 1-cycle.*

Due to lack of space, we only mention the following result (see preprint [6]):

▶ **Proposition 24.** *The index of a subgroup $\widetilde{\mathcal{G}}$ for which $sys(\mathbb{H}^2/\widetilde{\mathcal{G}}) > 2\delta_{\mathcal{M}}$ is greater than 32. There is a unique subgroup $\mathcal{G}_{128}$ of $\mathcal{G}$ of index 128 that is normal in $T(2,3,8)$. For any set of points $\mathcal{P}$, the graph of edges of the projection of $DT_{\mathbb{H}}(\mathcal{GP})$ to $\mathbb{H}^2/\mathcal{G}_{128}$ has no 2-cycle.*

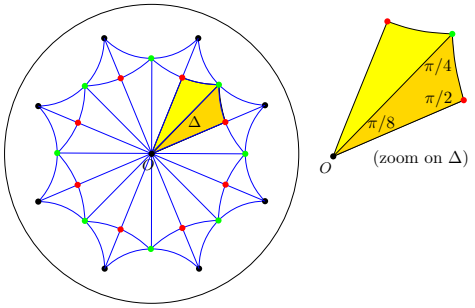


Figure 13 A fundamental domain Δ for $T(2,4,8)$.

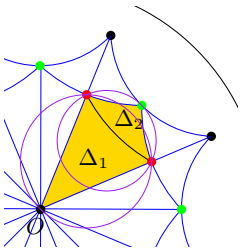


Figure 14 Flip.

4.2 Triangulating the Bolza surface in practice

As mentioned in introduction, the incremental algorithm for computing a Delaunay triangulation can be extended to $\mathcal{M}$, using a covering space of $\mathcal{M}$ in which the graph of its edges has no 2-cycles, similarly to what was done for Euclidean manifolds [14]. However, computing in $\mathbb{H}^2/\mathcal{G}_{128}$ (Proposition 24) would not be efficient, since we would have to consider 128 images of $\mathcal{P}$.

In this section, we propose a heuristic method instead, inspired from [10, 11]. It consists in first initializing the triangulation with a well chosen set $\mathcal{Q}$ of 14 points for which $sys(\mathcal{M}) > 2\delta_{\mathcal{Q}}$. Inserting new points in the triangulation cannot increase the size of empty disks, i.e., $\delta_{\mathcal{P}'\cup\mathcal{Q}} \leq \delta_{\mathcal{Q}}$ for any set $\mathcal{P}'$, so by Proposition 9, $\pi(\mathrm{DT}_{\mathbb{H}}(\mathcal{G}(\mathcal{P}'\cup\mathcal{Q})))$ has no 2-cycle. Thus, the incremental algorithm can be used to compute the Delaunay triangulation of $\mathcal{P}\cup\mathcal{Q}$ directly in $\mathcal{M}$. In a lot of applications, keeping these 14 extra vertices in the triangulation would not be an issue. However, there may be cases when this should be avoided. If $\mathcal{P}$ contains sufficiently many points, and if they are reasonably well distributed, which is likely to be the case in applications,[3] then the final triangulation $\pi(\mathrm{DT}_{\mathbb{H}}(\mathcal{G}\mathcal{P}))$ would have no 1- or 2-cycles. In such cases, all points of $\mathcal{Q}$ can be removed from $\pi(\mathrm{DT}_{\mathbb{H}}(\mathcal{G}(\mathcal{P}\cup\mathcal{Q})))$ (basically, as in a 2D Euclidean triangulation [18] – practical details are omitted). However, for badly distributed point sets, such short cycles will appear during the process, which can be checked after each removal using the geometric criterion; in such cases, points of $\mathcal{Q}$ cannot be removed further.

The rest of this section is devoted to presenting the construction of $\mathcal{Q}$.

The group $\mathcal{G}$ is a normal subgroup without fixed point of index 48 in the triangle group $T(2,4,8)$. The octagon $\mathcal{D}_O(\mathcal{G})$ can be partitioned into 48 triangles that are fundamental domains for $T(2,4,8)$ (Fig. 13). Let Δ be such a triangle incident to O. If $\mathcal{Q}_T$ denotes the set of vertices of Δ, we define $\mathcal{Q} = T(2,4,8)\mathcal{Q}_T \cap \mathcal{D}_O(\mathcal{G})$. Note that $T(2,4,8)\mathcal{Q}_T = \mathcal{G}\mathcal{Q}$. Finally, we flip an edge of Δ as shown on Fig. 14.

The disks circumscribing Δ_1 and Δ_2 contain no point of $T(2,4,8)\mathcal{Q}_T$ in their interiors. $\Delta_1 \cup \Delta_2$ is a fundamental domain for $T^+(2,4,8) = T(2,4,8) \cap \mathrm{Isom}^+(\mathbb{H}^2)$. This partition of $\mathbb{H}^2$ is $\mathrm{DT}_{\mathbb{H}}(T(2,4,8)\mathcal{Q}_T) = \mathrm{DT}_{\mathbb{H}}(\mathcal{G}\mathcal{Q})$ since $T(2,4,8)\mathcal{Q}_T = \mathcal{G}\mathcal{Q}$.

Let $\delta_{\mathcal{Q}}$ denote the largest empty disk diameter for $\mathcal{G}\mathcal{Q}$, and d_1 (resp. d_2) denote the diameters of the disks circumscribing Δ_1 (resp. Δ_2). There is a triangle in $\mathrm{DT}_{\mathbb{H}}(\mathcal{G}\mathcal{Q})$ whose circumscribing disk has diameter $\delta_{\mathcal{Q}}$, and since any triangle in $\mathrm{DT}_{\mathbb{H}}(\mathcal{G}\mathcal{Q})$ is the image of Δ_1 or Δ_2 under some isometry of $T^+(2,4,8)$, either d_1 or d_2 is equal to $\delta_{\mathcal{Q}}$.

[3] This was observed and analyzed in the 3D Euclidean case [10, 12].

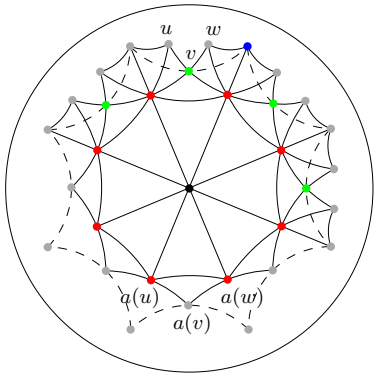


Figure 15 The set $\mathcal{Q}$.

Both d_1 and d_2 are less than $\frac{1}{2}sys(\mathcal{M})$, which actually is the radius of the circle inscribed in $\mathcal{D}_O(\mathcal{G})$. So $\delta_{\mathcal{Q}} < \frac{1}{2}sys(\mathcal{M})$.

Fig. 15 shows unique representatives in $\mathbb{H}^2$ of the triangles of $\pi(\mathrm{DT}_{\mathbb{H}}(\mathcal{G}\mathcal{Q}))$. The point set $\mathcal{Q}$ contains 9 points in the interior of $\mathcal{D}_O$ (origin and red points), 4 points on the boundary of $\mathcal{D}_O$ (green points) and 1 point at its vertex (blue point). In total $\mathcal{Q}$ consists of 14 points.

5 Open problems

Further work directions are twofold. On a theoretical side, we are planning to reduce the gap between the lower bound 32 and the upper bound 128 in Proposition 24; we conjecture that 32 sheets are always sufficient to avoid 2-cycles. We will also investigate generalizations of the results given here to closed hyperbolic surfaces of higher genus. On a practical side, we are working on arithmetic issues arising in the heuristic method presented in Section 4.2, and on its implementation for CGAL.

References

1 M. Artin. *Algebra*. Prentice-Hall, 2002.

2 R. Aurich, E. B. Bogomolny, and F. Steiner. Periodic orbits on the regular hyperbolic octagon. *Physica D: Nonlinear Phenomena*, 48(1):91–101, 1991. doi:10.1016/0167-2789(91)90053-C.

3 A. Bachelot-Motet. Wave computation on the hyperbolic double doughnut. *Journal of Computational Mathematics*, 28:1–17, 2010. doi:10.4208/jcm.1004.m3120.

4 M. Berger. *Geometry (vols. 1-2)*. Springer-Verlag, 1987.

5 M. Bogdanov, O. Devillers, and M. Teillaud. Hyperbolic Delaunay complexes and Voronoi diagrams made practical. *Journal of Computational Geometry*, 5:56–85, 2014. URL: http://jocg.org/index.php/jocg/article/view/141.

6 M. Bogdanov and M. Teillaud. Delaunay triangulations and cycles on closed hyperbolic surfaces. Research Report 8434, INRIA, December 2013. URL: http://hal.inria.fr/hal-00921157.

7 A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24:162–166, 1981.

8 P. Buser and P. Sarnak. On the period matrix of a Riemann surface of large genus. *Inventiones mathematicae*, 117:27–56, 1994. doi:10.1007/BF01232233.

9 J. W. Cannon, W. J. Floyd, R. Kenyon, and W. R. Parry. Hyperbolic geometry. *Flavors of geometry*, 31:59–115, 1997. URL: http://www.math.uwo.ca/~shafikov/teaching/winter2010/4156/hyperbolic.pdf.

10 M. Caroli. *Triangulating Point Sets in Orbit Spaces*. Phd thesis, Université de Nice-Sophia Antipolis, France, 2010. URL: http://tel.archives-ouvertes.fr/tel-00552215/.

11 M. Caroli and M. Teillaud. 3D periodic triangulations. In *CGAL Manual*. CGAL Editorial Board, 3.5 edition, 2009. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic3Triangulation3Summary.

12 M. Caroli and M. Teillaud. Computing 3d periodic triangulations. In *European Symposium on Algorithms*, volume 5757 of *LNCS*, pages 59–70, 2009. To appear in [14]. doi:10.1007/978-3-642-04128-0_6.

13 M. Caroli and M. Teillaud. Delaunay triangulations of point sets in closed Euclidean d-manifolds. In *Proc. 27th Symposium on Computational Geometry*, pages 274–282, 2011. To appear in [14]. doi:10.1145/1998196.1998236.

14 M. Caroli and M. Teillaud. Delaunay triangulations of closed Euclidean d-orbifolds. *Discrete & Computational Geometry*, To Appear. See also [12] and [13].

15 P. Chossat, G. Faye, and O. Faugeras. Bifurcation of hyperbolic planforms. *Journal of Nonlinear Science*, 21:465–498, 2011. doi:10.1007/s00332-010-9089-3.

16 É. Colin de Verdière, A. Hubard, and A. de Mesmay. Discrete systolic inequalities and decompositions of triangulated surfaces. In *Proc. 30th Symposium on Computational Geometry*, pages 335–344, 2014. To appear in Discrete & Computational Geometry.

17 H. S. M. Coxeter and W. O. J. Moser. *Generators and Relations for Discrete Groups*. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1957.

18 O. Devillers. Vertex removal in two-dimensional Delaunay triangulation: Speed-up by low degrees optimization. *CGTA*, 44:169–177, 2011. doi:10.1016/j.comgeo.2010.10.001.

19 O. Devillers and M. Teillaud. Perturbations for Delaunay and weighted Delaunay 3D triangulations. *CGTA*, 44:160–168, 2011. doi:10.1016/j.comgeo.2010.09.010.

20 N.P. Dolbilin and D.H. Huson. Periodic Delone tilings. *Periodica Mathematica Hungarica*, 34:1-2:57–64, 1997.

21 Gap – groups, algorithms, programming. URL: http://www.gap-system.org.

22 C. I. Grima and A. Márquez. *Computational Geometry on Surfaces*. Kluwer Academic Publishers, 2001.

23 M. Gromov. Systoles and intersystolic inequalities. In *Actes de la Table Ronde de Géométrie Différentielle (Luminy, 1992)*, volume 1 of *Sémin. Congr.*, pages 291–362. Soc. Math. France, 1996.

24 J. Hidding, R. vd Weygaert, G. Vegter, B.J.T. Jones, and M. Teillaud. Video: The sticky geometry of the cosmic web. In *Proceedings of the Twenty-eighth Annual Symposium on Computational Geometry*, pages 421–422, 2012. URL: http://www.computational-geometry.org/SoCG-videos/socg12video/.

25 M.K. Hurdal and K. Stephenson. Cortical cartography using the discrete conformal approach of circle packings. *Neuroimage*, 23:119–128, 2004.

26 S. Katok. *Fuchsian Groups*. Chicago Lectures in Mathematics. Uni. Chicago Press, 1992.

27 N. Kruithof. 2D periodic triangulations. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.4 edition, 2014. URL: http://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic2Triangulation2Summary.

28 M. Mazón and T. Recio. Voronoi diagrams on orbifolds. *Computational Geometry: Theory and Applications*, 8:219–230, 1997. doi:10.1016/S0925-7721(96)00017-X.

29 M. Naatanen. On the stability of identification patterns for Dirichlet regions. *Annales Academiæ Scientiarum Fennicæ, Series A.I. Mathematica*, 10:411–417, 1985.

30 J. Ratcliffe. *Foundations of Hyperbolic Manifolds*. Graduate Texts in Mathematics. Springer, 2006.

31 G. Ron, M. Jin, and X. Guo. Hyperbolic centroidal Voronoi tessellation. In *Proc. ACM Symp. on Solid and Physical Modeling*, pages 117–126, 2010. doi:10.1145/1839778.1839795.

32 F. Sausset, G. Tarjus, and P. Viot. Tuning the fragility of a glassforming liquid by curving space. *Physical Review Letters*, 101:155701(1)–155701(4), 2008. doi:10.1103/PhysRevLett.101.155701.

33 B. Schuetrumpf, M. A. Klatt, K. Iida, G. E. Schröder-Turk, J. A. Maruhn, K. Mecke, and P.-G. Reinhard. Appearance of the single gyroid network phase in "nuclear pasta" matter. *Physical Review C*, 91(025801), 2015. doi:10.1103/PhysRevC.91.025801.

34 C.C. Sims. *Computation with Finitely Presented Groups*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994.

35 T. Sousbie. The persistent cosmic web and its filament structure I. *Monthly Notices of the Royal Astronomical Soc.*, 414:350–383, 2011. doi:10.1111/j.1365-2966.2011.18394.x.

36 T. Sousbie, C. Pichon, and H. Kawahara. The persistent cosmic web and its filament structure II. *Monthly Notices of the Royal Astronomical Society*, 414:384–403, 2011. doi:10.1111/j.1365-2966.2011.18395.x.

37 J. Stillwell. *Geometry of Surfaces*. Springer-Verlag, New York, 1992.

38 W. P. Thurston. Three dimensional manifolds, Kleinian groups, and hyperbolic geometry. *Bull. Amer. Math. Soc.*, 6(3):357–381, 1982. URL: http://www.ams.org/mathscinet-getitem?mr=648524.

39 W. P. Thurston. *Three Dimensional Geometry and Topology, vol. I.* Princeton University Press, Princeton, New Jersey, 1997.

40 R. vd Weygaert, G. Vegter, H. Edelsbrunner, B.J.T. Jones, P. Pranav, C. Park, W. A. Hellwing, B. Eldering, N. Kruithof, E.G.P. Bos, J. Hidding, J. Feldbrugge, E. ten Have, M. v Engelen, M. Caroli, and M. Teillaud. Alpha, Betti and the megaparsec universe: on the homology and topology of the cosmic web. In *Trans. on Comp. Science XIV*, volume 6970 of *LNCS*, pages 60–101. Springer-Verlag, 2011. doi:10.1007/978-3-642-25249-5_3.

41 P M. H. Wilson. *Curved Spaces*. Cambridge University Press, Cambridge, 2008.

An Efficient Randomized Algorithm for Higher-Order Abstract Voronoi Diagrams*

Cecilia Bohler[1], Rolf Klein[2], and Chih-Hung Liu[3]

1 **Department of Computer Science, University of Bonn, Bonn, Germany**
bohler@cs.uni-bonn.de
2 **Department of Computer Science, University of Bonn, Bonn, Germany**
rolf.klein@uni-bonn.de
3 **Department of Computer Science, University of Bonn, Bonn, Germany**
chliu@uni-bonn.de

Abstract

Given a set of n sites in the plane, the order-k Voronoi diagram is a planar subdivision such that all points in a region share the same k nearest sites. The order-k Voronoi diagram arises for the k-nearest-neighbor problem, and there has been a lot of work for point sites in the Euclidean metric. In this paper, we study order-k Voronoi diagrams defined by an abstract bisecting curve system that satisfies several practical axioms, and thus our study covers many concrete order-k Voronoi diagrams. We propose a randomized incremental construction algorithm that runs in $O(k(n-k)\log^2 n + n\log^3 n)$ steps, where $O(k(n-k))$ is the number of faces in the worst case. Due to those axioms, this result applies to disjoint line segments in the L_p norm, convex polygons of constant size, points in the Karlsruhe metric, and so on. In fact, this kind of run time with a polylog factor to the number of faces was only achieved for point sites in the L_1 or Euclidean metric before.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Order-k Voronoi Diagrams, Abstract Voronoi Diagrams, Randomized Geometric Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.21

1 Introduction

Given a set S of n sites in the plane, the order-k abstract Voronoi diagram $V_k(S)$ of S partitions the plane into Voronoi regions such that all points within a region $\mathrm{VR}_k(H,S)$ share the same set H of k nearest sites in S where the underlying proximity is defined by an abstract bisecting curve system. Order-k Voronoi diagrams solve the k nearest neighbor problem. However, the distance measure in the real world is in general not the Euclidean but depends on the type of geometric sites and the underlying environment. To deal with diverse proximities, Klein [16] introduced the notion of *abstract Voronoi diagrams* to model the proximity by an abstract bisecting curve system $\mathcal{J}$. For two sites $p, q \in S$, he considered a simple curve $J(p,q)$ as a bisector that splits the plane into two domains $D(p,q)$ and $D(q,p)$. $D(p,q)$ represents the set of points closer to p than to q. Under these circumstances, the order-k Voronoi region $\mathrm{VR}_k(H,S)$ is defined as:

$$\mathrm{VR}_k(H,S) = \bigcap_{p\in H, q\in S\setminus H} D(p,q).$$

* This work was supported by Deutsche Forschungsgemeinschaft (DFG Kl 655/19) in a DACH project.

© Cecilia Bohler, Rolf Klein, and Chih-Hung Liu;
licensed under Creative Commons License CC-BY

If a bisecting curve system satisfies certain axioms, combinatorial properties and algorithms are directly applicable to the resulting Voronoi diagrams [3, 5, 16, 17, 18, 21].

We consider the following six axioms: for each subset S' of S of size at least 3,

(A1) Each first-order Voronoi region in $V_1(S')$ is pathwise connected.

(A2) Each point in the plane belongs to the closure of some first-order Voronoi region.

(A3) No first-order Voronoi region in $V_1(S')$ is empty.

(A4) Each curve $J(p,q)$, where $p \neq q$, is unbounded. After stereographic projection to the sphere, it can be completed to be a closed Jordan curve through the north pole.

(A5) Any two curves $J(p,q)$ and $J(s,t)$ have only finitely many intersection points, and these intersections are transversal. At most three bisecting curves $J(p,\cdot)$ associated with the same site p can pass through the same point.

(A6) The number of vertical tangencies of a curve $J(p,q)$ is $O(1)$, and for any two curves $J(p,q)$ and $J(s,t)$, their vertical tangencies are distinct.

Despite Axiom (A1), $\mathrm{VR}_k(H,S)$, where $k>1$, may consist of disjoint *faces*, i.e., connected components. The six axioms cover many concrete Voronoi diagrams, so the abstract Voronoi diagram is a prototype of many concrete ones. It is known that axioms (A1) and (A2) need only be verified for subsets S' of size 3, and (A3), for size 4 subsets [17, 3]. The properties postulated in (A5) are to avoid technical difficulties; for order-1 abstract Voronoi diagrams it has been shown in [16, 17] how to proceed without these assumptions, and we are confident that the results in this paper can be generalized, too.

Lee [19] first proved that the order-k Voronoi diagram has $O(k(n-k))$ faces for point sites in the Euclidean metric. Only recently, Papadopoulou and Zavershynskyi [23] showed that the number of faces for disjoint line segments remains $O(k(n-k))$, and this bound remains valid for intersecting segments if $k \geq n/2$. Soon later, Bohler et al. [3] proved that the number of faces in the order-k abstract Voronoi diagram is at most $2k(n-k)$, which is a tight bound in the abstract setting. Gemsa et al. [14] and Liu and Lee [20] studied point sites in the city and geodesic metrics, respectively, neither of which lies under the envelope of the above axioms. Construction algorithms have been well-studied for point sites in the Euclidean metric. Deterministic algorithms by Lee [19] and by Chazelle and Edelsbrunner [10] achieve $O(k^2 n \log n)$ and $O(n^2 + k(n-k)\log^2 n)$ time, respectively. Clarkson [12] proposed a randomized divide-and-conquer algorithm with $O(kn^{1+\epsilon})$ time. Moreover, Aurenhammer and Schwarzkopf [2] and Boissonnat et al. [6] studied on-line algorithms. Most efficient algorithms rely on a geometric transformation that maps each point site to a hyperplane in three dimensions which is tangent to a unit paraboloid $z = x^2 + y^2$ at the vertical projection of the point site. In this situation, computing the order-k Voronoi diagram is reduced to computing the so called k-level of the arrangement formed by the transformed hyperplanes.

In recent years, some algorithms for settings different from point sites in the Euclidean metric were invented. For point sites in the L_1 metric, Liu et al. [20] derived an output-sensitive algorithm with $O(m \log n)$ time, where $m = O(\min\{k(n-k), (n-k)^2\})$; for line segments, Papadopoulou and Zavershynskyi [23] obtained $O(k^2 n \log n)$ time. Bohler et al. [5] developed a randomized divide-and-conquer algorithm for the abstract version, and obtained $O(kn^{1+\epsilon})$ time, which works for many concrete cases including disjoint line segments. Their algorithm interprets Clarkson's general idea [12] and replaces geometric operations with combinatorial ones.

Agarwal et al. [1] proposed a randomized incremental algorithm to compute the k-level which intermediately maintains cells that possibly intersect the final k-level, and their algorithm yields $O(k(n-k)\log n)$ construction time for order-k Voronoi diagrams. Chan [7] proposed a framework to compute the k-level, and adopted Agarwal et al's algorithm as a

black-box to obtain $O(nk \log k + n \log n)$ time. Ramos [24] later improved the construction time to $O(n \log n + nk^{O(\log^* k)})$, and his algorithm is a mixture of divide-and-conquer and incremental scheme with Chan's framework on the top. Very recently, Chan and Tsakalidis [8] derandomized Chan's framework and achieved $O(nk \log k)$ (or $O(nk \log k \log \log k)$) deterministic time.

It is an interesting question if a construction time with a polylogarithmic factor to the output size can be achieved for settings different from point sites in L_1 or L_2. In other words, is it possible to extend classical k-level algorithms [1, 7, 24] to abstract Voronoi diagrams? Such an extension would be quite different from Bohler et al.'s [5] extension of Clarkson [12], because Clarkson's algorithm principally adopts two planar sub-algorithms although he explained the general idea in three dimensions. In contrast, those k-level algorithms fully perform in three dimensions, exploiting the geometry of planes tangent to the paraboloid, and it seems quite challenging to convert them to non-point sites and non-Euclidean metrics, or even to the abstract version, based only on combinatorial properties of curves.

In this paper we give a first positive answer, proposing an $O(k(n-k) \log^2 n + n \log^3 n)$-time randomized incremental algorithm for order-k abstract Voronoi diagrams. Due to the six axioms, this result applies to a wide range of concrete order-k Voronoi diagrams including point sites in any algebraic convex distance metric or the Karlsruhe metric, disjoint line segments and disjoint convex polygons of constant size in the L_p norms, or under the Hausdorff metric (assuming for now sites to be in general position; see the comment on axiom (A5) from above). Such near-optimal run time is achieved for the first time for examples different from point sites in the Euclidean and L_1 metrics.

Our algorithm is strongly inspired by the k-level algorithm of Agarwal et al. [1]. We also proceed incrementally and maintain all faces of the higher-order Voronoi regions that are "active". This notion can be translated to abstract Voronoi diagrams in the following way. For a subset R of S and for each face F of $\mathrm{VR}_j(Q, R)$ we maintain its intersections with the farthest Voronoi diagram $\mathrm{FV}(Q)$, and with the nearest Voronoi diagram $V(R \setminus Q)$.

Each resulting sub-face $F \cap \mathrm{FVR}(q, Q)$ is decomposed into trapezoids. Such a trapezoid $\triangle$ is called *active* if, for some representative point $y \in \triangle$, the level of q at y does not exceed k plus the number of conflicts of $\triangle$. Here, the *level* of q equals the number of $s \in S$ such that $y \in D(s, q)$ holds, plus 1. The *conflicts* of $\triangle$ are sites $s \in S$ whose bisector $J(s, q)$ intersects $\triangle$ (or one of its defining edges, which may exceed the trapezoid).

Similarly, a trapezoid in a sub-face $F \cap \mathrm{VR}(p, R \setminus Q)$ is called active if the level of p, at some point, is larger than k minus the number of conflicts. Face F is called active if both intersections contain an active trapezoid.

The analysis in [1] relies on two simple facts: a hyperplane that crosses a segment must separate its two endpoints, and a new hyperplane separates a cell into two adjacent cells. The former implies that both the conflict size of a simplex (the number of hyperplanes intersecting it) and the level difference between two points in a cell (the number of hyperplanes separating them) are upper bounded by a constant times the diameter of the cell (the maximal number of hyperplanes intersecting a line segment in it); the latter implies that the diameter of one generated cell is at least half that of the original cell. However, this geometric analysis could not be directly applicable to the abstract setting because a bisector that intersects a vertical segment or a Voronoi edge does not necessarily separate its two endpoints, and a new site can separate a face into a non-constant number of new faces and they are even not necessarily pairwise adjacent. The second reason also illustrates an important phenomenon in the abstract setting that an order-k region may be disconnected for $k > 1$.

Therefore, we are using a different approach that may be interesting in its own right.

What one would really like to maintain, in the incremental construction, are those faces $F \subseteq \mathrm{VR}_j(Q,R)$ that contain a non-empty face of some region $\mathrm{VR}_k(H,S)$, such that $Q = H \cap R$ holds. We call these faces F *essential*. It is easy to verify that all essential faces are active. We were able to show some sort of converse: with high probability, all faces our algorithm constructs, including all active faces and those who are found to be inactive and discarded, are essential with respect to some order k' in a certain interval around k.

In [1], a new hyperplane partitions a cell of the arrangement into two cells. The geometry of the lowest-vertex triangulation can be used to decide the activity of the two new cells in time proportional to the total number of new or destroyed simplices. In the abstract setting, a face may be split into many faces, and in each of them old trapezoids may survive. Since we cannot afford to re-check them, in order to decide the activity of the new faces, we employ, for each active face, a nested data structure storing sub-faces and their edges. This approach only works because the intersections $F \cap \mathrm{FV}(Q)$ and $F \cap V(R \setminus Q)$ can be shown to be trees; for the second structure, this fact was not previously known. The tree structure allows the sub-faces of F in these intersections to be stored in cyclic order, which behave well under insertion of a new site. However, using the data structure approach incurs an extra $\log n$ factor in our run time bound.

When inserting a new site s, for each face F of $\mathrm{VR}_j(Q,R)$, we compute the sets

$$\begin{aligned} F \cap \mathrm{FVR}(s, Q \cup \{s\}) &\subseteq \mathrm{VR}_j(Q, R \cup \{s\}) \\ F \cap \mathrm{VR}(s, R \cup \{s\} \setminus Q) &\subseteq \mathrm{VR}_{j+1}(Q \cup \{s\}, R \cup \{s\}). \end{aligned}$$

Each of these intersections can consist of several connected components, giving rise to several faces of the new Voronoi regions on the right hand side. But the way F may split up is controlled by an unexpected structural property, as we will see below. This property also shows how disconnected Voronoi regions emerge.

To conclude, our algorithm can be seen as a combinatorial interpretation of Agarwal et al's algorithm [1], with a different analysis and with geometric properties replaced by combinatorial facts about bisecting curves. The algorithm applies to a wide range of order-k Voronoi diagrams, at the cost of an extra $O(\log n)$ factor. To some extent, our work decodes the powerful geometric transformation between points in the plane and hyperplanes in three dimensions, and indicates that the happy marriage between arrangements and random sampling techniques can be extended to more general proximities than point sites in the Euclidean metric.

2 Preliminaries

The common boundary between two faces in $V_k(S)$ is called a *Voronoi edge*, and the common vertex among more than two faces in $V_k(S)$ is called a *Voronoi vertex*. The *order-k Voronoi diagram* $V_k(S)$ equals the union of the boundaries of all order-k regions or, equivalently, the union of the intersections of the closures of any two order-k Voronoi regions.

Due to Axiom (5), a Voronoi vertex v among $\mathrm{VR}_k(H_1,S)$, $\mathrm{VR}_k(H_2,S)$, and $\mathrm{VR}_k(H_3,S)$ can be categorised into two types: v is called *new* if $|H_1 \cap H_2 \cap H_3| = k-1$, and v is called *old* if $|H_1 \cap H_2 \cap H_3| = k-2$ [3].

Due to Axioms (A1) and (A5), any two bisecting curves $J(p,q)$ and $J(p,r)$ intersect at most twice [16]. Moreover, the following lemma holds [18].

▶ **Lemma 1.** *For any three sites $p,q,r \in S$ one has $D(p,q) \cap D(q,r) \subseteq D(p,r)$.*

This transitivity property ensures that, for each point x not situated on any bisecting curve, the relation $p <_x q \iff x \in D(p,q)$ is a total ordering on any subset of S. For $R \subseteq S$ we define

$$l_p(x,R) := 1 + |\{t \in R \setminus \{p\} \mid x \in D(t,p)\}|$$

to be the *level of* p at point x with respect to R, abbreviated $l_p(x)$ if $R = S$. For every Voronoi vertex v of $V_k(S)$, if v is the intersection between $J(p,q)$, $J(p,t)$, and $J(q,t)$, we have $l_q(v) = l_p(v) = l_t(v) = k-2$ or $k-1$ depending on whether v is old or new.

We define Γ to be a large closed Jordan curve such that no pair of bisectors cross on or outside Γ, and each bisector crosses Γ exactly twice and these intersections are transversal. If we add Γ to $V_k(S)$ and cut off all parts contained in the outer domain, we obtain a connected graph without unbounded edges, so we can view all faces in $V_k(S)$ to be bounded.

In our algorithm, the following basic operations are assumed to take $O(1)$ time:

1. For an arbitrary point x, determine if x belongs to $D(p,q)$, $J(p,q)$ or $D(q,p)$.
2. For a point x on $J(p,q)$, along one direction of $J(p,q)$, determine the next intersection with $J(s,t)$ or a straight line, or determine the next point where the curve reverses direction.
3. For two points x, y on $J(p,q)$, determine which point comes first in a given direction.

Notations. For simplicity, the first order region of a site p in R will be denoted by $\mathrm{VR}(p,R)$, and the first order diagram by $V(R)$. Similarly, if R is of size r, the Voronoi diagram of order $r-1$ is the farthest diagram, $\mathrm{FV}(R)$, and the farthest region of a site q is $\mathrm{FVR}(q,R)$.

For a subset R of S, we define $\mathrm{VF}(R)$ as the collection of all faces in $V_j(R)$ for $1 \le j \le |R|-1$. For an open set $A \subseteq \mathbb{R}^2$, we use ∂A and $\mathrm{cl}A$ to denote its boundary and closure, respectively.

3 Main concepts

Let $(s_1, s_2, \ldots, s_n)$ be a random sequence of S, and let R_i be the first i sites in the sequence, i.e., $R_i = \{s_1, s_2, \ldots, s_i\}$. We incrementally insert a site in the sequence and finally obtain $V_k(S)$. However, since $E[\sum_{i=k+1}^{n} |V_k(R_i)|] = \Omega(nk^2)$ [2] but $|V_k(S)| = O(k(n-k))$ [3], it is too expensive to compute all $V_k(R_i)$ for $k+1 \le i \le n$.

We now describe an alternate approach inspired by [1].

3.1 Dominance

To compute the faces of the non-empty regions of $V_k(S)$ we are maintaining certain faces of lower-order diagrams.

▶ **Definition 2.** A face F_1 of $\mathrm{VR}_{j_1}(Q_1,R_1)$ *dominates* a face F_2 of $\mathrm{VR}_{j_2}(Q_2,R_2)$, where $R_1 \subseteq R_2$ and $j_1 \le j_2$, if $F_1 \supseteq F_2$ and $Q_1 = R_1 \cap Q_2$. A face is called *essential* if it dominates a face of $V_k(S)$.

We observe that $Q_1 = R_1 \cap Q_2$ implies $(R_1 \setminus Q_1) \cap Q_2 = \emptyset$. Moreover, $Q_1 \subseteq Q_2$ and $R_1 \subseteq R_2$ imply $\mathrm{VR}_{j_1}(Q_1,R_1) \supseteq \mathrm{VR}_{j_2}(Q_2,R_2)$, so that $F_1 \supseteq F_2$ already follows from $F_1 \cap F_2 \neq \emptyset$ since faces are path-connected. Clearly, the dominance relation is transitive.

Essential faces can be characterized in the following way.

▶ **Lemma 3.** *A face F of $VR_j(Q,R)$ dominates a face of $V_k(S)$ if and only if there exists a point $x \in F$ where $x \in FVR(q,Q)$ and $x \in VR(p, R \setminus Q)$ such that $l_q(x) \le k < l_p(x)$.*

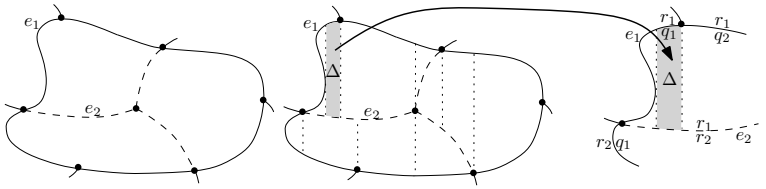


Figure 1 Left: $\check{F}= (V(R\setminus Q)\cap F)\cup\partial F$. Middle: $\check{F}^{\nabla}$. Right: $d(\triangle) = r_1$ and $D(\triangle)=\{r_1, r_2, q_1, q_2\}$.

Proof. *Necessity:* Let C be the face of $V_k(S)$ dominated by F and let C belong to $\mathrm{VR}_k(H, S)$. Consider a point $x \in C$ that does not lie on any bisector. Since H are the k nearest sites of x and $Q \subseteq H$, q is at most the k^{th} nearest neighbor of x, and since $(R \setminus Q) \cap H = \emptyset$, p is at least the $(k+1)^{\text{st}}$ nearest neighbor of x, implying $l_q(x) \leq k < l_p(x)$.

Sufficiency: Let C be the face of $V_k(S)$ that contains x and let C belong to $\mathrm{VR}_k(H, S)$. Since $l_q(x) \leq k$ and q is the farthest neighbor of x in Q, $Q \subseteq H$, and since $l_p(x) > k$ and p is the nearest neighbor of x in $R \setminus Q$, we have $(R \setminus Q) \cap H = \emptyset$, implying $Q = R \cap H$, so that F dominates C. ◀

Lemma 3 suggests to partition a face F of $\mathrm{VR}_j(Q, R)$ by $\mathrm{FV}(Q)$ and $V(R\setminus Q)$, respectively, and we use $\hat{F}$ and $\check{F}$ to denote these two subdivisions. Faces in $\hat{F}$ and $\check{F}$ are called *sub-faces*. The following lemma determines the structures of $\hat{F}$ and $\check{F}$, and implies that each sub-face of $\hat{F}$ or $\check{F}$ shares exactly one edge with the boundary of F, which is called an *outer* edge, while the edges of the refined Voronoi diagrams are called *inner* edges. (See Fig. 1)

▶ **Lemma 4.** *For a face F of $VR_j(Q, R)$, if $j > 1$, $FV(Q) \cap F$ is a tree, and if $j < |R| - 1$, $V(R \setminus Q) \cap F$ is a tree. (If $j = 1$ (resp. $j = |R| - 1$), $\hat{F}$ (resp. $\check{F}$) has exactly one sub-face.)*

3.2 Trapezoidal Decompositions

As before, let F be a face of $\mathrm{VR}_j(Q, R)$. For efficient computations, we partition (the sub-faces in) $\hat{F}$ and $\check{F}$ into vertical trapezoids, and denote the result by $\hat{F}^{\nabla}$ and $\check{F}^{\nabla}$, respectively. Abusing the notation slightly, we also use $\hat{F}$ and $\check{F}$ to denote the corresponding decompositions. F^{∇} is the set of trapezoids in $\hat{F}^{\nabla}$ and $\check{F}^{\nabla}$. For example, the middle drawing of Fig. 1 illustrates $\check{F}^{\nabla}$.

We assume that each trapezoid is adjacent to at most two trapezoids on either side. This assumption can be attained by inserting zero-width trapezoids whenever necessary [9].

▶ **Lemma 5.** *F^{∇} has $O(|F|)$ trapezoids, where $|F|$ is the number of edges bounding F.*

Now, we describe what information on single trapezoids we are interested in. For all trapezoids $\triangle$ in $\hat{F}^{\nabla}$ of a sub-face $F \cap \mathrm{FVR}(q, Q)$, we call q the *owner* of $\triangle$, denoted by $d(\triangle)$. Similarly, for all trapezoids $\triangle$ in $\check{F}^{\nabla}$ that decompose a sub-face $F \cap VR(p, R \setminus Q)$, we call $d(\triangle) := p$ the owner of $\triangle$.

By $D(\triangle)$ we denote the set of sites *defining* $\triangle$. Precisely, $\triangle$ is defined by at most two edges (top and bottom) and two vertical segments (left and right). Note that an edge here means an edge of $\hat{F}$ or $\check{F}$, and it could exceed the boundary of $\triangle$. For example, as shown in the right of Fig. 1, the top and bottom edges of $\triangle$ are e_1 and e_2. All the edges and the segments are associated with $d(\triangle)$-bisectors, and $|D(\triangle)| \leq 9$. See Fig. 1 for an illustration.

Moreover, for a trapezoid $\triangle$, we say a site $t \in S \setminus D(\triangle)$ is in *conflict with* $\triangle$ if $J(t, d(\triangle))$ intersects $\triangle$ or one of the two edges that define $\triangle$. In other words, $\triangle$ does not exist in $\mathrm{VF}^{\nabla}(R \cup \{t\})$, or one of its defining edges has been changed. It is clear that no site

$t \in R \setminus D(\triangle)$ can have a conflict with $\triangle$. We let $J(\triangle)$ be the set of sites t in conflict with $\triangle$, and $w(\triangle)$ be $|J(\triangle)|$. We call $J(\triangle)$ and $w(\triangle)$ the *conflict list* and *conflict size* of $\triangle$, respectively. In addition, we say a site $t \in S \setminus R$ *conflicts with* $F \in \mathrm{VF}(R)$ if t conflicts with a trapezoid of F^∇.

In the sequel, we will select an arbitrary point $y_\triangle \in \triangle$ for each trapezoid $\triangle \in F^\nabla$, and compute the level of its owner, $l_{d(\triangle)}(y_\triangle)$.

3.3 Active faces

Since it is hard to determine the existence of x in Lemma 3, instead of processing the essential faces, we define "active" faces in the following way.

▶ **Definition 6.** Consider a face F of $\mathrm{VR}_j(Q, R)$. For a trapezoid $\triangle$ of $\hat{F}^\nabla$, $\triangle$ is called *active* if $l_{d(\triangle)}(y_\triangle) - w(\triangle) \le k$; for a trapezoid $\triangle$ of $\check{F}^\nabla$, $\triangle$ is called *active* if $l_{d(\triangle)}(y_\triangle) + w(\triangle) > k$. Finally, F is called *active* if both $\hat{F}^\nabla$ and $\check{F}^\nabla$ contain an active trapezoid.

▶ **Lemma 7.** *An essential face must be active. Active faces in $VF(S)$ are faces in $V_k(S)$.*

4 Algorithm

Let $\mathrm{AF}(R)$ be the set of active faces in $\mathrm{VF}(R)$, the set of all faces of all higher order regions over site set R, and let AF_i be $\mathrm{AF}(R_i)$, for short. We first compute AF_{10} directly, and then from $i = 10$ to $i = n - 1$, we iteratively compute AF_{i+1} from AF_i, leading to the faces of $V_k(S)$ (Lemma 7). Finally, since adjacencies between faces are not recorded in AF_i, we gather all vertices of $V_k(S)$ from faces of $V_k(S)$, and link all those vertices in $O(k(n-k)\log n)$ time [5] to obtain $V_k(S)$.

During the incremental construction we maintain the following structures:

- The decompositions F^∇ for all faces $F \in \mathrm{AF}(R)$; for every trapezoid in F^∇, its at most four adjacent trapezoids, and its at most two defining edges; for every edge of $\hat{F}$ and $\check{F}$, all adjacent trapezoids on its two sides.
- The conflict lists: for every trapezoid $\triangle \in \mathrm{AF}(R)^\nabla$, the conflict list $J(\triangle)$, and for every site $s \in S \setminus R$, the set of all trapezoids of $\triangle \in \mathrm{AF}(R)^\nabla$ in conflict with s.
- Levels: for every trapezoid $\triangle \in \mathrm{AF}(R)^\nabla$, a chosen point $y_\triangle \in \triangle$ and $l_{d(\triangle)}(y_\triangle)$.
- Decompositions of nearest- and farthest-site Voronoi diagrams: $V(R)^\nabla$ and $\mathrm{FV}(R)^\nabla$.
- Data structures: a data structure for every face $F \in \mathrm{AF}(R)$, allowing the localization of trapezoids in F and a quick test of the active status of F; these data structures will be discussed in Section 4.4.

Once a new site $s \in S \setminus R$ has been randomly chosen for insertion, we find the trapezoids in conflict with s and, by means of the data structures, the faces they belong to.

4.1 Influence of a new site on an old face

Throughout the rest of this section, let F denote a face of $\mathrm{VR}_j(Q, R)$ in conflict with s. We compute the sets

$$F_- := F \cap \mathrm{FVR}(s, Q \cup \{s\})$$
$$F_+ := F \cap \mathrm{VR}(s, R \cup \{s\} \setminus Q)$$

by tracing their boundaries in F, denoted by J_- and J_+, through $\hat{F}^\nabla$ and $\check{F}^\nabla$, respectively.

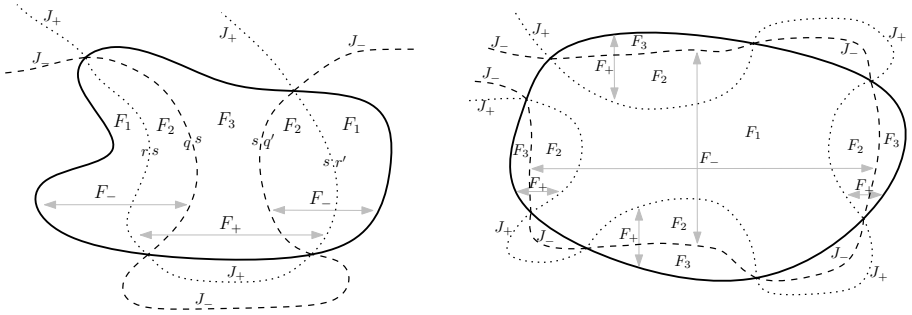


Figure 2 J_- partitions F into F_- and F_3, J_+ partitions F into F_+ and F_1, and $F_2 = F_+ \cap F_-$.

In this section we study which new faces arise from F and how to obtain their decompositions into farthest- and nearest-site Voronoi diagrams. In the subsequent sections, tracing of J_- and J_+ at the trapezoid level and updating trapezoids, including conflict lists and activity status, will be addressed.

The next lemma can be derived by evaluating local orderings.

▶ **Lemma 8.** *Each face of F_- is a face of $VR_j(Q, R \cup \{s\})$, and each face of F_+ is a face of $VR_{j+1}(Q \cup \{s\}, R \cup \{s\})$.*

Note that F_- and F_+ can be disconnected and overlap; see Fig. 2. Yet, these sets and their boundaries have surprising structural properties, as the following lemmas show.

▶ **Lemma 9.** *$F_- \cup F_+ = F$. J_- and J_+ do not intersect each other inside F.*

Proof. For the first statement, assume the contrary that there exists a point $x \in F$ such that $x \notin F_-$ and $x \notin F_+$. Since $x \notin F_-$, there exists a site $q \in Q$ such that $x \in D(s, q)$. Since $x \notin F_+$, there exists a site $p \in R \setminus Q$ such that $x \in D(p, s)$. By the transitivity (Lemma 1), $x \in D(p, q)$. However, since $x \in F$, $x \in D(q, p)$, leading to a contradiction.

For the second statement, if J_- and J_+ intersect inside F, $F_- \cup F_+ \subsetneq F$, contradicting the first statement. ◀

▶ **Lemma 10.** *Neither J_- nor J_+ contains a closed cycle in F*

Proof. If J_- contains a closed cycle, then $\mathrm{FV}(Q \cup \{s\})$ contains a bounded face, contradicting Lemma 4. Assume that J_+ contains a closed cycle in F, and let K be the region enclosed. Since J_+ is the boundary of $\mathrm{VR}(s, R \cup \{s\} \setminus Q)$, K must be $\mathrm{VR}(s, R \cup \{s\} \setminus Q)$; otherwise, there must exist a site $t \in R \setminus Q$ such that $\mathrm{VR}(t, R \cup \{s\} \setminus Q)$ is enclosed by $\mathrm{VR}(s, R \cup \{s\} \setminus Q)$, implying that $J(s, t)$ is closed, i.e., a contradiction. In this situation, J_+ is exactly a closed curve. Since $F_- \cup F_+ = F$ (Lemma 9), J_- does not contain any closed cycle, and J_+ is a closed curve, F_- is exactly F, and contains $\mathrm{VR}(s, R \cup \{s\} \setminus Q)$. By Lemma 8, F_- is a face of $\mathrm{VR}_j(Q, R \cup \{s\})$, and by Lemma 4, each face of $F_- \cap V(R \cup \{s\} \setminus Q)$ touches the boundary of F. However, since J_+ is a closed curve in F and does not intersect ∂F, $\mathrm{VR}(s, R \cup \{s\} \setminus Q)$ forms a bounded region in $F_- \cap V(R \cup \{s\} \setminus Q)$, contradicting Lemma 4. ◀

▶ **Lemma 11.** *J_- intersects ∂F at a point x if and only if J_+ intersects ∂F at x. Hence, J_- intersects F if and only if J_+ intersects F.*

Proof. For the first statement, we show necessity as follows. Let e be the Voronoi edge of F which contains x and assume that e is between $\mathrm{VR}_j(Q, R)$ and $\mathrm{VR}_j(Q \cup \{p\} \setminus \{q\}, R)$, where $q \in Q$ and $p \in R \setminus Q$, i.e., $e \subset J(p, q)$. In this situation, q is the farthest site of x in Q, and

p is the nearest site of x in $R \setminus Q$. Since $x \in \mathrm{FVR}(q, Q)$ and J_- passes through x, $J(q, s)$ passes through x. Since $J(q, s)$ and $J(p, q)$ intersect at x, $J(p, s)$ passes through x. Since $x \in \mathrm{VR}(p, R \setminus Q)$ and $J(p, s)$ passes through x, J_+ passes through x.

The proof of sufficiency is symmetric to the necessity proof. The second statement directly follows from the first statement and Lemma 10. ◀

For short, we let $F_1 := F \setminus F_+$ and $F_3 := F \setminus F_-$. By the above, the set $F_2 := F_- \cap F_+$ may consist of several faces, each of which is bounded by one segment of J_- and one of J_+. Each face of F_2 touches ∂F in exactly two points where J_- and J_+ meet; see Fig. 2. The following observation helps in constructing the farthest-site and nearest-site Voronoi diagrams inside the new faces.

▶ Observation. For points $x_1 \in F_1$, $x_2 \in F_2$, and $x_3 \in F_3$, the levels of the new site s at x_1, x_2, and x_3 with respect to $R \cup \{s\}$ are at least $j+2$, exactly $j+1$ and at most j, respectively.

For each face C of F_-, we obtain $\mathrm{FV}(Q) \cap C$ by clipping $\mathrm{FV}(Q) \cap F$ with the segments of J_- bounding C. To obtain $V(R \cup \{s\} \setminus Q) \cap C$ we observe that inside F_1 site s cannot be the nearest; here we can keep $V(R \setminus Q)$, clipped by J_+. To be added is a face of F_2 bounded by J_+ and J_- which equals the nearest Voronoi region of s in $V(R \cup \{s\} \setminus Q)$ inside C.

Similarly, let C' be a face of F_+. Inside F_3, site s cannot be the farthest in $Q \cup \{s\}$; here we can still use $\mathrm{FV}(Q) \cap F$, clipped by J_-. To obtain $\mathrm{FV}(Q \cup \{s\}) \cap C'$ we add the same face of F_2 as above, which also equals the farthest region of s in $\mathrm{FV}(Q \cup \{s\})$ inside C'. Moreover, we clip $V(R \setminus Q) \cap F$ by $J_+ \cap C'$ to obtain $V(R \cup \{s\} \setminus (Q \cup \{s\}) \cap C'$; see Figures 2 and 3.

To sum up, once we have J_- and J_+, we can generate new faces from F and update their farthest-site and nearest-site Voronoi diagrams. Since this process will not generate $\mathrm{VR}_1(\{s\}, R \cup \{s\})$ and $\mathrm{VR}_r(R, R \cup \{s\}) = \mathrm{FVR}(s, R \cup \{s\})$, we maintain $V(R)$ and $\mathrm{FV}(R)$ separately during the incremental construction, as well as their trapezoidal decompositions, conflict lists, and active statuses. The total expected time for this extra task is in $O(n \log n)$; see [21, 18, 4].

▶ Remark. Generating F_- from F is equivalent to removing F_3 from F. Since each face C of F_- is a face of $\mathrm{VR}_j(Q, R \cup \{s\})$, one may wonder which faces are adjacent to C in $V_j(R \cup \{s\})$. These faces belong to F'_+, for some faces F' in $V_{j-1}(R)$, and are obtained in the same way F_+ is obtained from F.

4.2 Tracing bisecting curves

The boundary J_- of F_- in F ($J_- = \partial F_- \cap F$) can be traced through $\hat{F}^\nabla$ using local information stored at trapezoids, starting from a trapezoid in conflict with s one of whose edges is external, i. e., belongs to ∂F. Namely, if site q is the owner $d(\triangle)$ of a trapezoid $\triangle$ then $J_- = J(q, s)$ holds inside $\triangle$. This stays true as J_- moves into a neighboring trapezoid through a vertical edge of $\triangle$. If J_- leaves $\triangle$ through its top or bottom edge e, two cases are possible. If $e \subset J(q, q')$ is an inner edge, J_- now becomes part of $J(q', s)$. In case of an outer edge on the boundary of F, tracing this segment of J_- halts.

Since two s-bisectors intersect at most twice, J_- intersects the top and the bottom edges of $\triangle$ at most four times, and since a bisector has a constant number of vertical tangencies, J_- intersects the left and right segments of $\triangle$ at most $O(1)$ times. Therefore, the size of $J_- \cap \triangle$ is $O(1)$, and only $O(1)$ time will be spent on computing $J_- \cap \triangle$. Thus, the time to trace J_- is upper bounded by the number of trapezoids in $\hat{F}^\nabla$ that are in conflict with s, times a constant.

The same holds for tracing J_+ through $\check{F}^\nabla$.

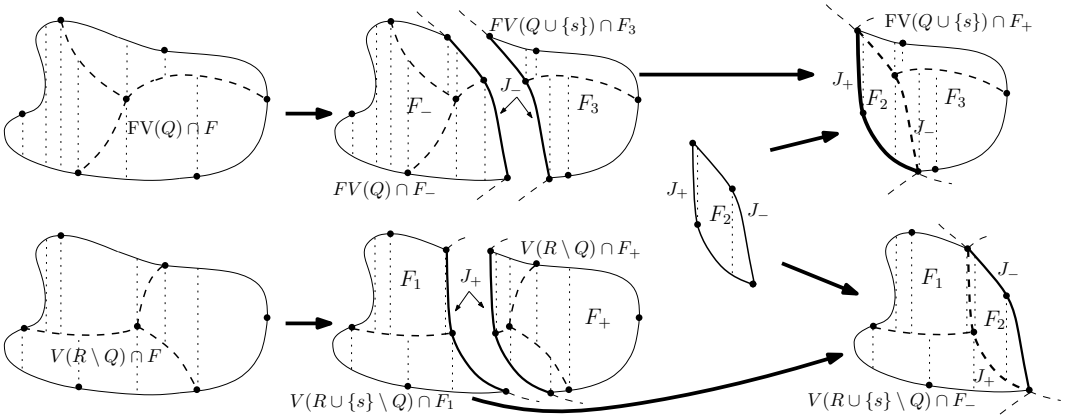


Figure 3 Computation of $F_-^{\triangledown}$ and $F_+^{\triangledown}$ from $F^{\triangledown}$. Two copies of F_2 are needed.

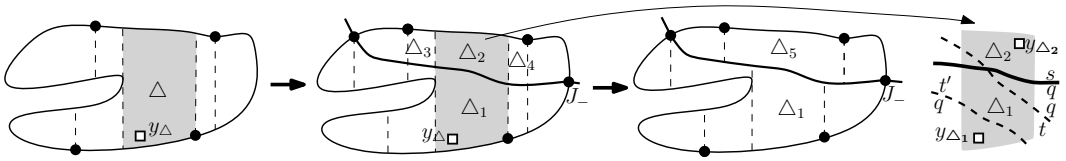


Figure 4 Separating a trapezoid and merging trapezoids.

4.3 Updating Trapezoidal Decompositions

As Fig. 3 illustrates, in F_- and F_3 existing trapezoids will be altered by J_- and J_+, whereas in F_2 a new trapezoid decomposition has to be built from scratch. While the geometric changes to the trapezoids are quite straightforward to implement, we also have to update conflict lists and activity statuses.

4.3.1 Trapezoids in F_- and F_3

Reconstructing $\hat{F_-}^{\triangledown}$ and $\hat{F_3}^{\triangledown}$, together with their conflict lists, is similar to the standard incremental construction for vertical trapezoidal decompositions (see Mulmuley's book [22]). It involves separating existing trapezoids by J_- and merging trapezoids sharing the same edges, and takes time proportional to the total conflict size of destroyed trapezoids; see Fig. 4. Also, testing the new trapezoids for activity is quite straightforward. Thus we only remark on a subtle fact here: a trapezoid $\triangle'$ generated from an inactive trapezoid $\triangle$ may be active although $\triangle'$ is smaller than $\triangle$. This is because the top or bottom edge changes such that a site t that does not conflict with $\triangle$ may conflict with $\triangle'$. As shown in Fig. 5, $J(q', t)$ and $J(q', t)$ intersect the top edge e' of $\triangle'$ but not the top edge e of $\triangle$.

4.3.2 Trapezoids in F_2

We separate each edge of F_2 into x-monotone curves, and apply Chazelle's algorithm [9] to compute the vertical trapezoidal decomposition for each face of F_2. His algorithm works for x-monotone curves, and takes time proportional to the number of created trapezoids.

To construct the conflict lists, we note that each face of F_2 is bounded by a curve of J_- and a curve of J_+, due to the results in Subsection 4.1, and each trapezoid in $F_2^{\triangledown}$ is dominated by s. Consider a face C of F_2. Since there is no closed bisector, if a trapezoid in

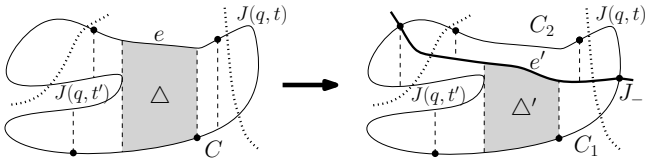


Figure 5 C is a sub-face of $F \cap \mathrm{FVR}(q, Q)$, and t and t' conflict with $\triangle'$ but not $\triangle$.

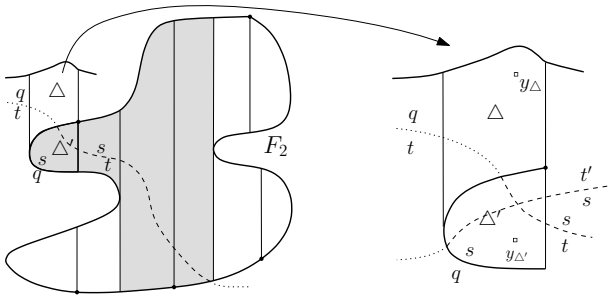


Figure 6 Left: Tracing conflicts for t where $d(\triangle) = q$ and $d(\triangle') = s$. Right: Walking from y to y' one must pass through $J(q, s)$, $J(t', s)$ and $J(s, t)$, so $l_s(y') = l_q(y) + 2 - 1$.

C^∇ is conflicted by a site $t \in S \setminus (R \cup \{s\})$, $J(s, t)$ must intersect the boundary of C, and thus t must also be in conflict with a trapezoid outside C and adjacent to the boundary of C. Therefore, we can compute conflict lists for trapezoids in C^∇ from the outside of C (see the left drawing of Fig. 6).

To check the active statuses, for each face C of F_2, we select a pair of trapezoids, $\triangle$ and $\triangle'$, outside and inside C, respectively, such that they share an edge along the boundary of C. Recall that $d(\triangle') = s$. Since $l_{d(\triangle)}(y_\triangle)$ is known, we arbitrarily choose a point $y'_\triangle \in \triangle'$, and compute $l_s(y_{\triangle'})$ by testing for each site $t \in J(\triangle) \cup J(\triangle')$ if $J(t, s)$ separates $y_\triangle$ and $y_{\triangle'}$ (see the right drawing of Fig. 6). Note that $J(s, d(\triangle))$ separates $y_\triangle$ and $y_{\triangle'}$, and $D(t, s) \cap D(s, d(\triangle)) \subseteq D(t, d(\triangle))$ holds by Lemma 1. Then, we traverse from $\triangle'$ to other trapezoids in C^∇, choose a point for each of them, and compute the corresponding level similarly. All these operations take time proportional to the total conflict size of those traversed trapezoids. Since all involved trapezoids are newly created and tested at most 4 times, the total time is proportional to the total conflict size of newly created trapezoids.

To sum up, the time to update trapezoids is proportional to the number of destroyed and newly created trapezoids and their total conflict size. Actually, the time for $\hat{F}_-^\nabla$, $\hat{F}_3^\nabla$, $\check{F}_+^\nabla$ and $\check{F}_1^\nabla$ is charged to destroyed trapezoids, and the time for F_2^∇ is charged to trapezoids newly created.

4.4 Updating Active Status of Faces

After generating new faces from a face F, we need to remove F, test new faces for being active, and remove the inactive ones, together with their trapezoids. The problem is that a new face may be active because it contains an old active trapezoid $\triangle$ that was *not* in conflict with s, so that we could not afford to visit $\triangle$.

We suggest the following solution. By Lemma 4, sub-faces of F in $\hat{F}$ or $\check{F}$ form a cyclic sequence along the boundary of F, where each sub-face appears once. Let us call a sub-face active if it contains an active trapezoid. We group consecutive inactive sub-faces into a single element and store a cyclic list of such elements and active sub-faces. Then, $\hat{F}$ contains an

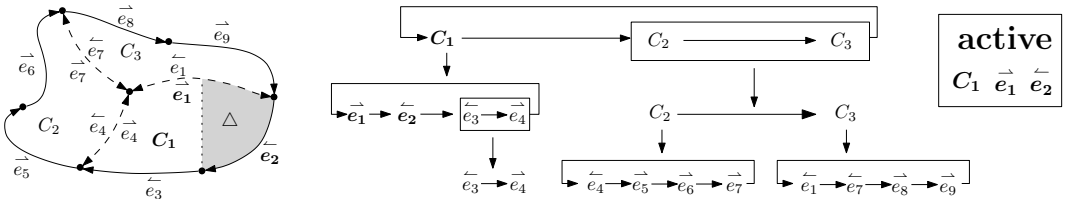


Figure 7 Left: Edges of $\hat{F}$ (dash: inner, solid: outer). Right: a 3-layer data structure for $\hat{F}$.

active sub-face if and only if this list contains more than one entry, or if its only entry is an active sub-face.

Each internal edge appears in two copies, one for the trapezoids on either side. Such a *half-edge* is called active if one of its adjacent trapezoids is active. For each sub-face C we group together inactive half-edges along ∂C, and obtain another cyclic sequence of such elements and active half-edges. Again, we can decide in $O(1)$ time if the sub-face C is active.

These lists are stored in a 3-level data structure, as shown in Fig. 7, one for each active face. If implemented as an enhanced red-black tree [25], operations insert, delete, concatenation, split, and find-set run in $O(\log m)$ amortized time (or faster), where m counts the total number of operations. When considering site set R, m can be upper bounded by $r^{O(1)}$, so that each operation takes amortized time in $O(\log r)$. This data structure allows to determine the activity of a face in constant time. How to maintain it under insertion of a new site s will be stated in a complete version later. We remark that the time to update the data structures is $\log r$ times the number of destroyed and newly created trapezoids.

5 Analysis

The time to insert a new site is proportional to the total conflict size of destroyed and newly created trapezoids plus $\log n$ times their number. However, not all the created trapezoids belong to an active face, so it is not sufficient to analyze active faces in AF_r.

While all essential faces (i. e., those dominating an order-k face) are active, the converse does not hold. But we are able to prove a slightly weaker fact.

▶ **Definition 12.** For a subset R of S of size r, let $M(R)$ denote the collection of faces in $\mathrm{VF}(R)$ each of which dominates a face of $V_{k'}(S)$, where $k - 6c\frac{n}{r}\log r \le k' \le k + 6c\frac{n}{r}\log r$.

We are proving that with high probability $M(R_{r+1})$ contains all faces created during the insertion of site s_{r+1}, both active ones and inactive ones that were discarded.

Since $\mathrm{VF}(R_r)^\nabla$ has $O(r^3)$ trapezoids, the time to insert the $(r+1)^{\mathrm{st}}$ site is trivially $O(nr^3)$. Therefore, if the probability for $M(R)$ to have the above property is high enough, say $1-O(r^{-5})$, the insertion time, provided that $M(R)$ fails to have it, is $O(nr^3)*O(r^{-5}) = O(\frac{n}{r^2})$. Since $\sum_{r=10}^{n} O(\frac{n}{r^2})$ is only $O(n)$, the analysis of the positive case will dominate the overall expectation.

To this end, we define a kind of ϵ-nets for $\epsilon = c\frac{1}{r}\log r$ as follows.

▶ **Definition 13.** For a subset R of S of size r and for a constant $c \ge 22$, R is an ϵ-net if

1. For each trapezoid $\triangle \in \mathrm{VF}^\nabla(R)$, $w(\triangle) \le c\frac{n}{r}\log r$,
2. and for each face $F \in \mathrm{VF}(R)$ where $F \subseteq \mathrm{VF}_j(Q,R)$ and for any two points $x, x' \in F$ where $x \in \mathrm{FVR}(q,Q) \cap \mathrm{VR}(p, R \setminus Q)$ and $x' \in \mathrm{FVR}(q',Q) \cap \mathrm{VR}(p', R \setminus Q)$,

$$\max\{|l_q(x) - l_{q'}(x')|, |l_p(x) - l_{p'}(x')|, l_p(x) - l_q(x)\} \le c\frac{n}{r}\log r.$$

▶ **Lemma 14.** *If R_r is an ϵ-net, then $M(R_{r+1})$ contains all the created faces due to the insertion of s_{r+1}.*

Proof. Consider a face F of $\mathrm{AF}(R_r)$ in conflict with s_{r+1}. It is sufficient to prove that all faces generated from F due to the insertion of s_{r+1} belong to $M(R_{r+1})$. Let F belong to $\mathrm{VR}_j(Q, R_r)$. We will prove that each face C of F_- belongs to $M(R_{r+1})$, and it is symmetric for each face of F_+. (Recall that $F_- = F \cap \mathrm{VR}_j(Q, R_{r+1})$ and $F_+ = F \cap \mathrm{VR}_{j+1}(Q \cup \{s_{r+1}\}, R_{r+1})$.) Take a point $x \in C$ where $x \in \mathrm{VR}(s_{r+1}, R_{r+1} \setminus Q)$ and $x \in \mathrm{FVR}(q, Q)$. Since $C \subseteq F$, we also let x belong to $\mathrm{VR}(p, R_r \setminus Q)$. In other words, $x \in D(s_{r+1}, p)$, so $l_q(x) < l_{s_{r+1}}(x) < l_p(x)$.

Since F is active, there exists a trapezoid $\triangle \in \hat{F}^{\nabla}$ such that there exists a point $y \in \triangle$ with $l_{d(\triangle)}(y) - w(\triangle) \le k$. Since R_r is an ϵ-net, $w(\triangle) \le c\frac{n}{r}\log r$ and $l_q(x) - l_{d(\triangle)}(y) \le c\frac{n}{r}\log r$, we have $l_q(x) \le k + 2c\frac{n}{r}\log r$. Similarly, we can derive $l_p(x) \ge k - 2c\frac{n}{r}\log r$. Since $l_p(x) - l_q(x) \le c\frac{n}{r}\log r$ (by the definiton of an ϵ-net), we have

$$k - 3c\frac{n}{r}\log r \le l_q(x) < l_p(x) \le k + 3c\frac{n}{r}\log r.$$

Finally, since $l_q(x) < l_{s_{r+1}}(x) < l_p(x)$ and $2\frac{n}{r+1}\log(r+1) \ge \frac{n}{r}\log r$, we have

$$k - 6\frac{n}{r+1}\log(r+1) \le l_q(x) < l_{s_{r+1}}(x) < l_p(x) \le k + 6\frac{n}{r+1}\log(r+1),$$

implying that C dominates a face of $V_{k'}(S)$ where $k - 6c\frac{n}{r+1}\log(r+1) \le k' \le k + 6c\frac{n}{r+1}\log(r+1)$ (see Lemma 3) and thus belongs to $M(R_{r+1})$. ◀

Using general ideas by Clarkson and Shor [13] and Haussler and Welzl [15], we analyze the probability of an ϵ-net as follows.

▶ **Lemma 15.** *With probability $1 - O(\frac{1}{r^5})$, a random sample R of S of size r is an ϵ-net.*

By Lemma 15, we can derive the expected size of $|M(R)^{\nabla}|$. The number of trapezoids in $M(R)^{\nabla}$ is proportional to the number of vertices in $M(R)$, so we analyze the latter. It is sufficient to consider the case in which R is an ϵ-net since $|M(R)|$ is trivially $O(r^3)$ and the probability that R is not an ϵ-net is only $O(r^{-5})$, leading to a product $O(r^{-2})$. We mainly prove that a vertex of $M(R)$ is a vertex of $V_m(S)$ where $k - 7c\frac{n}{r}\log r \le m \le k + 7c\frac{n}{r}\log r + 2$. If this claim holds, since $V_m(S)$ has $O(m(n-m))$ vertices [3], the total complexity of all candidates is proportional to the summation of $m(n-m)$ for $k - 7c\frac{n}{r}\log r \le m \le k + 7c\frac{n}{r}\log r + 2$, leading to $O(\frac{n}{r}k(n-k)\log r + \frac{n^3}{r^2}\log^2 r)$. Since each vertex of $\mathrm{VF}(S)$ is a vertex of $\mathrm{VF}(R)$ with probability $O(\frac{r^3}{n^3})$, this implies that the expected number of vertices in $M(R)$ is $O(\frac{r^2}{n^2}k(n-k)\log r + r\log^2 r)$.

The intuitive idea for the claim is to consider a vertex v of a face F in $M(R)$ and to let F belong to $\mathrm{VR}_j(Q, R)$ and v be an intersection between $J(t, t')$ and $J(t, t'')$. Since F is a face of $M(R)$, there exists a point $x \in F$ such that $x \in \mathrm{FVR}(q, Q)$ and $l_q(x) \le k + 6c\frac{n}{r}\log r$. Since R is an ϵ-net, $l_t(v) - l_q(x) \le c\frac{n}{r}\log r$, implying that $l_t(v) \le k + 7c\frac{n}{r}\log r$. Symmetrically, we have $l_t(v) \ge k + 7c\frac{n}{r}\log r - 2$. Since v is an order-$(l_t(v)+2)$ vertex, we prove the claim and conclude the following lemma.

▶ **Lemma 16.** $E[|M(R)^{\nabla}|] = O(\frac{r^2}{n^2}k(n-k)\log r + r\log^2 r)$.

Chazelle et al. [11] proposed an abstract framework for randomized incremental construction, and to adopt their result, we show that $M(R)$ has some monotonicity property.

▶ **Lemma 17.** *For all trapezoids $\triangle \in M^{\nabla}(R_r)$, if $D(\triangle) \subseteq R_{r-1}$, then $\triangle \in M^{\nabla}(R_{r-1})$.*

Proof. Let F be the face in $M(R_r)$ that contains $\triangle$, i.e., $\triangle \in F^\nabla$, and let F be a face of $\mathrm{VR}_j(Q, R)$. Since $J(\triangle) \cap R_r = \emptyset$ and $D(\triangle) \subseteq R_{r-1}$, $\triangle$ must belong to some face in $\mathrm{VF}(R_{r-1})$, and we use F' to denote it. It is clear that $F' \supseteq F$. If $s_r \in Q$, F' is a face of $\mathrm{VR}_{j-1}(Q \setminus \{s_r\}, R_{r-1})$; otherwise, F' is a face of $\mathrm{VR}_j(Q, R_{r-1})$. In either case, F' dominates F. Since $F \in M^\nabla(R_r)$, F dominates a face C of $V_{k'}(S)$ where $k - 6c\frac{n}{r}\log r \leq k' \leq k + 6c\frac{n}{r}\log r$, and since F' dominates F, F' also dominates C. Since $\frac{n}{r-1}\log(r-1) > \frac{n}{r}\log r$ (for $r > 2$), $k - 6c\frac{n}{r-1}\log(r-1) \leq k' \leq k + 6c\frac{n}{r-1}\log(r-1)$, and thus F' belongs to $M(R_{r-1})$, implying that $\triangle \in M^\nabla(R_{r-1})$. ◀

Finally, we analyze the expected construction time. The whole idea is that if R_r is an ϵ-net, we charge the insertion time of s_{r+1} through $M(R_r)$; otherwise, we use $O(r^3 n)$ to bound the insertion time. The latter has been shown to be $O(n)$ over the entire construction. Let T be $\bigcup_{r=10}^{n} M^\nabla(R_r)$. Since each destroyed trapezoid must be created before and $M^\nabla(R_{r+1})$ contains all created trapezoids due to the insertion of s_{r+1} when R_r is an ϵ-net (Lemma 14), the former case during the whole incremental construction is bounded by $O(\sum_{\triangle \in T} \log n + w(\triangle))$. By Lemma 17, we can adopt Chazelle et al.'s framework [11] to prove that

$$E[|T|] = \sum_{r=10}^{n} O(1/r) E[|M(R_r)^\nabla|] \text{ and } E[\sum_{\triangle \in T} w(\triangle)] = \sum_{r=10}^{n} O(n/r^2) E[|M(R_r)^\nabla|].$$

Since $E[|M(R_r)^\nabla|] = O(\frac{r^2}{n^2} k(n-k)\log r + r\log^2 r)$ (by Lemma 16),

$$E[\sum_{\triangle \in T} \log n + w(\triangle))] = \log n \cdot E[|T|] + E[\sum_{\triangle \in T} w(\triangle)] = O(k(n-k)\log^2 n + n\log^3 n).$$

Since we can link all vertices of $V_k(S)$ in $O(k(n-k)\log n)$ time [5], we conclude the following.

▶ **Theorem 18.** *The expected time to compute $V_k(S)$ is $O(k(n-k)\log^2 n + n\log^3 n)$.*

Acknowledgement. We deeply appreciate all valuable comments from the SoCG reviewers. Due to the page limit, we could not address all the comments, but will make the remaining ones clear in the journal version.

References

1 Pankaj K. Agarwal, Mark de Berg, Jirí Matousek, and Otfried Schwarzkopf. Constructing levels in arrangements and higher order Voronoi diagrams. *SIAM J. Comput.*, 27(3):654–667, 1998.

2 Franz Aurenhammer and Otfried Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order Voronoi diagrams. In *Proceedings of the Seventh Annual Symposium on Computational Geometry (SoCG)*, pages 142–151, 1991.

3 Cecilia Bohler, Panagiotis Cheilaris, Rolf Klein, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. On the complexity of higher order abstract Voronoi diagrams. *Computational Geometry*, 48(8):539–551, 2015.

4 Cecilia Bohler and Rolf Klein. Abstract Voronoi diagrams with disconnected regions. *Int. J. Comput. Geometry Appl.*, 24(4):347–372, 2014.

5 Cecilia Bohler, Chih-Hung Liu, Evanthia Papadopoulou, and Maksym Zavershynskyi. A randomized divide and conquer algorithm for higher-order abstract Voronoi diagrams. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC)*, pages 27–37, 2014.

6 Jean-Daniel Boissonnat, Olivier Devillers, and Monique Teillaud. A semidynamic construction of higher-order Voronoi diagrams and its randomized analysis. *Algorithmica*, 9(4):329–356, 1993.

7 Timothy M. Chan. Random sampling, halfspace range reporting, and construction of (<= k)-levels in three dimensions. *SIAM J. Comput.*, 30(2):561–575, 2000.

8 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. In *Proceeding of the 31st International Symposium on Computational Geometry (SoCG)*, pages 719–732, 2015.

9 Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6:485–524, 1991.

10 Bernard Chazelle and Herbert Edelsbrunner. An improved algorithm for constructing k th-order Voronoi diagrams. *IEEE Trans. Computers*, 36(11):1349–1354, 1987.

11 Bernard Chazelle, Herbert Edelsbrunner, Leonidas J. Guibas, Micha Sharir, and Jack Snoeyink. Computing a face in an arrangement of line segments and related problems. *SIAM J. Comput.*, 22(6):1286–1302, 1993.

12 Kenneth L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Computational Geometry*, 2:195–222, 1987.

13 Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.

14 Andreas Gemsa, D. T. Lee, Chih-Hung Liu, and Dorothea Wagner. Higher order city Voronoi diagrams. In *Proceedings of the 13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 59–70, 2012.

15 David Haussler and Emo Welzl. epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

16 Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer, 1989.

17 Rolf Klein, Elmar Langetepe, and Zahra Nilforoushan. Abstract Voronoi diagrams revisited. *Comput. Geom.*, 42(9):885–902, 2009.

18 Rolf Klein, Kurt Mehlhorn, and Stefan Meiser. Randomized incremental construction of abstract Voronoi diagrams. *Comput. Geom.*, 3:157–184, 1993.

19 D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Computers*, 31(6):478–487, 1982.

20 Chih-Hung Liu and D. T. Lee. Higher-order geodesic Voronoi diagrams in a polygonal domain with holes. In *Proceedings of the Twenty-Fourth Annual Symposium on Discrete Algorithms (SODA)*, pages 1633–1645, 2013.

21 Kurt Mehlhorn, Stefan Meiser, and Ronald Rasch. Furthest site abstract Voronoi diagrams. *Int. J. Comput. Geometry Appl.*, 11(6):583–616, 2001.

22 Ketan Mulmuley. *Computational geometry – an introduction through randomized algorithms*. Prentice Hall, 1994.

23 Evanthia Papadopoulou and Marksim Zavershynskyi. On higher order Voronoi diagrams of line segments. *Algorithmica*, 2014. Published on-line.

24 Edgar A. Ramos. On range reporting, ray shooting and k-level construction. In *Proceedings of the Fifteenth Annual Symposium on Computational Geometry (SoCG)*, pages 390–399, 1999.

25 Robert Endre Tarjan and Christopher J. Van Wyk. An $O(n \log \log n)$-time algorithm for triangulating a simple polygon. *SIAM J. Comput.*, 17(1):143–178, 1988.

All-Pairs Minimum Cuts in Near-Linear Time for Surface-Embedded Graphs*†

Glencora Borradaile[1], David Eppstein[2], Amir Nayyeri[3], and Christian Wulff-Nilsen[4]

1 **School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA**
glencora@eecs.oregonstate.edu
2 **Computer Science Department, Donald Bren School of Information and Computer Sciences, University of California, Irvine, USA**
eppstein@uci.edu
3 **School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA**
nayyeria@eecs.oregonstate.edu
4 **Department of Computer Science, University of Copenhagen (DIKU), Copenhagen, Denmark**
koolooz@diku.dk

Abstract

For an undirected n-vertex graph G with non-negative edge-weights, we consider the following type of query: given two vertices s and t in G, what is the weight of a minimum st-cut in G? We solve this problem in preprocessing time $O(n \log^3 n)$ for graphs of bounded genus, giving the first sub-quadratic time algorithm for this class of graphs. Our result also improves by a logarithmic factor a previous algorithm by Borradaile, Sankowski and Wulff-Nilsen (FOCS 2010) that applied only to planar graphs. Our algorithm constructs a Gomory–Hu tree for the given graph, providing a data structure with space $O(n)$ that can answer minimum-cut queries in constant time. The dependence on the genus of the input graph in our preprocessing time is $2^{O(g^2)}$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases minimum cuts, surface-embedded graphs, Gomory-Hu tree

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.22

1 Introduction

In recent years, problems of speeding up graph algorithms by taking advantage of low-genus embeddings onto topological surfaces have become an important subtopic of computational topology [11, 12, 7, 8, 6, 13, 10, 5]; in this paper, we do so for the all-pairs minimum cut problem. In the *all-pairs minimum cut problem* we seek the minimum st-cut for every pair $\{s, t\}$ of vertices in an edge-weighted, undirected graph G. Gomory and Hu [15] showed that these minimum cuts can be represented by a single edge-weighted tree such that:

- the nodes of the tree correspond one-to-one with the vertices of G,

* This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-0963921, CCF-1228639, and CCF-1252833 and by the Office of Naval Research under Grant No. N00014-08-1-1015.

† For the full version of the paper see http://arxiv.org/abs/1411.7055.

© Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen;
licensed under Creative Commons License CC-BY

- for any distinct vertices s and t, the minimum-weight edge on the unique s-to-t path in the tree has weight equal to the min st-cut in G, and
- removing this minimum-weight edge from the tree creates a partition of the nodes into two sets corresponding to a min st-cut in G.

We call such a tree a minimum cut tree; it is also known as a Gomory–Hu tree or cut-equivalent tree. Gomory and Hu showed how to find this tree with $n-1$ calls to a minimum cut algorithm by building up a collection of nested cuts, in each step adding a minimum st-cut that separates a previously-unseparated pair of vertices.

New results. We provide the first subquadratic algorithm for all-pairs minimum cuts in bounded-genus graphs. We can find the Gomory–Hu tree of a graph of genus g in time $2^{O(g^2)}n\log^3 n$, giving a data structure of size $O(n)$ which can answer minimum-cut queries in constant time. The best previous method for this class of graphs uses the standard Gomory–Hu algorithm and has a running time of $O(g^8n^2\log^2 n\log^2 C)$ (for integer edge weights summing to C) using the best maximum-flow algorithm to find minimum cuts [8] or $2^{O(g)}n^2\log n$ using the best minimum-cut algorithm [12] for graphs of bounded genus. Our result hinges in part on an improvement in the time for constructing Gomory–Hu trees in planar graphs. Borradaile, Sankowski and Wulff-Nilsen showed how to solve this problem in $O(n\log^4 n)$ time [4]. In this paper, we improve this running time to $O(n\log^3 n)$ time (Section 4). Due to space constraints, some of the proofs are deferred to the full version.

From planar to bounded genus. We reduce the problem of computing the minimum-cut tree in a graph of genus g to the same problem in a set of $2^{O(g^2)}$ planar graphs. The minimum cut, viewed as an even subgraph of the dual graph, is a collection of cycles belonging to one of 2^{2g} homology classes. Our main observation is that one can reduce the problem of finding a minimum cut that is composed of dual cycles in certain homology classes to a planar problem *before* taking into account the vertices that should be separated. This allows us to find a cut tree whose cuts are composed of dual cycles in certain homology classes. We describe this reduction in Section 3. Through this reduction, we compute minimum cut trees in $2^{O(g^2)}$ different planar graphs such that the minimum st-cut in the original graph is represented in at least one of these $2^{O(g^2)}$ cut trees. Although this would already solve the minimum cut query problem, we also show in the full version of this paper how to produce a single minimum cut tree for the original graph, by merging these cut trees in a way that preserves minimum cuts in time $O(kn\log^2 n)$ where k is the number of cut trees to merge. This tree-merging algorithm does not rely on the surface embedding. Note that one could use the Gomory-Hu algorithm of Gusfield to merge these trees, but Gusfield's algorithm has an $O(n^2)$ overhead independent of the method for finding minimum cuts [16].

Planar speed-up. The algorithm of Borradaile et al. [4] for planar graphs implements the Gomory–Hu algorithm by using Miller's recursive cycle separator decomposition [20] to guide the selection of pairs of vertices to separate. Starting with the leaf-most pieces of the decomposition, the algorithm separates all pairs of vertices in a piece that are not yet separated. We refer to this algorithm as the *cycle-based* algorithm as it works in the dual graph finding minimum separating cycles of pairs of faces.

We improve the running time of this algorithm by addressing two bottlenecks. The first bottleneck is in finding a separating cycle. The cycle-based algorithm incurs an $O(\log^3 n)$ factor per cycle; multiplied by the depth of the recursive decomposition of the planar graph this results in an $O(\log^4 n)$ factor in the overall runtime. Instead, we find a minimum cut by

way of first computing a maximum flow, using the recursive flow techniques [2, 19], which, surprisingly (because max flow computations usually dominate minimum cut computations), reduces the overhead per cycle to $O(\log^2 n)$. We refer to this algorithm as the *flow-based* algorithm (Section 4.4). The second bottleneck is in adding a cut to the collection; we improve the overhead from an amortized $O(\log^4 n)$ per cut to an amortized $O(\log^3 n)$ per cut (Section 4.5).

Minimum cycle bases. By duality [17], our algorithm also finds the cycle weights in a minimum cycle basis of a planar graph in the same time bound; in fact, the planar all pairs min cut algorithm is really a minimum cycle basis algorithm in the dual graph. However, in graphs of higher genus and even in toroidal graphs, the minimum cycle basis appears to be mostly unrelated to the dual of minimum cuts. Borradaile et al. [1] study computing the minimum cycle bases and the minimum homology basis on surface embedded graphs.

2 Preliminaries

We consider a graph G with n vertices with a cellular embedding on an orientable surface of genus g.[1] In this paper, we use cycle in its topological sense, that is a closed curve; graph theoretically, a cycle in this sense corresponds to a closed walk. We specify simple cycle when we refer to a closed curve that visits each point (vertex, edge) at most once.

Duality. For every connected, surface embedded graph G (the *primal*) there is another connected graph embedded on the same surface, the *dual* G^*. The faces of G are the vertices of G^* and vice versa. The edges of G correspond one-for-one with the edges of G^*: for each edge e in G, there is an edge e^* in G^* whose endpoints correspond to the faces of G incident to e. Dual edges inherit the weight of the corresponding primal edges; namely, $w(e^*) = w(e)$.

Surgery. For a cycle C we define the operation of *cutting* along C in G and denote it $G \not\downarrow C$. $G \not\downarrow C$ is the graph obtained by cutting along C in the drawing of G on the surface, creating two copies of every edge in C. The edges in the copies of C inherit the weights of the original edges. We view $G \not\downarrow C$ as being embedded on a surface with two punctures, corresponding to the two resulting copies of C. Similarly, for a path P with endpoints on different boundary components of G, we define cutting along P in G and denote it $G \not\downarrow P$. We view $G \not\downarrow P$ as being embedded on a surface with the boundary components containing P's endpoints unified.

$\mathbb{Z}_2$-homology. $\mathbb{Z}_2$ homology as we use it in this paper is described by Erickson and Nayyeri [12]; we refer the reader to their paper for formal definitions of the following. For further background on surface topology and homology we refer the reader to Hatcher [18]. Here, when we talk about homology we mean homology with $\mathbb{Z}_2$ coefficients.

A subgraph is called *even* if it has even degree at every vertex, or equivalently if it is the edge-disjoint union of simple cycles. An even subgraph is *null-homologous* if it is the boundary of a union of faces in G. Two even subgraphs are homologous if their symmetric difference is null-homologous. A *homology basis* is a set $\{C_1, C_2, \ldots, C_{2g}\}$ of simple cycles in G^* that generates the homology class of all cycles of G^*; it can be constructed in linear

[1] An embedding is cellular if every face is a topological disk. If G has any embedding on a surface of genus g, then the rotation system of the embedding gives a cellular embedding of G on a surface of genus at most g.

time [12]. The *signature* of an edge $[e]$ is defined as a $2g$-bit vector, whose ith bit is 1 if and only if $e^* \in C_i$. Given two faces a and b and an a^*-to-b^* path P in G^*, $[e]^{ab}$ is the extended $(2g+1)$-bit vector whose first bit is 1 if and only if $e^* \in P$ and whose second to $(2g+1)$st bits are $[e]$. Note that $[e]^{ab}$ depends on the choice of P. For any subgraph $X \subseteq G$ we define $[X] = \bigoplus_{e \in X} [e]$ and $[X]^{ab} = \bigoplus_{e \in X} [e]^{ab}$. Two even subgraphs X and X' are homologous if and only if $[X \oplus X'] = [X] \oplus [X'] = 0$. We use $\oplus$ to denote the symmetric difference of sets and the exclusive or of binary numbers.

Minimum cuts and minimum separating subgraphs. The dual of a minimum st-cut is the minimum even subgraph X that is null-homologous and such that $[X]_0^{s^*t^*} = 1$, i.e. $[X]^{s^*t^*} = [1\,0 \cdots 0]$ (Lemma 3.1 [7]). We will call this the minimum s^*t^*-separating subgraph. In a surface of genus g, a minimum separating subgraph is composed of at most $g+1$ simple cycles. In particular, a minimum separating subgraph in a planar graph is a simple cycle. The all-pairs minimum cut problem is equivalent to the all-pairs minimum separating subgraph problem in the dual.

Faces and boundaries. For a set of faces F, we define ∂F to be the boundary of F, that is, the set of edges that bound faces of F and $\bar{F}$. Finally, for any graph G, we use $F(G)$ to denote the set of faces of G. We additionally consider a special type of faces, *boundary* faces, which correspond to the boundary of punctures in the surface and are introduced over the course of our algorithm from the cutting along operation. If $H = G \not\downarrow X$ for some set of edges X, then $F(H)$ is $F(G)$ plus a set of additional boundary faces, bounded by edges of X.

▶ **Claim 1.** Let X be a set of edges such that $G \not\downarrow X$ has one more boundary face than G. Let S be an ab-separating subgraph in $G \not\downarrow X$ for non-boundary faces $a, b \in F(G \not\downarrow X)$. Then S is ab-separating in G.

Tight cycles and paths. We say that a cycle C is *tight* if it is the shortest cycle with $\mathbb{Z}_2$-homology signature $[C]$. Tight cycles in all homology classes can be found in time $2^{O(g)} n \log n$ time [12]. We say that an x-to-y path P is tight if $P \cup xy$ is the shortest cycle with signature $[P \cup xy]$ in the graph $G \cup xy$ where xy is embedded on a handle added to the surface connecting a face incident with x to a face incident with y. Note that the faces may coincide, and we may need to add extra edges (with large weights to ensure they are not part of any tight cycle) to make the embedding cellular. In this way, a tight path can be found in the same time bound as a tight cycle.

Crossing or non-crossing. Let H_1 and H_2 be two subsets of edges. We say that H_1 *crosses* H_2 if there is a subset of edges S of $H_1 \cap H_2$ such that contracting S results in a vertex s such that the edges e_1, e_2, e_1', e_2' are incident to s and are in this clockwise order around s with $e_1, e_1' \in H_1$ and $e_2, e_2' \in H_2$. Otherwise, H_1 and H_2 do not cross.

A cycle C is *non-self-crossing* if no two subpaths of C cross. A cycle is *weakly simple* if it is non-self-crossing and traverses each edge at most once. A *cycle decomposition* of an even subgraph H is a partition $\mathcal{C} = \{C_1, C_2, \ldots\}$ of the edges of H such that each cycle in $\mathcal{C}$ is simple and no two cycles in $\mathcal{C}$ cross. Chambers, Erickson and Nayyeri (Lemma 3.2 [7]) prove that every even subgraph of a surface-embedded graph has a cycle decomposition.

Cuts. By abuse of notation, we use an xy-cut to refer to two notions: it can be either a subset C of edges whose removal from the graph separates x and y, or a bipartition (X, Y) of the vertices such that $x \in X$ and $y \in Y$. The edge subset C is the set of edges with one

endpoint in X and the other in Y. We say that a cut (A, B) *crosses* a cut (X, Y) if neither $A \subseteq X$ nor $A \subseteq Y$.

Uniqueness of minimum cuts and shortest paths. We assume that minimum cuts do not cross. That is, if (X, Y) is a minimum xy-cut and (A, B) is a minimum ab-cut, then (X, Y) and (A, B) do not cross. This is automatically true when all minimum cuts are unique, which, in turn can be assumed to be true with high probability by randomly perturbing the edge weights slightly [4, 21]. This perturbation also allows us to assume that shortest paths in the dual graph are unique, which is required for both the cycle- and flow-based planar algorithms.

Cut trees. For a graph G, an edge-weighted tree T on the same vertex set as G is a cut-tree for G if, for every edge e in T, the weight of the cut in G corresponding to the bipartition of the vertices given by $T \setminus \{e\}$ is $w(e)$. T is a *minimum* cut-tree of G if for every pair of vertices x, y the minimum xy-cut in T is the same (in value and bipartition of vertices) as the minimum xy-cut in G.

Region trees. Within our algorithms, it is convenient to associate with a cut tree T a *region tree* R [4]. R is the unique tree obtained from T by adding a leaf vertex to every node of T. Thus, R has one leaf and one internal node for each of the n vertices of T. The internal edges of R are exactly the edges of T. Such a region tree is called a *complete* region tree. The cut tree T can be recovered from R by contracting all the edges incident to leaves.

A *partial* region tree is any tree that can be obtained from a region tree by contracting a subset of internal edges. A region tree may refer either to a complete region tree or to a partial region tree, and should be clear from context. Contracting the leaf edges of a partial region tree and mapping subsets of V to the resulting nodes gives a representation of a *partial* cut tree as maintained throughout the standard Gomory–Hu algorithm.

If a region tree R is rooted at an arbitrary internal node, then each of its internal nodes u represents a *region* which is the graph obtained from G by contracting, for each non-leaf child of u in T the leaves below it to a single vertex, and similarly contracting all leaves of T not below u to a single vertex (this definition is the same as in [4] except that we define a region in the primal rather than the dual graph). For a non-root, internal node v of R, we define S_v to be the set of leaf descendents of v; S_v is one side of the cut $T \setminus \{e\}$ where e is the parent edge of v. The root of R is the region that corresponds to the entire graph G.

Cartesian trees. Our query data structure is based on a *Cartesian tree* for an edge-weighted tree T. This is a binary tree in which the interior nodes represent edges of T and the leaves represent vertices of T. The root node represents the lightest edge of T and its two children are constructed recursively from the two subtrees formed from T by removing this lightest edge. The minimum cut between any two vertices in T can then be found by answering a lowest common ancestor query between the corresponding two leaves of the Cartesian tree. Given an edge-weighted tree T whose edges have been sorted by weight, the Cartesian tree for T can be constructed and processed for constant-time lowest common ancestor queries in time and space $O(n)$ [9].

3 Reduction from bounded genus to planar

We show how to reduce the all-pairs minimum cut problem for a surface-embedded graph G to the planar case. We do so by recursively cutting along (i) a tight cycle that belongs to

some minimum cut or (ii) a tight cycle and a tight path connecting sides of the tight cycle that some minimum cut does not cross.

3.1 Reducing the genus

If S is a minimum ab-separating subgraph in graph G and C is a cycle in a cycle decomposition of S then C is a minimum ab-separating cycle in $G \not\curvearrowleft (S \setminus C)$ for otherwise there would be a cheaper minimum ab-separating subgraph G. The following lemma allows us to reduce the problem of finding S to that of finding C. That is, it allows us to find $S \setminus C$, and, in particular to do so without specifying the faces we wish to separate.

▶ **Lemma 2.** *Let S be the minimum ab-separating subgraph and let $C_1, C_2, \ldots, C_t$ be a cycle decomposition of S with cycles ordered by increasing cost. Then C_i is the cheapest cycle having $\mathbb{Z}_2$-homology signature $[C_i]$ for $i = 1, \ldots, t-1$. Moreover, for any $1 \leq h \leq t$, $\bigcup_{j=h}^{t} C_j$ is a minimum ab-separating subgraph in $G \not\curvearrowleft \bigcup_{i=1}^{h-1} C_i$.*

Proof. If $t = 1$, the lemma is trivially true. Herein, assume $t \geq 2$. For a contradiction, let C_i be the first cycle such that C_i is not the cheapest cycle having its homology signature (with $i < t$). Let C_i' be the cheapest cycle such that $[C_i'] = [C_i]$. We have:

$$w(C_i') < w(C_i) \leq w(C_{i+1}) \leq w(S \setminus C_i). \tag{1}$$

Since S is null-homologous, $[C_i'] = [C_i] = [S \setminus C_i]$. So, both $C_i' \oplus C_i$ and $C_i' \oplus (S \setminus C_i)$ are separating. Since $[S]_0^{ab} = 1 = [C_i]_0^{ab} \oplus [S \setminus C_i]_0^{ab}$ exactly one of $[C_i]_0^{ab}$ or $[S \setminus C_i]_0^{ab} = 1$. That is, exactly one of $[C_i']_0^{ab} \oplus [C_i]_0^{ab}$ or $[C_i']^{ab} \oplus [S \setminus C_i]_0^{ab} = 1$. Thus, either $C_i' \oplus C_i$ or $C_i' \oplus (S \setminus C_i)$ is ab-separating. By Equation 1, both $w(C_i' \oplus C_i) < w(S)$ and $w(C_i' \oplus (S \setminus C_i)) < w(S)$. This contradicts that S is the minimum ab-separating subgraph.

Since S is ab-separating in G, $H = \bigcup_{j=h}^{t} C_j$ is ab-separating in $G' = G \not\curvearrowleft \bigcup_{i=1}^{h-1} C_i$. For a contradiction, suppose that $H' \neq H$ is the minimum ab-separating cycle in G', that is $w(H') < w(H)$. It follows that $S' = (\bigcup_{i=1}^{h-1} C_i) \cup H'$ is ab-separating in G, and $w(S') < w(S)$, contradicting that S is the minimum ab-separating subgraph in G. ◀

The following lemma is stronger than Lemma 6.1 of Erickson et al. [11], but the proof technique is similar.

▶ **Lemma 3.** *Let S be the minimum ab-separating subgraph. Let A be any subgraph that does not cross S, and let H be the minimum subgraph such that $A \oplus H$ is null-homologous. Then H does not cross S.*

Proof. The even subgraph S separates the faces of G into two sets F_a and F_b. Without loss of generality, we assume that A and a are on the same side of S (i.e. contained in the same piece of $G \not\curvearrowleft S$), $a \in F_a$ and $b \in F_b$.

$A \oplus H$ is a null-homologous even subgraph, so it separates the faces of G into subsets J_b and J_b'. Assume, without loss of generality, that $b \in J_b$. Let $F = F_b \cap J_b$ and note that $a \notin F$ and $b \in F$, so $\partial(F)$ is ab-separating. Also, each edge of $\partial(F)$ is on the boundary of a face of F_b, so it does not cross S. Finally, $b \in F$, therefore, $\partial(F)$ and b are on the same side of S. In the rest of the proof, we show that $\partial(F)$ must be identical to S, thus H does not cross S.

Since S is the minimum ab-separating subgraph,

$$w(\partial F) \geq w(S). \tag{2}$$

Each boundary edge of $F = F_b \cap J_b$ should be a boundary edge in at least one of F_b and J_b. Additionally, no boundary edge of F belongs to $A \backslash S$ as A and b (so A and $\partial(F)$) are on different sides of S. Thus, $\partial F \subseteq H \cup S$.

Let $H' = H \oplus \partial F \oplus S$. Note that $H' \subset H \cup S$ (since $\partial F \subseteq H \cup S$). Because ∂F and S are boundaries of sets of faces, they are both null-homologous, so $[H'] = [H]$. Since H is minimum,

$$w(H') \geq w(H). \tag{3}$$

Finally, we show the following inequality by bounding the contribution of each edge $e \in H \cup S$ to the sides of the inequality.

$$w(\partial F) + w(H') \leq w(S) + w(H). \tag{4}$$

- If $e \in S \cap H$, then e contributes $2w(e)$ to the right side. Since, by the construction, at most one copy of each edge is included in each of ∂F and H', e can contribute at most $2w(e)$ to the left side.
- If $e \in S \oplus H$, then e is in exactly one of ∂F and H' by the definition of H' ($H' = \partial F \oplus H \oplus S$). In this case, e contributes exactly $w(e)$ to both sides of the inequality.

Therefore, all Inequalities (2), (3) and (4) must be equalities. In particular, $w(\partial F) = w(S)$. Thus, the uniqueness of the minimum cut (in the dual graph) implies that ∂F and S are identical, which in turn implies that H does not cross S. ◀

▶ **Lemma 4.** *Let S be a minimum ab-separating subgraph. Let P be an x-to-y path that does not cross S. Let H be the minimum weight subgraph such that $H \oplus P$ is a null-homologous even subgraph. H contains a tight x-to-y path P' that does not cross S.*

Proof. For $H \oplus P$ to be an even subgraph, all vertices of H except x and y must be even degree and x and y must have odd degree. Therefore, x and y must be in the same connected component C_{xy} of $H \oplus P$. Further, C_{xy} has an Eulerian x-to-y walk (i.e., a walk that uses every edge exactly once), and so C_{xy} can be decomposed into an x-to-y path P' and a set $\mathcal{C}_{xy}$ of cycles by iteratively removing cycles from the walk.

Since H is the minimum weight subgraph such that $H \oplus P$ is null-homologous, P' must be a tight path and all the cycles in $\mathcal{C}_{xy}$ must be tight, for otherwise, one could replace P' or a cycle in $\mathcal{C}_{xy}$ with a cheaper path or cycle while not changing $[H \oplus P]$.

By Lemma 3, H does not cross S, and so in particular, P' does not cross S, giving this lemma. ◀

Although cutting along a cycle C that does not cross a minimum ab-separating subgraph S reduces the genus of the surface, S may not be a minimum ab-separating subgraph in $G \not\downarrow C$ since the minimum ab-separating subgraph in $G \not\downarrow C$ may separate copies of C and so may not be separating in G. To overcome this, we use Lemma 4 to witness a tight path P connecting the two copies of C in $G \not\downarrow C$. $G \not\downarrow (P \cup C)$ then has one boundary formed by two copies of each edge in $P \cup C$. By Claim 1, a separating subgraph in $G \not\downarrow (P \cup C)$ will also be separating in G.

▶ **Lemma 5.** *Let G be a graph embedded on a surface with b boundaries and genus g. Let S be the minimum ab-separating subgraph of G. If S is composed of at most g cycles then there is a tight cycle C and a tight C-to-C path P such that*

1. *$G \not\downarrow (C \cup P)$ is embedded on a surface of genus $g - 1$ with $b + 1$ boundaries and*
2. *neither C nor P crosses S.*

Moreover, S is the minimum ab-separating subgraph in $G \not\downarrow (C \cup P)$.

Proof. Since S has at most g cycles, there is a non-separating cycle C' in $G \not\downarrow S$. Let H' be the shortest even subgraph that is homologous to C', and let C be any cycle in the cycle decomposition of H'. By Lemma 3, H' does not cross S, so, in particular, C does not cross S. Since C is non-separating, $G \not\downarrow C$ has genus one less than G, or $g-1$. Taking the copies of C in $G \not\downarrow C$ to be boundaries, $G \not\downarrow C$ has $b+2$ boundaries.

Since C is non-separating for $G \not\downarrow S$, there exists a path P' between the two copies of C in $G \not\downarrow C$ that does not cross S. Let H be the minimum subgraph such that $P' \oplus H$ is null-homologous. By Lemma 4, H contains a tight path P that does not cross S and connects the two copies of C in $G \not\downarrow C$. Then $G \not\downarrow C \not\downarrow P$ has genus $g-1$ and $b+1$ boundaries.

Since S does not cross C and P, it is ab-separating in $G' = G \not\downarrow (C \cup P)$. For a contradiction, suppose that $S' \neq S$ is the minimum ab-separating subgraph in G', that is $w(S') < w(S)$. By Claim 1, S' is ab-separating in G, which contradicts the assumption that S is the minimum ab-separating subgraph in G. ◀

We call the cycle and path described by Lemma 5 a *tight cycle-path pair*. The following lemma ensures the possibility of reducing the genus of a graph, while looking for a minimum ab-separating subgraph, by cutting a long either a tight cycle or a tight cycle path pair.

▶ **Lemma 6.** *Let S be a minimum ab-separating subgraph in a graph G embedded in a surface of genus g. If $g \geq 1$, at least one of the following conditions holds.*

1. *There is a tight cycle $C \subsetneq S$. In this case, $S \backslash C$ is a minimum ab-separating subgraph in $G \not\downarrow C$.*
2. *There is a tight cycle-path pair (C, P) in G such that C and P do not cross S. In this case, S is a minimum ab-separating subgraph in $G \not\downarrow (C \cup P)$.*

3.2 A collection of planar problems

We recursively use Lemma 6 to construct a set of planar graphs, each annotated with a set of non-separating cycles of the original graph, that collectively contain the minimum ab-separating subgraphs for all pair of faces, a and b. Starting with graph G of genus g, we create a set of new graphs with genus $g-1$, each obtained by either cutting along a tight cycle C that will belong to the separating subgraphs in the derived graphs (as per Lemma 6, part (1)) or by cutting along a tight cycle-path pair that will not cross the separating subgraphs of the derived graphs (as per Lemma 6, part (2)).

We use $\mathrm{sep}(H, a, b)$ and $w(\mathrm{sep}(H, a, b))$ to refer to the minimum ab-separating subgraph in H and its weight, respectively. Let $\mathcal{C}$ be the set of all tight cycles in G. Note that $|\mathcal{C}| = 2^{2g}$ by the definition of $\mathbb{Z}_2$-homology. Let $\mathcal{CP}$ be the set of all tight cycle-path pairs. For each tight cycle C and each homology class h in $G \not\downarrow C$, $\mathcal{CP}$ contains one pair (C, P), where P is the shortest path in homology class h that connects copies of C in $G \not\downarrow C$. Therefore, $|\mathcal{CP}| \leq 2^{2g} \times 2^{2g} = 2^{4g}$. We find the following lemma helpful for computing $\mathcal{C}$ and $\mathcal{CP}$.

▶ **Lemma 7** (Erickson and Nayyeri [12], Theorem 6.3)**.** *Let G be an undirected graph with nonnegative edge weights, cellularly embedded on a surface of genus g with b boundary components. A minimum-weight even subgraph and a minimum weight cycle or path (with endpoints on the boundary) of G in every $\mathbb{Z}_2$-homology class can be computed in $2^{O(g+b)} n \log n$ time.*

▶ **Lemma 8.** *The sets $\mathcal{C}$ and $\mathcal{CP}$ can be computed in $2^{O(g)} n \log n$ time.*

In the following, the positive real numbers are the annotations to the planar graphs that correspond to the set of tight cycles that are cut along that should belong to the separating subgraphs (as per Lemma 6, part (1)).

▶ **Lemma 9.** *Let G be a graph embedded in a surface of genus g. There exist a set $\mathcal{H}$ of at most 2^{2g^2} planar graphs, each annotated with a set of at most g cycles, such that:*
1. *For any $(H,\mathcal{C}) \in \mathcal{H}$, $F(H)$ is $F(G)$ plus a set of boundary faces.*
2. *For any $(H,\mathcal{C}) \in \mathcal{H}$, for any $a,b \in F(G)$, $w(\mathrm{sep}(H,a,b)) + w(\mathcal{C}) \geq w(\mathrm{sep}(G,a,b))$.*
3. *For any $a,b \in F(G)$ there exists $(H,\mathcal{C}) \in \mathcal{H}$ such that $\mathrm{sep}(H,a,b) \cup \mathcal{C} = \mathrm{sep}(G,a,b)$.*

Moreover, $\mathcal{H}$ can be computed in $2^{O(g^2)}n\log n$ time.

Proof. We use induction on the genus of G. For $g = 0$, $\mathcal{H} = \{(G,\emptyset)\}$ and the properties of the lemma trivially hold.

Let $\mathcal{C}_G$ and $\mathcal{CP}_G$ be the set of tight cycles and tight cycle-path pairs for graph G of genus g. For any $C \in \mathcal{C}_G$, $G \not\downarrow C$ has genus $g-1$; let $\mathcal{H}_{G\not\downarrow C}$ be the set of at most $2^{2(g-1)^2}$ annotated planar graphs guaranteed by the inductive hypothesis for the graph $G \not\downarrow C$. For any $(C,P) \in \mathcal{CP}_G$, $G \not\downarrow (C \cup P)$ has genus $g-1$; let $\mathcal{H}_{G\not\downarrow(C\cup P)}$ be the set of at most $2^{2(g-1)^2}$ annotated planar graphs guaranteed by the inductive hypothesis for the graph $G \not\downarrow (C \cup P)$. Let

$$\mathcal{H} = \left(\bigcup_{C\in\mathcal{C}_G} \{(H,\mathcal{C}\cup C) : (H,\mathcal{C}) \in \mathcal{H}_{G\not\downarrow C}\} \right) \cup \left(\bigcup_{(C,P)\in\mathcal{CP}_G} \mathcal{H}_{G\not\downarrow(C\cup P)} \right).$$

Since $|\mathcal{H}_{G\not\downarrow C}| \leq 2^{2(g-1)^2}$ and $|\mathcal{H}_{G\not\downarrow(C\cup P)}| \leq 2^{2(g-1)^2}$ by the inductive hypothesis,

$$|\mathcal{H}| = 2^{2g} \times 2^{2(g-1)^2} + 2^{4g} \times 2^{2(g-1)^2} \leq 2 \times 2^{4g} \times 2^{2(g-1)^2} \leq 2^{2g^2}.$$

Let $T(n,g)$ be the running time of our algorithm to compute $\mathcal{H}$ (for a graph of genus g with n faces). Lemma 8 implies that $\mathcal{C}$ and $\mathcal{CP}$ can be computed in $2^{O(g)}n\log n$ time. Since $G \not\downarrow C$ has two more faces that G and $G \not\downarrow (C \cup P)$ than G,

$$T(n,g) \leq 2^{O(g)}T(n+2,g-1) + 2^{O(g)}n\log n = 2^{O(g^2)}n\log n.$$

We show that $\mathcal{H}$ satisfies the remaining properties of the lemma.

Property (1). For any C (resp. (C,P)) we have $F(G \not\downarrow C) \supseteq F(G)$ (resp. $F(G \not\downarrow (C \cup P)) \supseteq F(G)$ since cutting along C (resp. $C \cup P$) only adds boundary faces to G. Thus, this property holds by induction.

Property (2). Let $a,b \in F(G)$. We prove that Property (2) holds for the annotated graphs derived from $\mathcal{H}_{G\not\downarrow C}$ and from $\mathcal{H}_{G\not\downarrow(C\cup P)}$ separately.

Consider any $C \in \mathcal{C}$. If S is an ab-separating subgraph in $G \not\downarrow C$ then $S \cup C$ is ab-separating in G by Claim 1. In particular, the minimum ab-separating subgraph in G has length at most $w(\mathrm{sep}(G \not\downarrow C,a,b)) + w(C)$: $w(\mathrm{sep}(G \not\downarrow C,a,b)) + w(C) \geq w(\mathrm{sep}(G,a,b))$. By induction, we have that for any $(H,\mathcal{C}) \in \mathcal{H}_{G\not\downarrow C}$, $w(\mathrm{sep}(H,a,b)) + w(\mathcal{C}) \geq w(\mathrm{sep}(G \not\downarrow C,a,b))$. Therefore, since $(H,\mathcal{C}\cup C) \in \mathcal{H}$, and by combining the previous two inequalities gives $w(\mathrm{sep}(H,a,b)) + w(\mathcal{C}\cup C) = w(\mathrm{sep}(H,a,b)) + w(\mathcal{C}) + w(C) \geq w(\mathrm{sep}(G \not\downarrow C,a,b)) + w(C) \geq w(\mathrm{sep}(G,a,b))$.

Likewise, for any $(C,P) \in \mathcal{CP}$, if S is an ab-separating subgraph in $G\not\downarrow(C\cup P)$ then it must be ab-separating in G, also by Claim 1. In particular, the minimum ab-separating subgraph in G is not heavier than S: $w(\mathrm{sep}(G \not\downarrow (C \cup P),a,b)) \geq w(\mathrm{sep}(G,a,b))$. By the inductive hypothesis, for any $(H,\mathcal{C}) \in \mathcal{H}_{G\not\downarrow(C\cup P)}$, we have $\mathrm{sep}(H,a,b) + w(\mathcal{C}) \geq \mathrm{sep}(G \not\downarrow (C \cup P),a,b)$. Since $(H,\mathcal{C}) \in \mathcal{H}$ and by combining these two inequalities we have $\mathrm{sep}(H,a,b) + w(\mathcal{C}) \geq w(\mathrm{sep}(G,a,b))$.

Property (3). Let S be the minimum ab-separating subgraph in G. At least one of the conditions of Lemma 6 must hold.

If the first condition of Lemma 6 holds, then there is a tight cycle $C \in \mathcal{C}$ such that $S \backslash C$ is a minimum ab-separating subgraph in $G \not\downarrow C$. By the induction hypothesis, there is an annotated planar graph $(H, \mathcal{C}) \in \mathcal{H}_{G \not\downarrow C}$ such that $\mathrm{sep}(H, a, b) \cup \mathcal{C} = \mathrm{sep}(G \not\downarrow C, a, b) = S \backslash C$. Since $(H, \mathcal{C} \cup C) \in \mathcal{H}$ and $\mathrm{sep}(H, a, b) \cup \mathcal{C} \cup C = \mathrm{sep}(G \not\downarrow C, a, b) \cup C = S = \mathrm{sep}(G, a, b)$, we achieve the desired property.

If the second condition of Lemma 6 holds, then there is a tight cycle-path pair $(C, P) \in \mathcal{CP}$ such that S is a minimum ab-separating subgraph in $G \not\downarrow (C \cup P)$. By the induction hypothsis, there is an annotated planar graph $(H, \mathcal{C}) \in \mathcal{H}_{G \not\downarrow (C \cup P}$ such that $\mathrm{sep}(H, a, b) \cup \mathcal{C} = \mathrm{sep}(G \not\downarrow (C \cup P), a, b)$. Since $(H, \mathcal{C}) \in \mathcal{H}$ and $\mathrm{sep}(H, a, b) \cup \mathcal{C} = \mathrm{sep}(G \not\downarrow (C \cup P), a, b) = S = \mathrm{sep}(G, a, b)$, we achieve the desired property. ◀

3.3 The algorithm

We compute the set of annotated planar graphs $\mathcal{H}$ as per Lemma 9, and then for each $(H, \mathcal{C}) \in \mathcal{H}$, we solve the all-pairs separating cycle problem in the graph H; note that it is only necessary to compute the minimum separating cycle for all pairs of non-boundary faces of H. We represent the set of minimum separating cycles of H using a minimum cut-tree T_H for the dual of the graph H. Since H is planar, we can compute T_H in $O(n \log^3 n)$ time using the planar algorithm (Section 4). We then increase the weight of each edge in T_H by $w(\mathcal{C})$.

Let $\mathcal{T}$ be the collection of all such cut trees. Note that $|\mathcal{T}| = |\mathcal{H}| = 2^{O(g^2)}$. All cut trees in $\mathcal{T}$ can be computed in $2^{O(g^2)} n \log^3 n$ time, and can be stored using $2^{O(g^2)} n$ space. For faces a and b of G, a and b are non-boundary faces of H (Lemma 9, Property (1)). The minimum ab-separating cycle in H corresponds to the minimum a^*b^*-cut represented by T_H, and can be determined in $O(1)$ time using a Cartesian tree representation of T_H.

By Lemma 9 (Property (2) and (3) and the weight added to each edge of T_H), the weight of the minimum ab-separating cycle is given by

$$\min_{T \in \mathcal{T}} w(\text{minimum } a^*b^*\text{-cut in } T).$$

Therefore, the minimum ab-separating cycle can be determined in $O(|\mathcal{T}|) = O(|\mathcal{H}|) = 2^{O(g^2)}$ time.

In the full version, we show how to merge k cut trees to preserve minimality in time $O(kn \log^2 n)$ time. That is, we show how to, from the set $\mathcal{T}$, compute a single min-cut tree T for minimum separating subgraphs in G (or minimum cuts in G^*). The total time to compute T by this method is $2^{O(g^2)}(n \log^2 n)$ plus the time to compute $\mathcal{T}$, or $2^{O(g^2)}(n \log^3 n)$. This will give:

▶ **Theorem 10.** *Let G be a graph embedded on a surface of genus g. The Gomory-Hu tree of G can be computed in $2^{O(g^2)} n \log^3 n$ time.*

4 Speed-up for planar graphs

Borradaile, Sankowski and Wulff-Nilsen [3, 4] gave an $O(n \log^4 n)$-time *cycle-based* algorithm for computing a Gomory–Hu tree of a planar graph $G = (V, E)$, assuming that minimum cuts are unique and so any two minimum cuts are guaranteed to nest. The algorithm is guided by a *recursive decomposition* of the graph by small, balanced separators. Working from leaf-to-root in this recursive decomposition, the algorithm finds all the minimum cuts

between unseparated vertices in a *piece* of the decomposition by finding the corresponding minimum separating cycles in the dual graph. The cycles are found by computing shortest paths explicitly within the piece and implicitly outside the piece by relying on precomputed distances outside the piece between all pairs of boundary vertices of the piece represented by an *external dense distance graph*. Computing one cycle incurs a $\log^3 n$ factor in the runtime; combined with the logarithmic depth of the recursive decomposition results in a $\log^4 n$ factor in the running time. We overcome this bottleneck by instead computing the maximum flow between each pair of unseparated vertices and then extracting the minimum cut from this flow. Similar ideas have been used for flow problems, but not for cut problems [2, 19].

In both the cycle- and flow-based algorithms, a partial region tree is updated with each newly found cut. The running time due to this update, in the original cycle-based algorithm, also met $\log^4 n$-factor bottleneck. We improve this by using a slightly modified version of the region-tree update step of the cycle-based algorithm. As in [4], we may assume, without loss of generality, that G is triangulated and has bounded degree.

4.1 Recursive Decomposition

A *piece* P is a subset of E that we regard as a subgraph of G that inherits its embedding from G. A *boundary vertex* of P is a vertex of P incident in G to a vertex not in P and we let δP denote the set of boundary vertices of P. A *hole* H of P is a face of P which is not a face of G. We sometimes regard H as the subgraph of G contained in H. Define the *boundary* of H as $\delta H = \delta P \cap H$.

A *decomposition* of P is a set of sub-pieces of P such that every edge of P belongs to a unique subpiece, except that edges with both endpoints in δP may belong to more than one subpiece. A *recursive decomposition* of G is obtained by first finding a decomposition of G and then recursing on each sub-piece until pieces of constant size are obtained. Ancestor/descendant relations between resulting set of pieces $\mathcal{P}$ are defined by their relations in the recursion tree. The specific type of recursive decomposition we use can be computed in $O(n \log n)$ time and has the following properties (Section 6, [4]):

- each piece of $\mathcal{P}$ is connected and has a constant number of holes and a constant number of child pieces
- $\sum_{P \in \mathcal{P}} |E(P)| = O(n \log n)$
- $\sum_{P \in \mathcal{P}} |\delta P|^2 = O(n \log n)$

To simplify the analysis, we shall further assume that each non-leaf piece has exactly two children; generalizing to a constant number of children is straightforward.

4.2 Dense dual-distance graphs

A dense distance graph is a weighted, complete graph on a subset of vertices of the original graph where the weight of an edge equals the shortest path distance in the original graph.

In the cycle-based algorithm, computations of recursive decompositions, distances, etc. stayed completely in the dual graph G^*. In our flow-based algorithm, we instead use a recursive decomposition of G, and compute flows and cuts in G; however, we rely on distances precomputed in G^*. For a piece P, rather than computing distances in G between vertices of δP, we compute distances in G^* between vertices of G^* that correspond to faces of G that are incident to δP. Let $F_I^*(P)$ (resp. $F_E^*(P)$) denote the vertices of G^* that correspond to faces of G that are incident to δP and in P (resp. not in P). Since G is triangulated and has bounded degree, $\sum_{P \in \mathcal{P}} |F_I^*(P)|^2 = O(n \log n)$ and $\sum_{P \in \mathcal{P}} |F_E^*(P)|^2 = O(n \log n)$.

The external dense dual-distance graph $\mathrm{DDG}(P)$ for a piece P is the dense distance graph for the vertex set $F_E^*(P)$ representing distances in the subgraph of G^* induced by the dual of edges that are not in P. Some external distances may not be finite since the complement of P is not necessarily connected; we can represent $\mathrm{DDG}(P)$ instead as a union of dense distance graphs, one corresponding to each component of the complement of P. The set of all external dense dual-distance graphs, $\{\mathrm{DDG}(P) : P \in \mathcal{P}\}$, can be computed in $O(n \log^3 n)$ time using minor modifications to an algorithm by Łacki, Nussbaum, Sankowski and Wulff-Nilsen [19]; see the full version for details.

4.3 Region subpieces

As in the cycle-based algorithm, our flow-based algorithm *processes* pieces of the recursive decomposition in a leaf-to-root order. Processing a piece P involves separating every pair of unseparated vertices in P. We maintain a region tree as described in the preliminaries. For a pair of vertices s and t in P that are not yet separated, there is a corresponding region R in the region tree that contains s and t. We focus our attention on a *region subpiece* which is $R_P = P \cap R$. Borradaile, Sankowski and Wulff-Nilsen [4] argue that for a leaf-most unprocessed piece P with child pieces P_1 and P_2, a region subpiece contains at most one pair of unseparated vertices, the number of region subpieces corresponding to P is $O(|\delta P_1 \cup \delta P_2|)$, and that all the region subpieces corresponding to P can be computed in time $O((|P| + |\delta P_1 \cup \delta P_2|^2) \log^2 n)$, have total size $O(|P|)$ and inherit a total of $O(|\delta P|)$ boundary vertices from P.

Given these bounds, in the sequel, we focus on a single region subpiece R_P with unseparated vertices s and t. The boundary vertices δR_P of R_P are inherited from P.

4.4 Separating s and t

Separating s and t is done by first computing a maximum st-flow in G which is explicitly represented on $E(R_P)$ and implicitly represented on $E(G) \setminus E(R_P)$. Given $\mathrm{DDG}(R_P)$, the running time of the algorithm is $O((|R_P| + |\delta R_P|^2) \log^2 n)$ which by the properties of the recursive decomposition and the bounds on the region subpieces is $O(n \log^3 n)$ over all pieces of the recursive decomposition. The algorithm is nearly identical to a part of the single-source, all-sinks maximum flows algorithm due to Łacki et al. (Section III C [19]) wherein they compute the flow between two cycles rather than two vertices; since s and t can be regarded as degenerate cycles, we can use the same algorithm. The main difference is that, in order to update the region tree, we must identify the cut edges corresponding to the maximum st-flow which is represented largely implicitly.

In order to explain how we determine the cut edges, it suffices to explain how flows are represented implicitly by Łacki et al., rather than explain their entire algorithm which we use as a black box. The flow is given by an explicit flow f_P on each edge of P and a circulation f_C defined in the entire graph. The latter is given by a potential function ϕ on the set of faces of G that is updated during the algorithm. The circulation f_C is defined by $f_C(uv) = \phi(f_2) - \phi(f_1)$, where $f_1 f_2$ is the (directed) dual edge corresponding to uv. It turns out that it suffices to maintain $\phi(f)$ for faces f incident to δR_P; this compact representation has been used in the recursive planar flow algorithms by Łacki et al. [19] and Borradaile et al. [2]. To see why, consider an edge fg of $\mathrm{DDG}(R_P)$; recall that f and g are faces incident to δR_P not in R_P. Edge fg corresponds to a shortest path Q_{fg} in $G^*[E(G) \setminus E(P)]$. The total flow crossing Q_{fg} is given by the sum of the flow $\phi(u) - \phi(v)$ on each edge uv of Q_{fg}; the sum is telescoping and the total flow crossing Q_{fg} is $\phi(g) - \phi(f)$. Q_{fg} is saturated by

the flow if $\phi(g) - \phi(f)$ is equal to the weight of fg in $\mathrm{DDG}(R_P)$. Since ϕ and $\mathrm{DDG}(R_P)$ are maintained, we can find all such *saturated* edges of $\mathrm{DDG}(R_P)$ in time $O(|\delta R_P|^2)$. Let $\mathrm{DDG}_0(R_P)$ be the subgraph of $\mathrm{DDG}(R_P)$ of saturated edges.

Consider a hole of P with boundary C; take the hole not to be the infinite face and order the vertices of C cyclically in a clockwise order. For two vertices a and b of $V(R_P) \cap C$, there is a residual a-to-b path in $G \setminus R_P$ only if there is no edge of $\mathrm{DDG}_0(R_P)$ from a face f incident to a part of C from a to b to a face g incident to a part of C from b to a; the path in G^* corresponding to fg consists of edges that are saturated from the a side on the hole to the b side of the hole.

For a vertex a of $V(R_P) \cap C$, we argue that we can determine the subset S_a of $V(R_P) \cap C$ of vertices that are reachable by a residual path in $G \setminus R_P$ in $O(|\delta R_P| \log n)$ time. Represent the out-neighbors of a face f of $\mathrm{DDG}_0(R_p)$ in clockwise order around C. By binary search, we can determine the last out-neighbor g of f in this order that is on C between f and a or determine that no such edge of $\mathrm{DDG}_0(R_P)$ exists. This restricts the vertices reachable from a by residual paths in $G \setminus R_P$ to those vertices on C between g and a. By considering each of the faces in order around C starting with the face immediately clockwise of a on C, we can, in this way, determine S_a in time $O(|\delta R_P| \log n)$. Repeating for every vertex of $V(R_P) \cap C$ and for every hole of P, we can build an external reachability graph representing reachability via residual paths in $G \setminus R_P$ among vertices of δR_P in $O(|\delta R_P|^2 \log n)$ time.

The s-side of the cut is given by those vertices reachable by paths that are residual with respect to the flow. We can find the subset of these vertices in R_P by searching alternately inside R_P via a straightforward search along residual edges (since the flow on edges of R_P are represented explicitly) and search along edges of the external reachability graph in time $O(|R_P| + |\delta R_P|^2)$. The total time to compute the flow using the algorithm of Łacki et al., build the external reachability graph and determine the vertices of R_P on the s-side of the cut is dominated by the Łacki et al. algorithm, which is $O((|R_P| + |\delta R_P|^2) \log^2 n)$.

4.5 Updating the region tree

We will describe how to update the region tree with this cut in terms of the corresponding separating cycle C. While performing this update we can, from the s-side of the cut within R_P, represent the minimum separating cycle by a subset of edges (non-residual edges at the boundary of the search) of R_P and a subset of the edges of $\mathrm{DDG}_0(R_P)$; the total size of this representation is $|C| = O(|R_P| + |\delta R_P|)$ and can be determined as part of the search. The algorithm we describe is a modified version of that described by Borradaile, Sankowski and Wulff-Nilsen [4] that achieves a logarithmic speed-up. We say that an edge $e \in E$ is a *boundary edge* of a region R if e is contained in the bounding cycle of R; Borradaile et al. showed that by maintaining the region tree as a top tree, we can determine in $O(\log n)$ time whether a given edge is a boundary edge of a given region. We use this fact in our analysis below.

We show how to update the region tree with C. C partitions the children of region R in the region tree. Let $\mathcal{C}_R$ be the child regions of R in the region tree and let $\mathcal{C}'_R$ be the subset contained in the inside of C. To correctly update the region tree, we need to remove $\mathcal{C}'_R$ as children of R, add them as children of C, and add C as a new child of R. If we can identify $\mathcal{C}'_R$, this update can be done in $O(|\mathcal{C}'_R| \log n)$ time since there are $O(|\mathcal{C}'_R|)$ topological changes and each change requires $O(\log n)$ update time in a top-tree representation of the region tree. If instead we have identified $\mathcal{C}_R - \mathcal{C}'_R$ then updating the region tree can be done in $O(|\mathcal{C}_R - \mathcal{C}'_R| \log n)$ time. We shall run two algorithms in parallel, one identifying $\mathcal{C}'_R$, the other identifying $\mathcal{C}_R - \mathcal{C}'_R$, and terminate both algorithms when the smaller set has been

identified in time $O(\min\{|\mathcal{C}'_R|, |\mathcal{C}_R - \mathcal{C}'_R|\} \log n)$. For simplicity, we assume that $\mathcal{C}'_R$ is the smaller set (the other case is symmetric) and show how to find $\mathcal{C}'_R$.

Let m'_R be the total number of edges (with multiplicity) that bound cycles of regions in $\mathcal{C}'_R$, excluding edges of C. We will show how to identify $\mathcal{C}'_R$ in time $O(m'_R \log^2 n + \log^3 n + |C|)$.

Over the course of all region tree updates, the second term sums up to $O(n \log^3 n)$ since only a linear number of cycles are added to the region tree. Likewise, since $|C| = O(|R_P| + |\delta R_P|)$ and $\sum_{P \in \mathcal{P}} \big(|R_P| + |\delta R_P|\big) = O(\sum_{P \in \mathcal{P}} |P|) = O(n \log n)$, the third term adds up to $O(n \log n)$. For the first term, consider distributing this cost among the cycles. Then note that a cycle pays for its edges no more than $O(\log n)$ times. To see this, note that each time a cycle pays, it gets a parent in the region tree with at most $c/2 + 1$ children where c denotes the number of children of the previous parent. This is a constant-factor decrease for $c > 2$. We cannot have $c < 2$ and if $c = 2$ the problem is trivial since the two cycles of $\mathcal{C}'_R$ must be faces (of constant complexity) of G^* since C separates at least one pair of unseparated faces. Hence, we get a total running time for all region tree updates of $O(n \log^3 n)$.

4.5.1 Identifying $\mathcal{C}'_R$

We identify $\mathcal{C}'_R$ in two steps. In the first step, we identify those edges that have exactly one endpoint in C and belong to cycles in $\mathcal{C}'_R$. In the second step, we explore the interior of C starting at these edges to identify the boundaries of all the cycles of $\mathcal{C}'_R$. We shall only describe the first step as the second step is done exactly as by Borradaile et al. in $O(m'_R \log n)$ time [4]. Recall that C is represented by edges of a region subpiece, which we refer to as *regular edges*, together with *super edges* in the external dense dual distance graphs, each representing a path in G^*. In the following, we assume that C consists only of super edges as the regular edges are easy to handle using the top tree representation [4].

Each super edge fg represents a shortest path in G^* and was found using a fast Dijkstra implementation of Fakcharoenphol and Rao's recursive shortest path algorithm [14] to construct a dense dual distance graph. In this construction, the path in G^* corresponding to fg has a recursive representation in line with the recursive decomposition of G. That is, fg decomposes into a path Q_{fg} of edges which themselves are super edges in (internal and external) dense dual distance graphs. The super edges of this path can be recursively decomposed until reaching edges of G^*. The number of recursion levels is at most the depth $O(\log n)$ of the recursive decomposition of G^*. We shall assume that any super edge ab (i) points to the endpoints in G^* of the subpath Q_{ab} that ab decomposes into at the next recursion level, (ii) points to the super edge of Q_{ab} that contains the midpoint of Q_{ab} (as a path in G^*), and (iii) is annotated with its length in terms of number of edges of G^* (as well as its length in terms of weights of those edges). It is easy to maintain this information during the construction of the dense dual distance graphs without an asymptotic time increase.

A minimum separating cycle in a planar graph is an isometric cycle: for any vertex r on C, C consists of two shortest paths Q_1 and Q_2 from r to vertices a and b, respectively, together with the single edge ab of G^* [17]. Borradaile, Sankowski and Wulff-Nilsen show how to find an a, b and r in time $O(\log^3 n + |C|)$ along with a representation of Q_i consisting of at most $|C|$ super edges at the top level of the recursion and $O(\log n)$ paths of super edges from dense dual distance graphs at deeper recursion levels for $i = 1, 2$ (Section 4.2 [4]). For $i = 1, 2$, recall that we need to find all edges of G^* with exactly one endpoint on Q_i that are contained in boundary cycles of $\mathcal{C}'_R$. By symmetry, we restrict our attention to Q_1.

The algorithm is as follows. First, identify the first and last edge (of G^*) on Q_1. Denote them by e_1 and e_2, respectively. Then check if e_1 and e_2 are incident to the same cycle

C' of $\mathcal{C}'_R$. As noted above, each such check can be done in $O(\log n)$ time using a top tree representation of the region tree.

If this check is positive then, since bounding cycles are isometric and shortest paths are unique, all edges of Q_1 are in $C' \cap C$ and we are done since none of these edges can be interior to a cycle of $\mathcal{C}'_R$ since these cycles have disjoint interiors. (A special case is when a cycle of $\mathcal{C}'_R$ is a face of G^* which need not be isometric but, since faces have constant size, this will not affect the analysis.)

If this check is negative, we find an edge e_m of G^* belonging to Q_1 such that a constant fraction of the (regular) edges of Q_1 are before e_m and a constant fraction of them are after e_m. We then recurse on the two subpaths of Q_1 from e_1 to e_m and from e_m to e_2, respectively. If there are k cycles of $\mathcal{C}'_R$ incident to Q_1 then the total number of regular edges e_m considered is $O(k \log n)$. This is where our improvement differs from the original region-tree-updating technique of Borradaile et al. [4]: we show how to identify each edge e_m in $O(\log n)$ time instead of $O(\log^2 n)$ time, which was the bottleneck of their method. Total time to identify all $O(k \log n)$ edges is then $O(k \log^2 n) = O(m'_R \log^2 n)$.

First we need a lemma. For a path Q, we denote by $Q[x, y]$ the subpath from x to y and we call a vertex x a *split point* of Q if at most $1/8$ of the vertices of Q are before resp. after x.

▶ **Lemma 11.** *Let $Q = a \rightsquigarrow c \rightsquigarrow e \rightsquigarrow f \rightsquigarrow d \rightsquigarrow b$ be a directed path where possibly some of the subpaths are empty. Let m (resp. $m(c, d)$) denote the midpoint of Q (resp. $Q[c, d]$), splitting the path into two (almost) equal-size subpaths such that $|Q[c, d]| > \frac{1}{4}|Q|$, $m \in Q[c, d]$, and $m(c, d) \in Q[e, f]$. We have:*

If $m \in Q[c, e]$ then e is a split point of Q; if $m \in Q[f, d]$ then f is a split point of Q; if $m \in Q[e, f]$ and $|Q[e, f]| \leq \frac{1}{4}|Q|$ then e is a split point of Q.

We use this lemma to find an edge e_m with the property described above. Recall that we store the number of edges of G^* represented by each super edge ab as well as a pointer to the sub-super edge that contains the midpoint of Q_{ab}. As described above, path Q_1 is represented by $O(|C| + \log n)$ subpaths and since we know the length of each of them, we can in $O(|C| + \log n)$ identify the subpath Q'_1 that contains the midpoint of Q_1. Q'_1 plays the role of $Q[e, f]$ and Q_1 plays the role of $Q[c, d]$ and $Q[a, b]$ in Lemma 11 (so that $a = c$ and $b = d$). The assumptions in the lemma hold and we can check in constant time whether any of the three cases apply. If so, we are done as we have found a split point and we can pick e_m as an edge incident to this point. Otherwise, we have $m \in Q[e, f]$ and $|Q'_1| = |Q[e, f]| > \frac{1}{4}|Q|$. The path $Q[e, f]$ belongs to a shortest path of a dense dual distance graph, so in constant time, we can identify the super edge uv of this path that contains its midpoint. Now apply the lemma again but with $Q[e, f]$ defined as uv, $Q[c, d]$ defined as Q'_1, and $Q[a, b]$ defined as Q_1. Then the assumptions in the lemma hold again. Applying it, we either find a split point or we decompose uv into a shortest path of super edges, and recurse on the super edge containing the midpoint of this path. Each step runs in $O(1)$ time and since there are $O(\log n)$ recursion levels, we obtain a split point and hence e_m in $O(\log n)$ time.

We conclude from the above that the total time to add bounding cycles to the region tree is $O(n \log^3 n)$.

References

1 G. Borradaile, E. Chambers, K. Fox, and A. Nayyeri. Computing minimum homology basis and minimum cycle basis in surface embedded graphs. In submission., 2015.

2 G. Borradaile, P.N. Klein, S. Mozes, Y. Nussbaum, and C. Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In *Proc. 52nd*

Symp. Found. of Computer Science (FOCS 2011), pages 170–179, 2011. doi:10.1109/FOCS.2011.73.

3 G. Borradaile, P. Sankowski, and C. Wulff-Nilsen. Min *st*-cut oracle for planar graphs with near-linear preprocessing time. In *Proc. 51st IEEE Symp. Foundations of Computer Science (FOCS 2010)*, pages 601–610, 2010. doi:10.1109/FOCS.2010.63.

4 G. Borradaile, P. Sankowski, and C. Wulff-Nilsen. Min *st*-cut oracle for planar graphs with near-linear preprocessing time. *ACM Trans. Algorithms*, 2014. To appear.

5 S. Cabello, É. Colin de Verdière, and F. Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proc. of the 26th Annual Symposium on Computational Geometry*, SoCG'10, pages 156–165, New York, NY, USA, 2010. ACM. doi:10.1145/1810959.1810988.

6 E. Chambers, J. Erickson, and A. Nayyeri. Homology flows, cohomology cuts. In *Proc. 41st ACM Symposium on Theory of Computing (STOC'09)*, pages 273–282, 2009. doi:10.1145/1536414.1536453.

7 E. Chambers, J. Erickson, and A. Nayyeri. Minimum cuts and shortest homologous cycles. In *Proc. 25th Symp. Computational Geometry (SoCG'09)*, pages 377–385, 2009. doi:10.1145/1542362.1542426.

8 Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Homology flows, cohomology cuts. *SIAM J. Comput.*, 41:1605–1634, 2009. doi:10.1137/090766863.

9 E. Demaine, G. Landau, and O. Weimann. On Cartesian trees and range minimum queries. *Algorithmica*, 68(3):610–625, 2014. doi:10.1007/s00453-012-9683-x.

10 J. Erickson. Shortest non-trivial cycles in directed surface graphs. In *Proc. of the 27th Annual Symp. on Computational Geometry*, SoCG'11, pages 236–243, New York, NY, USA, 2011. ACM. doi:10.1145/1998196.1998231.

11 J. Erickson, K. Fox, and A. Nayyeri. Global minimum cuts in surface embedded graphs. In *Proc. 23rd Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 1309–1318, 2012.

12 J. Erickson and A. Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proc. 22nd ACM-SIAM Symp. Discrete Algorithms (SODA 2011)*, pages 1166–1176, 2011.

13 J. Erickson and A. Sidiropoulos. A near-optimal approximation algorithm for asymmetric tsp on embedded graphs. In *Proc. 30th Annual Symposium on Computational Geometry*, SOCG'14, pages 130:130–130:135. ACM, 2014. doi:10.1145/2582112.2582136.

14 J. Fakcharoenphol and S. Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006. doi:10.1016/j.jcss.2005.05.007.

15 R. Gomory and T. Hu. Multi-terminal network flows. *Journal of SIAM*, 9(4):551–570, 1961. doi:10.1137/0109047.

16 D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, 1990. doi:10.1137/0219009.

17 D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM J. on Discrete Math.*, 7(3):403–418, 1994. doi:10.1137/S0895480190177042.

18 A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

19 J. Łacki, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Single source – all sinks max flows in planar digraphs. In *Proc. 53rd Symp. Found. of Computer Science (FOCS 2012)*, pages 599–608, 2012. doi:10.1109/FOCS.2012.66.

20 G. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

21 K. Mulmuley, V. Vazirani, and U. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):345–354, 1987.

Minimum Cycle and Homology Bases of Surface Embedded Graphs*

Glencora Borradaile[1], Erin Wolf Chambers[2], Kyle Fox[3], and Amir Nayyeri[4]

1 **Oregon State University, Corvallis, USA**
`glencora@eecs.orst.edu`
2 **St. Louis University, St. Louis, USA**
`echambe5@slu.edu`
3 **Duke University, Durham, USA**
`kylefox@cs.duke.edu`
4 **Oregon State University, Corvallis, USA**
`nayyeria@eecs.orst.edu`

Abstract

We study the problems of finding a minimum cycle basis (a minimum weight set of cycles that form a basis for the cycle space) and a minimum homology basis (a minimum weight set of cycles that generates the 1-dimensional ($\mathbb{Z}_2$)-homology classes) of an undirected graph embedded on an orientable surface of genus g. The problems are closely related, because the minimum cycle basis of a graph contains its minimum homology basis, and the minimum homology basis of the 1-skeleton of any graph is exactly its minimum cycle basis.

For the minimum cycle basis problem, we give a deterministic $O(n^\omega + 2^{2g}n^2)$-time algorithm. The best known existing algorithms for surface embedded graphs are those for general sparse graphs: an $O(n^\omega)$ time Monte Carlo algorithm [2] and a deterministic $O(n^3)$ time algorithm [27]. For the minimum homology basis problem, we give an $O(g^3 n \log n)$-time algorithm, improving on existing algorithms for many values of g and n.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Cycle basis, Homology basis, Topological graph theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.23

1 Introduction

Minimum cycle basis

Let $G = (V, E)$ be a connected simple undirected graph with n vertices and m edges. We define a *cycle* of G to be a subset $E' \subseteq E$ where each vertex $v \in V$ is incident to an even number of edges in E'. The *cycle space* of G is the vector space over cycles in G where addition is defined as the symmetric difference of cycles' edge sets. It is well known that the cycle space of G is isomorphic to $\mathbb{Z}_2^{m-n+1}$; in particular, the cycle space can be generated by the fundamental cycles of any spanning tree of G. A *cycle basis* is a maximal set of independent cycles. A *minimum cycle basis* is a cycle basis with a minimum number of edges

* This material is based upon work supported by the National Science Foundation under grants CCF-12-52833, CCF-10-54779, IIS-13-19573, CCF-11-61359, IIS-14-08846, CCF-15-13816, and IIS-14-47554; by an ARO grant W911NF-15-1-0408; and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation.

© Glencora Borradile, Erin Wolf Chambers, Kyle Fox, and Amir Nayyeri;
licensed under Creative Commons License CC-BY

(counted with multiplicity) or minimum total weight if edges are weighted[1]. Minimum cycle bases have applications in many areas such as electrical circuit theory [24, 9], structural engineering [8], surface reconstruction [29], and the analysis of algorithms [26].

Sets of independent cycles form a matroid, so the minimum cycle basis can be computed using the standard greedy algorithm. However, there may be an exponential number of cycles in G. Horton [21] gave the first polynomial time algorithm for the problem by observing that every cycle in the minimum cycle basis is the fundamental cycle of a shortest path tree. Several other algorithms have been proposed to compute minimum cycle bases in general graphs [10, 17, 3, 23, 27, 2]. The fastest of these algorithms are an $O(n^\omega)$ time Monte Carlo randomized algorithm of Amaldi *et al.* [2] and an $O(nm^2/\log n + n^2 m)$ time deterministic algorithm of Mehlhorn and Michail [27]. Here, $O(n^\omega)$ is the time it takes to multiply two $n \times n$ matrices using fast matrix multiplication.

For the special case of *planar graphs*, faster algorithms are known. Hartvigsen and Mardon [20] observed that the cycles in the minimum cycle basis nest, implicitly forming a tree; in fact, the edges of each cycle span an s, t-minimum cut between two vertices in the dual graph, and the Gomory-Hu tree [18] of the dual graph is precisely the tree of minimum cycle basis cycles in the primal. Hartvigsen and Mardon [20] gave an $O(n^2 \log n)$ time algorithm for the minimum cycle basis problem in planar graphs, and Amaldi *et al.* [2] improved their running time to $O(n^2)$. Borradaile, Sankowski, and Wulff-Nilsen [4] showed how to compute an *oracle* for the minimum cycle basis and dual minimum cuts in $O(n \log^4 n)$ time that is able to report individual cycles or cuts in time proportional to their size. Borradaile *et al.* [5] recently generalized the minimum cut oracle to graphs embeddable on surfaces of genus g. Their oracle takes $O(2^{O(g^2)} n \log^3 n)$ time to construct (improving upon the original planar oracle by a factor of $\log n$). Unfortunately, their oracle does not help in finding the minimum cycle basis in higher genus graphs, because there is no longer a bijection between cuts in the dual graph and cycles in the primal graph.

That said, it is not surprising that the cycle basis oracle has not been generalized beyond the plane. While cuts in the dual continue to nest in higher genus surfaces, cycles do not. In fact, the minimum cycle basis of a toroidal graph must *always* contain at least one pair of crossing cycles, because any cycle basis must contain cycles which are topologically distinct, in different *homology classes* of the surface.

Minimum homology basis

Given a graph G embedded in a surface Σ of genus g, the homology of G is an algebraic description of the topology of Σ and of G's embedding. In this paper, we focus on orientable surfaces and one-dimensional cellular homology over the coefficient ring $\mathbb{Z}_2$. Homology of this type allows for simplified definitions. We say a cycle η is *null-homologous* if η is the boundary of a subset of faces $F' \subseteq F$. Two cycles η and η' are *homologous* or in the same *homology class* if their symmetric difference $\eta \oplus \eta'$ is null-homologous. The homology classes form a vector space isomorphic to $\mathbb{Z}_2^{2g}$ known as the *homology space*. A *homology basis* of G is a set of $2g$ cycles in linearly independent homology classes, and the *minimum homology basis* of G is the homology basis with either the minimum number of edges or with minimum total weight if edges of G are weighted.

Erickson and Whittlesey [15] described an $O(n^2 \log n + gn^2 + g^3 n)$ time algorithm for computing the minimum homology basis. Like Horton [21], they apply the greedy matroid

[1] There is a notion of minimum cycle bases in directed graphs as well, but we focus on the undirected case in this paper.

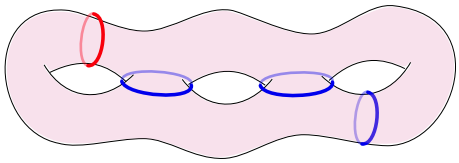


Figure 1 Two homologous cycles, one shown in red and the other in blue.

basis algorithm to a set of $O(n^2)$ candidate cycles. A set of 2^{2g} candidate cycles containing the minimum homology basis can be computed easily by applying the algorithms of Italiano *et al.* [22] or Erickson and Nayyeri [14] for computing the minimum homologous cycle in any specified homology class. These algorithms take $g^{O(g)}n \log\log n$ and $2^{O(g)}n \log n$ time respectively. Dey, Sun, and Wang [11] generalized these results to arbitrary simplicial complexes, and Busaryev *et al.* [6] improved the running time of their generalization from $O(n^4)$ to $O(n^\omega + n^2 g^{\omega-1})$. Note that all of the algorithms above either take quadratic time in n (or worse) *or* they have exponential dependency on g. In contrast, it is well understood how to find exactly one cycle of the minimum homology basis of G in only $O(g^2 n \log n)$ time, since the minimum weight non-separating cycle will always be in the basis [7, 13].

Our results

We describe new algorithms for computing the minimum cycle basis and minimum homology basis of the graph G. Our algorithm for minimum cycle basis is deterministic and runs in $O(n^\omega + 2^{2g}n^2)$ time, matching the running time of the randomized algorithm of Amaldi *et al.* [2] when g is sufficiently small. Our algorithm for minimum homology basis is also deterministic and runs in $O(g^3 n \log n)$ time assuming shortest paths are unique[2]. Ours is the first algorithm for minimum homology basis that has a running time simultaneously near-linear in n and polynomial in g.

At a high level both of our algorithms are based on the $O(nm^2 + n^2 m \log n)$ time algorithm of Kavitha *et al.* [23] who in turn use an idea of de Pina [10]. We compute our basis cycles one by one. Over the course of the algorithm, we maintain a set of *support vectors* that form the basis of the subspace that is *orthogonal* to the set of cycles we have already computed. Every time we compute a new cycle, we find the one of minimum weight that is *not* orthogonal to a chosen support vector S, and then update the remaining support vectors so they remain orthogonal to our now larger set of cycles. Using the divide-and-conquer approach of Kavitha *et al.* [23], we are able to maintain these support vectors in only $O(n^\omega)$ time total in our minimum cycle basis algorithm and $O(g^\omega)$ time total in our minimum homology basis algorithm. Our approaches for picking the minimum weight cycle not orthogonal to S form the more technically interesting parts of our algorithms and are unique to this work.

For our minimum cycle basis algorithm, we compute a collection of $O(2^{2g}n)$ cycles that contain the minimum cycle basis and then partition these cycles according to their homology classes. The cycles within a single homology class nest in a similar fashion to the minimum cycle basis cycles of a planar graph. Every time we compute a new cycle for our minimum cycle basis, we walk up the 2^{2g} trees of nested cycles and find the minimum weight cycle

[2] This assumption is only necessary to use the multiple-source shortest path data structure of Cabello, Chambers, and Erickson. It can be avoided with high probability by using randomization or deterministically by increasing the running time of our algorithm by a factor of $O(\log n)$ [7]. For simplicity, we will assume shortest paths are unique during presentation of our minimum homology basis algorithm.

not orthogonal to S in $O(n)$ time per tree. Overall, we spend $O(2^{2g}n^2)$ time finding these cycles; if any improvement is made on the time it take to update the support vectors, then the running time of our algorithm as a whole will improve as well.

Our minimum homology basis algorithm uses a covering space called the cyclic double cover. As shown by Erickson [13], the cyclic double cover provides a convenient way to find a minimum weight closed walk γ crossing an arbitrary non-separating cycle λ an odd number of times. We extend his construction so that we may consider not just one λ but any arbitrarily large collection of cycles. Every time we compute a new cycle in our minimum homology basis algorithm, we let S determine a set of cycles that must be crossed an odd number of times, build the cyclic double cover for that set, and then compute our homology basis cycle in $O(g^2 n \log n)$ time by computing minimum weight paths in the covering space[3].

The rest of the paper is organized as follows. We provide more preliminary material on surface embedded graphs in Section 2. In Section 3, we describe a characterization of cycles and homology classes using binary vectors. These vectors are helpful in formally defining our support vectors. We give a high level overview of our minimum cycle basis algorithm in Section 4 and describe how to pick individual cycles in Section 5. Finally, we give our minimum homology basis algorithm in Section 6.

2 Preliminaries

In this paper, we will be working with an edge-weighed undirected graph $G = (V, E)$ cellularly embedded on an orientable surface of genus g without boundary. To simplify the exposition, we assume G is connected and simple. We let F denote the faces of G and let f_∞ denote an arbitrary face we refer to as the *infinite face* for convenience. Let n, m, and ℓ be the number of vertices, edges, and faces of G respectively. The *Euler characteristic* χ of Σ is $2 - 2g$. By Euler's formula, $\chi = n - m + \ell$. We assume $g = O(n^{1-\varepsilon})$ for some constant $\varepsilon > 0$ (otherwise, our algorithms offer no improvement over previously known results.) This assumption implies $m = O(n)$ and $\ell = O(n)$. Embedded graphs can be *dualized*: G^* is the graph embedded on the same surface, with a vertex in G^* for every face in G and a face in G^* for every vertex of G. Two vertices in G^* are then adjacent if the corresponding faces are separated by an edge in G. We generally do not distinguish between edges in the primal and dual graphs.

A *spanning tree* of the graph G is a subset of edges of G which is a tree containing every vertex. Similarly, a *cotree* C is a spanning tree of the dual graph G^*. A *tree-cotree decomposition* of G is a partition of G into 3 edge disjoint subsets, (T, L, C), where T is a spanning tree of G, C is a cotree, and L is the set of leftover edges $E \setminus (T \cup C)$ [12]. Euler's formula implies that $|L| = 2g$.

A w, w'*-path* p is an ordered sequence of edges $\{u_1v_1, u_2v_2, \ldots, u_kv_k\}$ where $w = u_1$, $w' = v_k$, and $v_i = u_{i+1}$ for all positive $i < k$; a *closed path* is a path which starts and ends on the same vertex. A path is *simple* if it repeats no vertices (except the first and last). A path in the dual graph G^* is referred to as a *co-path* and (abusing terminology) a closed path in the dual is referred to as a *co-cycle*. We sometimes use *simple cycle* to mean

[3] In addition to the above results, we note that it is possible to improve the $g^{O(g)}n \log\log n$ time algorithm of Italiano *et al.* [22] for minimum homology basis to run in $2^{O(g)}n \log\log n$ time. However, this improvement is a trivial adaption of techniques used by Fox [16] to get a $2^{O(g)}n \log\log n$ time algorithm for minimum weight non-separating and non-contractible cycle in undirected graphs. We will not further discuss this improvement in the paper.

a simple closed path. Every member of the minimum cycle basis (and subsequently the minimum homology basis) is a simple cycle [21]. We let $\sigma(u, v)$ denote an arbitrary shortest (minimum weight) u, v-path in G. Let $p[u, v]$ denote the *subpath* of p from u to v. Given a u, v-path p and a v, w-path p', let $p \cdot p'$ denote their concatenation. Two paths p and p' *cross* if their embeddings in Σ cannot be be made disjoint through infinitesimal perturbations; more formally, they cross if there is a maximal (possibly trivial) common subpath p'' of p and p' such that, upon contracting p'' to a vertex v, two edges each of p and p' alternate in their embedded around v. Two closed paths cross if they have subpaths which cross.

Let γ be a closed path that does not cross itself. We define the operation of *cutting* along γ and denote it $G \not\leftarrow \gamma$. Graph $G \not\leftarrow \gamma$ is obtained by cutting along γ in the drawing of G on the surface, creating two copies of γ. The two copies of γ each form boundary components in the cut open surface.

Let $F' \subseteq F$ be a subset of faces and let η be the boundary of F'. We sometimes call F' a *cut* of G^* and say η *spans* the cut. A co-path p with edge $uv \in \eta$ *crosses* the cut at uv.

3 Cycle and Homology Signatures

We begin the presentation of our algorithms by giving a characterization of cycles and homology classes using binary vectors. These vectors will be useful in helping us determine which cycles can be safely added to our minimum cycle and homology bases. Let (T, L, C) be an arbitrary tree, co-tree decomposition of G; set L contains exactly $2g$ edges $e_1, \ldots, e_{2g}$. For each index $i \in \{1, \ldots, 2g\}$, let p_i denote the unique co-cycle in $C \cup \{e_i\}$. Let f_∞ be any designated face of G. Let $p_{2g+1}, \ldots, p_{m-n+1}$ be any collection of simple co-paths from f_∞ to each of the $m - n + 1 - 2g$ other faces of G.

For each edge e in G, we define its *cycle signature* $[e]$ as an $(m - n + 1)$-bit vector whose ith bit is equal to 1 if and only if e appears in p_i. The cycle signature $[\eta]$ of any cycle η is the bitwise exclusive-or of the signatures of its edges. Equivalently, the ith bit of $[\eta]$ is 1 if and only if η and p_i share an odd number of edges. Similarly, for each edge e in G, we define its *homology signature* $[e]_h$ as a $2g$-bit vector whose ith bit is equal to 1 if and only if e appears in p_i. The homology signature of cycles are defined similarly.

The following lemma is immediate.

▶ **Lemma 1.** *Let η and η' be two cycles. We have $[\eta \oplus \eta'] = [\eta] \oplus [\eta']$ and $[\eta \oplus \eta']_h = [\eta]_h \oplus [\eta']_h$.*

Cycle and homology signatures provide a convenient way to distinguish between cycles and their homology classes as shown by the following lemmas.

▶ **Lemma 2** (Erickson and Nayyeri [14, Corollary 3.3])**.** *Two cycles η and η' are homologous if and only if $[\eta]_h = [\eta']_h$.*

▶ **Lemma 3.** *Let η and η' be two homologous cycles and let $\eta \oplus \eta'$ be the boundary of a subset of faces F' such that $f_\infty \notin F'$ (set F' may be empty.) Cycle signatures $[\eta]$ and $[\eta']$ differ at bit i if and only if p_i is a co-path with exactly one endpoint in F'.*

Proof. By Lemma 2, the first $2g$ bits of $[\eta]$ and $[\eta']$ are identical. Now, consider any other bit i. The boundary of set F' is a cut in G^* that does not contain f_∞, and $\eta \oplus \eta'$ spans the cut. Suppose one endpoint of p_i lies in F'. Co-path p_i must cross the cut an odd number of times. In particular, p_i intersects exactly one of η and η' an odd number of times and the ith bits of $[\eta]$ and $[\eta']$ differ. Now, suppose instead that both endpoints of p_i lie outside F'. Here, p_i crosses the cut an even number of times. Co-path p_i either intersects both η and η' an even number of times or it intersects them both an odd number of times. Either way, the ith bits of $[\eta]$ and $[\eta']$ are equal. ◀

▶ **Lemma 4.** *Let η and η' be two cycles. We have $\eta = \eta'$ if and only if $[\eta] = [\eta']$.*

Proof. Necessity is trivial. Now, suppose $[\eta] = [\eta']$. Lemma 2 states that η and η' are $\mathbb{Z}_2$-homologous. In particular, $\eta \oplus \eta'$ forms the boundary of a subset F' of faces such that $f_\infty \notin F'$. However, subset F' must be empty by Lemma 3. We conclude $\eta = \eta'$. ◀

▶ **Corollary 5.** *Cycle signatures are an isomorphism between the cycle space and $\mathbb{Z}_2^{m-n+1}$, and homology signatures are an isomorphism between the first homology space and $\mathbb{Z}_2^{2g}$.*

4 Minimum Cycle Basis

Our algorithm for computing a minimum cycle basis is based on one of Kavitha, Mehlhorn, Michail and Paluch [23] which is in turn based on an algorithm of de Pina [10]. Our algorithm incrementally adds simple cycles $\gamma_1, \ldots, \gamma_{m-n+1}$ to the minimum cycle basis. In order to do so, it maintains a set of $(m-n+1)$-bit *support vectors* $S_1, \ldots, S_{m-n+1}$ with the following properties:

- The support vectors form a basis for $\mathbb{Z}_2^{m-n+1}$.
- When the algorithm is about to compute the jth simple cycle γ_j for the minimum cycle basis, $\langle S_j, [\gamma_{j'}] \rangle = 0$ for all $j' < j$.

Our algorithm chooses for γ_j the minimum weight cycle γ such that $\langle S_j, [\gamma] \rangle = 1$. Note that S_j must have at least one bit set to 1, because the set of vectors $S_1, \ldots, S_{m-n+1}$ forms a basis. Therefore, such a γ does exist; in particular, we could choose $[\gamma]$ to contain exactly one bit equal to 1 which matches any 1-bit of S_j. The correctness of choosing γ_j as above is guaranteed by the following lemma.

▶ **Lemma 6.** *Let S be an $(m+n-1)$-bit vector with at least one bit set to 1, and let η be the minimum weight cycle such that $\langle S, [\eta] \rangle = 1$. Then, η is a member of the minimum cycle basis.*

Proof. Let $\eta_1, \ldots, \eta_{2^{m-n+1}}$ be the collection of cycles ordered by increasing weight, and choose j such that $\eta_j = \eta$. For any subset Υ of $\{\eta_1, \ldots, \eta_{j-1}\}$, we have $\langle [\bigoplus_{\eta' \in \Upsilon} \eta'], S \rangle = 0$, where $\bigoplus$ is the symmetric difference of its operands. Therefore, η is independent of $\{\eta_1, \ldots, \eta_{j-1}\}$. Sets of independent cycles form a matroid, so η must be a member of the minimum weight cycle basis. ◀

4.1 Computing support sets

Our algorithm updates the support vectors and computes minimum cycle basis vectors in a recursive manner. Initially, each support vector S_i has only its ith bit set to 1. Borrowing nomenclature from Kavitha *et al.* [23], we define two procedures, extend(j, k) which extends the current set of basis cycles by adding k cycles starting with γ_j, and update(j, k) which updates support vectors $S_{j+\lfloor k/2 \rfloor}, \ldots, S_{j+k-1}$ so that for any j', j'' with $j + \lfloor k/2 \rfloor \leq j' < j+k$ and $1 \leq j'' < j + \lfloor k/2 \rfloor$, we have $\langle S_{j'}, [\gamma_{j''}] \rangle = 0$. Our algorithm runs extend$(1, m-n+1)$ to compute the minimum cycle basis.

We implement extend(j, k) in the following manner: If $k > 1$, then our algorithm recursively calls extend$(j, \lfloor k/2 \rfloor)$ to add $\lfloor k/2 \rfloor$ cycles to the partial minimum cycle basis. It then calls update(j, k) so that support vectors $S_{j+\lfloor k/2 \rfloor}, \ldots, S_{j+k-1}$ become orthogonal to the newly added cycles of the partial basis. Finally, it computes the remaining $\lceil k/2 \rceil$ basis cycles by calling extend$(j + \lfloor k/2 \rfloor, \lceil k/2 \rceil)$. If $k = 1$, then $\langle S_j, [\gamma_{j'}] \rangle = 0$ for all $j' < j$. Our algorithm is ready to find basis cycle γ_j. We describe an $O(2^{2g} n)$ time procedure to find γ_j in Section 5.

We now describe update(j,k) in more detail. Our algorithm updates each support vector $S_{j'}$ where $j+\lfloor k/2\rfloor \le j' < j+k$. The vector $S_{j'}$ becomes $S'_{j'} = S_{j'} + \alpha_{j'0}S_j + \alpha_{j'1}S_{j+1} + \cdots + \alpha_{j'(\lfloor k/2\rfloor-1)}S_{j+\lfloor k/2\rfloor-1}$ for some set of scalar bits $\alpha_{j'0}\ldots\alpha_{j'(\lfloor k/2\rfloor-1)}$. After updating, the set $S_1,\ldots,S_{m-n+1}$ remains a basis for $\mathbb{Z}_2^{m-n+1}$ regardless of the choices for the α bits. Further, $\langle S'_{j'}, [\gamma_{j''}]\rangle = 0$ for all $j'' < j$ for all choices of the α bits, because extend(j,k) is only called after its support vectors are updated to be orthogonal to all minimum basis cycles $\gamma_1,\ldots,\gamma_{j-1}$. However, it is non-trivial to guarantee $\langle S'_{j'}, [\gamma_{j''}]\rangle = 0$ for all j'' where $j \le j'' < j+\lfloor k/2\rfloor$. Kavitha *et al.* [23, Section 4] describe how to select the new vectors $S'_{j'}$ in $O(nk^{\omega-1})$ time using linear algebraic manipulations and fast matrix multiplication.

We can bound the running time of extend(j,k) using the following recurrence:

$$T(k) = \begin{cases} 2T(k/2) + O(nk^{\omega-1}) & \text{if } k > 1 \\ O(2^{2g}n) & \text{if } k = 1 \end{cases}$$

The total time spent in calls to extend(j,k) where $k>1$ is $O(nk^{\omega-1})$. The total time spent in calls to extend$(j,1)$ is $O(2^{2g}nk)$. Therefore, $T(k) = O(nk^{\omega-1} + 2^{2g}nk)$. The running time of our minimum cycle basis algorithm is $T(O(n)) = O(n^\omega + 2^{2g}n^2)$.

5 Selecting cycles

A *Horton cycle* is a simple cycle given by a shortest x,u-path, a shortest x,v-path, and the edge uv; in particular, the set of all Horton cycles is given by the set of $m-n+1$ elementary cycles for each of the n shortest path trees [21]. Thus, in graphs of fixed genus, there are $O(n^2)$ Horton cycles. A simple cycle γ of a graph G is *isometric* if for every pair of vertices $x,y \in \gamma$, γ contains a shortest x,y-path. Hartvigsen and Mardon prove that the cycles of any minimum cycle basis are all isometric [20]. Therefore, it suffices for us to focus on the set of isometric cycles to find the cycle γ_j as needed for Section 4.1.

Amaldi, Iuliano, Jurkiewicz, Mehlhorn, and Rizzi show how to, in a graph with n vertices and m edges, extract a set of distinct isometric cycles from a set of Horton cycles in $O(nm)$ time [2]. Each isometric cycle is identified by a shortest path tree's root and a non-tree edge.

Here, we show that there are, in fact, at most $O(2^{2g}n)$ isometric cycles in a graph of genus g (Section 5.1), and they can be partitioned into sets according to their homology classes. We can represent the isometric cycles in a given homology class using a tree that can be built in $O(n^2)$ time (Section 5.2). We then show that we can use these trees to find the minimum-cost cycle γ_j as needed for Section 4.1 in linear time per homology class of isometric cycles. We close with a discussion on how to improve the running time for computing and representing isometric cycles (Section 5.4). While these improvements do not improve the overall running time of our algorithm (by maintaining separate representations of the cycles according to their homology class, we require linear time per representation to process the support vector with respect to which γ_j is non-orthogonal; we also require $O(n^\omega)$ time to update the support vectors), it does further emphasize the bottleneck our algorithm faces in updating and representing the support vectors.

5.1 Isometric cycles in surface embedded graphs

Here we prove some additional structural properties that isometric cycles have in surface embedded graphs. To this end, we herein assume that shortest paths are unique. Hartvigsen and Mardon show how to achieve this assumption algorithmically when, in particular, all

pairs of shortest paths are computed, as we do [20]. We first prove a generalization of the following lemma for the planar case by Borradaile, Sankowski and Wulff-Nilsen.

▶ **Lemma 7** (Borradaile *et al.* [4, Lemma 1.4])**.** *Let G be a graph in which shortest paths are unique. The intersection between an isometric cycle and a shortest path in G is a (possibly empty) shortest path. The intersection between two distinct isometric cycles γ and γ' in G is a (possibly empty) shortest path; in particular, if G is a planar embedded graph, γ and γ' do not cross.*

▶ **Lemma 8.** *Two isometric cycles in a given homology class in a graph with unique shortest paths do not cross.*

Proof. Let γ and γ' be two isometric cycles in a given homology class. Suppose for a contradiction that γ and γ' cross. By the first part of Lemma 7, and the assumption that γ and γ' cross, $\gamma \cap \gamma'$ is a single simple path p. Therefore, γ and γ' cross exactly once.

Suppose γ and γ' are not null-homologous. Cutting the surface open along γ results in a connected surface with two boundary components which are connected by γ'. Cutting the surface further along γ' does not disconnect the surface. Therefore $\gamma \oplus \gamma'$ does not disconnect the surface, and so γ and γ' are not homologous, a contradiction.

If γ and γ' are null-homologous, then cutting the surface open along γ results in a disconnected surface in which $\gamma' \setminus p$ is a path, but between different components of the surface, a contradiction. ◀

▶ **Corollary 9.** *There are at most ℓ distinct isometric cycles in a given homology class in a graph with ℓ faces and unique shortest paths.*

Proof. Consider the set $\{C_1, C_2, \ldots\}$ of distinct isometric cycles in a given homology class other than the null homology class. We prove by induction that $\{C_1, C_2, \ldots, C_i\}$ cut the surface into non-trivial components, each of which is bounded by exactly two of $C_1, C_2, \ldots, C_i$; this is true for C_1, C_2 since they are homologous, distinct and do cross. C_{i+1} must be contained in one component, bounded by, say, C_j and C_k since C_{i+1} does not cross any other cycle. Cutting this component along C_{i+1} creates two components bounded by C_j, C_{i+1} and C_k, C_{i+1}, respectively. Since the cycles are distinct, these component must each contain at least one face. A similar argument holds for the set of null-homologous isometric cycles. ◀

Since there are 2^{2g} homology classes and $\ell = O(n)$, we get:

▶ **Corollary 10.** *There are $O(2^{2g}n)$ distinct isometric cycles in a graph of genus g with unique shortest paths.*

5.2 Representing isometric cycles in each homology class

We begin by determining the homology classes of each of the $O(2^{2g}n)$ isometric cycles in the following manner. Let p be a simple path, and let $[p]_h$ denote the bitwise exclusive-or of the homology signatures of its edges. Let r be the root of any shortest path tree T. Recall, $\sigma(r, v)$ denotes the shortest path between r and v. It is straightforward to compute $[\sigma(r, v)]_h$ for every vertex $v \in V$ in $O(gn)$ time by iteratively computing signatures in a leafward order. Then, the homology signature of any isometric cycle $\gamma = \sigma(r, u) \cdot uv \cdot \sigma(v, r)$ can be computed in $O(g)$ time as $[\sigma(r, u)]_h \oplus [uv]_h \oplus [\sigma(r, v)]_h$. We spend $O(2^{2g}gn) = O(2^{2g}n^2)$ time total computing homology signatures and therefore homology classes. For the remainder of this section, we consider a set of isometric cycles $\mathcal{C}$ in a single homology class.

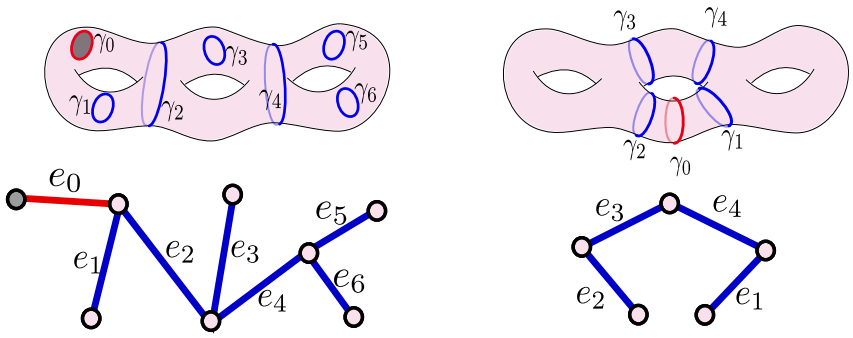


Figure 2 Two collections of homologous cycles and their generalized region trees. Left: The cycles are null-homologous. Right: The cycles lie in a non-trivial homology class.

Let $\gamma, \gamma' \in \mathcal{C}$ be two isometric cycles in the same homology class. The combination $\gamma \oplus \gamma'$ forms the boundary of a subset of faces. That is, $G \not\downarrow (\gamma \cup \gamma')$ contains at least two components. We represent the cycles in $\mathcal{C}$ by a tree $T_{\mathcal{C}}$ where each edge e of $T_{\mathcal{C}}$ corresponds to a cycle $\gamma(e) \in \mathcal{C}$ and each node v in $T_{\mathcal{C}}$ corresponds to a subset of faces $F(v)$; specifically, the nodes correspond to sets of faces in the components of $G \not\downarrow \mathcal{C}$. This tree generalizes the *region tree* defined by Borradaile, Sankowski and Wulff-Nilsen for planar graphs [4] to more general surface embedded graphs. We also designate a single representative cycle $\gamma(\mathcal{C})$ of $\mathcal{C}$ and pre-compute its cycle signature $[\gamma(\mathcal{C})]$ for use in our basis cycle finding procedure. See Figure 2.

We describe here the construction of $T_{\mathcal{C}}$. Initially, $T_{\mathcal{C}}$ is a single vertex with one (looping) edge to itself (we will guarantee $T_{\mathcal{C}}$ is a tree later). Let γ_0 be an arbitrary cycle in $\mathcal{C}$. We compute $G' = G \not\downarrow \gamma_0$. For the one vertex v of $T_{\mathcal{C}}$, we set $F(v)$ to be every face of G' and for the one edge e, we set $\gamma(e) = \gamma_0$.

We maintain the invariants that every component of G' is bound by at least two cycles of $\mathcal{C}$ (initially the cycle γ_0 is used twice), each vertex of $T_{\mathcal{C}}$ is associated with all faces in one component of G', and each edge e in $T_{\mathcal{C}}$ is associated with the cycle in $\mathcal{C}$ bounding the faces for the two vertices incident to e. Assuming these invariants are maintained, and because cycles in $\mathcal{C}$ do not cross, each cycle in $\mathcal{C}$ lies entirely within some component of G'. For each cycle $\gamma \in \mathcal{C} \setminus \{\gamma_0\}$, we set $G' := G' \not\downarrow \gamma$, subdivide the vertex associated with the faces of C's component, associate the two sets of faces created in G' with the two new vertices of $T_{\mathcal{C}}$, and associate the new edge of $T_{\mathcal{C}}$ with γ.

Let r be the vertex of $T_{\mathcal{C}}$ associated with f_∞. If cycles in $\mathcal{C}$ have trivial homology, then they each separate G, and $T_{\mathcal{C}}$ is a tree. We root $T_{\mathcal{C}}$ at r and let $\gamma(\mathcal{C})$ be an arbitrary cycle. Otherwise, let e be an arbitrary edge incident to r. We set $\gamma(\mathcal{C})$ to be $\gamma(e)$, remove e from $T_{\mathcal{C}}$, and root T at r. Observe that T has exactly one leaf other than r in this case.

Computing $G' \not\downarrow \gamma$ for one cycle γ takes $O(n)$ time. Therefore, we can compute $T_{\mathcal{C}}$ in $O(n^2)$ time total.

5.3 Selecting an isometric cycle from a homology class

Let S be an $(m - n + 1)$-bit support vector. We describe a procedure to compute $\langle S, [\gamma] \rangle$ for every isometric cycle γ in G in $O(2^{2g}n)$ time. Using this procedure, we can easily return the minimum weight cycle such that $\langle S, [\gamma] \rangle = 1$.

We begin describing the procedure for cycles in the trivial homology class. Let $\mathcal{C}$ be the collection of null-homologous isometric cycles computed above, and let $T_{\mathcal{C}}$ be the tree

computed for this set. Consider any edge e of $T_{\mathcal{C}}$. The first $2g$ bits of $[\gamma(e)]$ are equal to 0 [14, Lemma 3.2]. Cycle $\gamma(e)$ bounds a subset of faces F' such that $f_\infty \notin F'$. In particular, F' is the set of faces associated with vertices lying *below* e in $T_{\mathcal{C}}$. By Lemma 3, the ith bit of $[\gamma(e)]$ is 1 if and only if F' contains an endpoint of p_i. Therefore, the ith bit of $[\gamma(e)]$ is 1 if and only if the endpoint of p_i other than f_∞ is associated with a vertex lying below e in $T_{\mathcal{C}}$.

We compute $\langle S, [\gamma]\rangle$ for every cycle $\gamma \in \mathcal{C}$ in $O(n)$ time by essentially walking up $T_{\mathcal{C}}$ in the following manner. For each edge e in $T_{\mathcal{C}}$ going to a leaf v, we maintain a bit z initially equal to 0 and iterate over the faces in $F(v)$. Each face is the endpoint of some path p_i. If the ith bit of S is equal to 1 then we flip z. After going through all the faces, z is equal to $\langle S, [\gamma(e)]\rangle$.

We then iterate up the edges of $T_{\mathcal{C}}$ toward the root. For each edge e, we let v be the lower endpoint of e and set bit z equal to the symmetric difference over all $\langle S, \gamma(e')\rangle$ for edges e' lying below v. We then iterate over the faces of $F(v)$ as before and set $\langle S, [\gamma(e)]\rangle$ equal to z as before. We iterate over every face of G at most once during this procedure, so it takes $O(n)$ time total.

Now, consider the set of isometric cycles $\mathcal{C}$ for some non-trivial homology class. Consider any edge e of $T_{\mathcal{C}}$. Let F' be the subset of faces bound by $\gamma(\mathcal{C}) \oplus \gamma(e)$ such that $f_\infty \notin F'$. By Lemma 3, the ith bit of $[\gamma(e)]$ disagrees with the ith bit of $[\gamma(\mathcal{C})]$ if and only if path p_i has one endpoint in F'. By construction, $\gamma(\mathcal{C})$ lies on the boundary of $F(r)$ and $F(v)$ where r and v are the root and other leaf of $T_{\mathcal{C}}$ respectively. Root r is the only node of $T_{\mathcal{C}}$ associated with f_∞. We conclude the ith bit of $[\gamma(e)]$ disagrees with $[\gamma(\mathcal{C})]$ if and only if the endpoint of p_i other than f_∞ is associated with a vertex lying below e in $T_{\mathcal{C}}$.

We again walk up $T_{\mathcal{C}}$ to compute $\langle S, [\gamma]\rangle$ for every cycle $\gamma \in \mathcal{C}$. Recall, $[\gamma(\mathcal{C})]$ is precomputed and stored with $T_{\mathcal{C}}$. For each edge e of $T_{\mathcal{C}}$ in rootward order, let v be the lower endpoint of e. Let e' be the edge lying below e in $T_{\mathcal{C}}$ if it exists. If e' does not exist, we denote $\gamma(e')$ as $\gamma(\mathcal{C})$. We set z equal to $\langle S, \gamma(e')\rangle$. We then iterate over the faces of $F(v)$ as before, flipping z once for every bit i where p_i has an endpoint in $F(v)$ and bit i of S is equal to 1. We set $\langle S, \gamma(e)\rangle := z$. As before, we consider every face at most once, so the walk up $T_{\mathcal{C}}$ takes $O(n)$ time.

We have shown the following lemma, which concludes the discussion of our minimum cycle basis algorithm.

▶ **Lemma 11.** *Let G be a graph with n vertices cellulary embedded in a surface of genus g. We can preprocess G in $O(2^{2g}n^2)$ time so that for any $(m - n + 1)$-bit support vector S we can compute the minimum weight cycle γ such that $\langle S, \gamma\rangle = 1$ in $O(2^{2g}n)$ time.*

▶ **Theorem 12.** *Let G be a graph with n vertices, cellularly embedded in an orientable surface of genus g. We can compute a minimum weight cycle basis of G in $O(n^\omega + 2^{2g}n^2)$ time.*

5.4 Improving the time for computing and representing isometric cycles

Here we discuss ways in which we can improve the running time for finding and representing isometric cycles using known techniques, thereby isolating the bottleneck of the algorithm to updating the support vectors and computing γ_j.

The set and representation of isometric cycles can computed recursively using $O(\sqrt{gn})$ balanced separators (e.g. [1]) as inspired by Wulff-Nilsen [30]. Briefly, given a set S of $O(\sqrt{gn})$ separator vertices (for a graph of bounded genus), find all the isometric cycles in each component of $G \setminus S$ and represent these isometric cycles in at most 2^{2g} region trees per component, as described above. Merging the region trees for different components of $G \setminus S$ is relatively simple since different sets of faces are involved. It remains to compute the set

of isometric cycles that contain vertices of S and add them to their respective region trees. First note that a cycle that is isometric in G and does not contain a vertex of S is isometric in $G \setminus S$, but a cycle that is isometric in $G \setminus S$ may not be isometric in G, so indeed we are computing a superset of the set of isometric cycles via this recursive procedure. However, it is relatively easy to show that an isometric cycle of $G \setminus S$ can cross an isometric cycle of G at most once, so, within a given homology class, isometric cycles will nest and be, therefore, representable by a region tree.

To compute the set of isometric cycles that intersect vertices of S, we first compute shortest paths trees rooted at each of the vertices of S, generating the Horton cycles rooted at these vertices; this procedure takes $O(\sqrt{gn} \cdot n)$ time using the linear time shortest path algorithm for graphs excluding minors of sub-linear size [28]. We point out that the algorithm of Amaldi et al. [2] works by identifying Horton cycles that are not isometric and by identifying, among different Horton-cycle representations of a given isometric cycle, one representative; this can be done for a subset of Horton cycles, such as those rooted in vertices of S, and takes time proportional to the size of the representation of the Horton cycles (i.e., the $O(\sqrt{n})$ shortest path trees, or $O(\sqrt{g}n^{1.5})$).

For a given homology class of cycles, using the shortest-path tree representation of the isometric cycles, we can identify those isometric cycles in that homology class by computing the homology signature of root-to-node paths in the shortest path tree as before; this process can be done in $O(\sqrt{g}n^{1.5})$ time. We must now add these cycles to the corresponding region tree. Borradaile, Sankowski and Wulff-Nilsen [4] describe a method for adding n cycles to a region tree in $O(n \operatorname{poly} \log n)$ time that is used in their minimum cycle basis algorithm for planar graphs; this method will generalize to surfaces for nesting cycles. Therefore computing the homology classes of these isometric cycles and adding these isometric cycles to the region trees takes a total of $O(2^{2g}\sqrt{g}n^{1.5})$ time.

In total, this recursive method for computing and building a representation of a superset of the isometric cycles takes time given by the recurrence relation $T(n) = 2T(n/2) + O(2^{2g}\sqrt{g}n^{1.5})$ or $O(2^{2g}\sqrt{g}n^{1.5})$ time.

6 Homology Basis

At a high level, our algorithm for minimum homology basis is very similar to our algorithm for minimum cycle basis. As before, our algorithm incrementally adds simple cycles $\gamma_1, \ldots, \gamma_{2g}$ to the minimum homology basis by maintaining a set of $2g$ support vectors $S_1, \ldots, S_{2g}$ such that the following hold:

- The support vectors form a basis for $\mathbb{Z}_2^{2g}$.
- When the algorithm is about to compute the jth cycle γ_j for the minimum homology basis, $\langle S_j, [\gamma_{j'}]_h \rangle = 0$ for all $j' < j$.

Our algorithm chooses for γ_j the minimum weight simple cycle γ such that $\langle S_j, [\gamma]_h \rangle = 1$. The following lemma has essentially the same proof as Lemma 6.

▶ **Lemma 13.** *Let S be a $2g$-bit vector with at least one bit set to 1, and let η be the minimum weight cycle such that $\langle S, [\eta]_h \rangle = 1$. Then, η is a member of the minimum homology basis.*

As before, our algorithm updates the support vectors and computes minimum homology basis cycles in a recursive manner. We define extend(j, k) and update(j, k) as before, using homology signatures in place of cycle signatures when applicable. Our algorithm runs extend$(1, 2g)$ to compute the minimum homology basis.

The one crucial difference between our minimum cycle basis and minimum homology basis algorithms is the procedure we use to find each minimum homology basis cycle γ_j

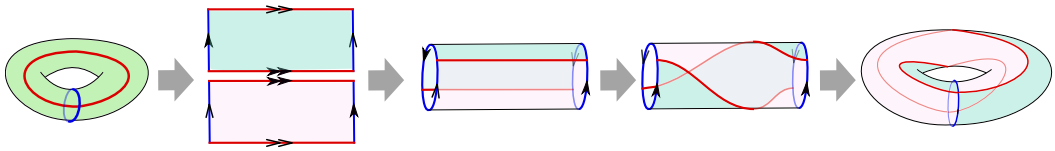


Figure 3 Constructing the cyclic double cover. Left to right: A pair of co-cycles Ψ on the torus Σ; the surfaces $(\Sigma', 0)$ and $(\Sigma', 1)$; identifying copies of one co-cycle; preparing to identify copies of the other co-cycle; the cyclic double cover.

given support vector S_j. This procedure takes $O(g^2 n \log n)$ time instead of $O(2^{2g} n)$ time and requires no preprocessing step. We describe the procedure in Sections 6.1 and 6.2.

The procedure update(j, k) takes only $O(gk^{\omega-1})$ time in our minimum homology basis algorithm, because signatures have length $O(g)$. Therefore, we can bound the running time of extend(j, k) using the following recurrence:

$$T(k) = \begin{cases} 2T(k/2) + O(gk^{\omega-1}) & \text{if } k > 1 \\ O(g^2 n \log n) & \text{if } k = 1 \end{cases}$$

The total time spent in calls to extend(j, k) where $k > 1$ is $O(gk^{\omega-1})$. The total time spent in calls to extend$(j, 1)$ is $O(g^2 kn \log n)$. Therefore, $T(k) = O(gk^{\omega-1} + g^2 kn \log n)$. The running time of our minimum homology basis algorithm[4] is $T(O(n)) = O(g^\omega + g^3 n \log n) = O(g^3 n \log n)$.

6.1 Cyclic double cover

In order to compute minimum homology basis cycle γ_j, we lift the graph into a *covering space* known as the *cyclic double cover*. Our presentation of the cyclic double cover is similar to that of Erickson [13]. Erickson describes the cyclic double cover relative to a single simple non-separating cycle; however, we describe it relative to an arbitrary *set* of non-separating *co-cycles* determined by a support vector S, similar to the homology cover construction of Erickson and Nayyeri [14].

Let S be a $2g$-bit support vector for the minimum homology basis problem as defined above. We define the cyclic double cover relative to S using a standard *voltage construction* [19, Chapter 4]. Let G_S^2 be the graph whose vertices are pairs (v, z), where v is a vertex of G and z is a bit. The edges of G_S^2 are ordered pairs $(uv, z) := (u, z)(v, z \oplus \langle S, [uv]_h \rangle)$ for all edges uv of G and bits z. Let $\pi : G_S^2 \to G$ denote the *covering map* $\pi(v, z) = v$. The *projection* of any vertex, edge, or path in G_S^2 is the natural map to G induced by π. We say a vertex, edge, or path p in G *lifts* to p' if p if the projection of p'. A closed path in G_S^2 is defined to be a face of G_S^2 if and only if its projection with regard to π is the boundary of a face of G. This construction defines an embedding of G_S^2 onto a surface Σ_S^2 (we will prove G_S^2 and Σ_S^2 are connected shortly).

We can also define G_S^2 in a more topologically intuitive way as follows. Let Ψ be a set of co-cycles which contains each co-cycle p_i for which the ith bit of S is equal to 1. Let Σ' be

[4] Our minimum homology basis algorithm can be simplified somewhat by having extend(j, k) recurse on extend$(j, 1)$ and extend$(j + 1, k - 1)$ and by using a simpler algorithm for update(j, k). This change will increase the time spent in calls to extend(j, k) where $k > 1$, but the time taken by calls with $k = 1$ will still be a bottleneck on the overall run time.

the surface obtained by cutting Σ along each cycle of Ψ. Note that Σ' may be disconnected. Each co-cycle $p_i \in \Psi$ appears as two copies on the boundary of Σ' denoted p_i^- and p_i^+ (note that p_i^- and p_i^+ may themselves be broken into multiple components if p_i crosses other co-cycles of Ψ). Create two copies of Σ' denoted $(\Sigma', 0)$ and $(\Sigma', 1)$, and let (p_i^-, z) and (p_i^+, z) denote the copies of p_i^- and p_i^+ in surface (Σ', z). For each co-cycle $p_i \in \Psi$, we identify $(p_i^+, 0)$ with $(p_i^-, 1)$ and we identify $(p_i^+, 1)$ with $(p_i^-, 0)$, creating the surface Σ_S^2 and the graph G_S^2 embedded on Σ_S^2. See Figure 3.

The first three of the following lemmas are immediate.

▶ **Lemma 14.** *Let γ be any simple cycle in G, and let s be any vertex of γ. Then γ is the projection of a unique path in G_S^2 from $(s, 0)$ to $(s, \langle S, [\gamma]_h \rangle)$.*

▶ **Lemma 15.** *Every lift of shortest path in G is a shortest path in G_S^2.*

▶ **Lemma 16.** *Let γ be the minimum weight simple cycle of G such that $\langle S, [\gamma]_h \rangle = 1$, and let s be any vertex of γ. Then γ is the projection of the shortest path in G_S^2 from $(s, 0)$ to $(s, 1)$.*

▶ **Lemma 17.** *The cyclic double cover G_S^2 is connected.*

Proof. There exists some simple cycle γ in G such that $\langle S, [\gamma]_h \rangle = 1$. Let s be any vertex of γ. Let v be any vertex of G. We show there exists a path from (v, z) to $(s, 0)$ in G_S^2 for both bits z. By Lemma 14, there is a path in G_S^2 from $(s, 0)$ to $(s, 1)$. There exists a path from v to s in G so there is a path from (v, z) to one of $(s, 0)$ or $(s, 1)$ in G_S^2. The other of $(s, 0)$ or $(s, 1)$ may be reached by following the lift of γ. ◀

Observe that G_S^2 has $2n$ vertices and $2m$ edges. Each co-cycle p_i shares an even number of edges with each face f of G. By Lemma 14, both lifts of f to G_S^2 are cycles; in particular both lifts are faces. We conclude G_S^2 contains 2ℓ faces. Surface Σ_S^2 has Euler characteristic $2n - 2m + 2\ell = 2\chi = 2 - 2(2g - 1)$. The genus of G_S^2 is exactly $2g - 1$.

6.2 Selecting homology basis cycles

Let S be any $2g$-bit support vector. We now describe our algorithm to select the minimum weight cycle γ such that $\langle S, [\gamma]_h \rangle = 1$. Our algorithm is based on one by Erickson and Nayyeri [14] for computing minimum weight cycles in arbitrary homology classes, except we use the cyclic double cover instead of their $\mathbb{Z}_2$-homology cover. We begin by computing a *greedy* tree-cotree decomposition (T^*, L^*, C^*) of G where T^* is a shortest path tree of an arbitrary vertex r and C^* is an arbitrary co-tree. Again, set L^* contains $2g$ edges $\{u_1v_1, \ldots, u_{2g}v_{2g}\}$. Let $\Lambda = \{\lambda_1, \ldots, \lambda_{2g}\}$ be the set of $2g$ loops where $\lambda_i = \sigma(r, u_1) \cdot u_1v_1 \cdot \sigma(v_1, r)$. Set Λ is called a *greedy system of loops*; cutting along each loop of Λ unfolds Σ into a topological disk [15]. Minimum homology basis cycle γ is non-separating; therefore, it crosses some loop of Λ at least once. Our algorithm constructs set Λ in $O(n \log n + gn)$ time[5]. Let G_S^2 be the cyclic double cover of G with regard to S. Our algorithm constructs G_S^2 in $O(gn)$ time.

Suppose our desired cycle γ crosses loop $\lambda_i \in \Lambda$. Simple cycle γ intersects one of two shortest paths, $\sigma(r, u_i)$ or $\sigma(v_i, r)$. Without loss of generality, it intersections $\sigma(r, u_i)$ at some vertex s. By Lemma 16, simple cycle γ is the projection of the shortest path in G_S^2 from $(s, 0)$ to $(s, 1)$. Let $\hat{\gamma}$ be this shortest path in G_S^2. Let $\hat{\sigma}$ be the lift of $\sigma(r, u_i)$ to G_S^2

[5] We only need to construct Λ once for the entire minimum homology basis algorithm, but constructing it once per basis cycle does not affect the overall run time.

that contains vertex $(s, 0)$. By Lemma 15, path $\hat{\sigma}$ is also a shortest path in G_S^2. If $\hat{\gamma}$ uses any other vertex (v, z) of $\hat{\sigma}$ other than $(s, 0)$, then it can use the entire subpath of $\hat{\sigma}$ between $(s, 0)$ and (v, z).

Now, consider the surface $\Sigma_S^2 \not\downarrow \hat{\sigma}$ which contains a single face bounded by two copies of $\hat{\sigma}$ we denote $\hat{\sigma}^-$ and $\hat{\sigma}^+$. For each vertex (v, z) on $\hat{\sigma}$, let $(v, z)^-$ and $(v, z)^+$ denote its two copies on $\hat{\sigma}^-$ and $\hat{\sigma}^+$ respectively. From the above discussion, we see $\hat{\gamma}$ is a shortest path in $\Sigma_S^2 \not\downarrow \hat{\sigma}$ from one of $(s, 0)^-$ or $(s, 0)^+$ to $(s, 1)$.

To find γ, we use the following generalization of Klein's [25] multiple-source shortest path algorithm:

▶ **Lemma 18** (Cabello *et al.* [7]). *Let G be a graph with n vertices, cellularly embedded in a surface of genus g, and let f be any face of G. We can preprocess G in $O(gn \log n)$ time and $O(n)$ space so that the length of the shortest path from any vertex incident to f to any other vertex can be retrieved in $O(\log n)$ time.*

Our algorithm iterates over the $O(g)$ shortest paths present in loops of Λ. For each such path σ, it computes a lift $\hat{\sigma}$ in G_S^2, cuts Σ_G^2 along $\hat{\sigma}$, and runs the multiple-source shortest path procedure of Lemma 18 to find the shortest path from some vertex $(s, z)^{\pm}$ on $\hat{\sigma}^{\pm}$ to $(s, z \oplus 1)$. Each shortest path it finds projects to a closed path γ' such that $\langle S, [\gamma']_h \rangle = 1$. By the above discussion, the shortest such projection can be chosen for γ. Running the multiple-source shortest path procedure $O(g)$ times on a graph of genus $O(g)$ takes $O(g^2 n \log n)$ time total. We conclude the discussion of our minimum weight homology basis algorithm.

▶ **Lemma 19.** *Let G be a graph with n vertices cellulary embedded in a surface of genus g. For any $2g$-bit support vector S we can compute the minimum weight cycle γ such that $\langle S, \gamma \rangle = 1$ in $O(g^2 n \log n)$ time.*

▶ **Theorem 20.** *Let G be a graph with n vertices, cellularly embedded in an orientable surface of genus g. We can compute a minimum weight homology basis of G in $O(g^3 n \log n)$ time.*

References

1. L. Aleksandrov and H. Djidjev. Linear algorithms for partitioning embedded graphs of bounded genus. *SIAM J. of Disc. Math.*, 9(1):129–150, 1996.
2. Edoardo Amaldi, Claudio Iuliano, Tomasz Jurkiewicz, Kurt Mehlhorn, and Romeo Rizzi. Breaking the $O(m^2 n)$ barrier for minimum cycle bases. In *Proc. 17th Ann. Euro. Symp. Algo.*, pages 301–312, 2009.
3. Franziska Berger, Peter Gritzmann, and Sven de Vries. Minimum cycle bases for network graphs. *Algorithmica*, 40(1):51–62, 2004.
4. G. Borradaile, P. Sankowski, and C. Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear preprocessing time. *ACM Trans. Algo.*, 11(3):16, 2015.
5. Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *Proc. 32nd Ann. Int. Symp. Comput. Geom.*, 2016.
6. Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K. Dey, and Yusu Wang. Annotating simplicies with a homology basis and its applications. In *Proc. 13th Scandinavian Workshop on Algo. Theory*, pages 189–200, 2012.
7. Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013.

8 A. C. Cassell, J. C. de Henderson, and K. Ramachandran. Cycle bases of minimal measure for the structural analysis of skeletal structures by the flexibility method. *Proc. R. Soc. Lond. Ser. A*, 350:61–70, 1976.

9 L. O. Chua and Li-Kuan Chen. On optimally sparse cycle and coboundary basis for a linear graph. *IEEE Trans. Circuit Theory*, 20:495–503, 1973.

10 José Coelho de Pina. *Applications of Shortest Path Methods.* PhD thesis, University of Amsterdam, 1995.

11 Tamal K. Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In *Proc. 26th Ann. Symp. Comput. Geom.*, pages 166–175, 2010.

12 David Eppstein. Dynamic generators of topologically embedded graphs. In *Proc. 14th Ann. ACM-SIAM Symp. Disc. Algo.*, pages 599–608, 2003.

13 Jeff Erickson. Shortest non-trivial cycles in directed surface graphs. In *Proc. 27th Ann. Symp. Comput. Geom.*, pages 236–243, 2011.

14 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proc. 22nd Ann. ACM-SIAM Symp. Disc. Algo.*, pages 1166–1176, 2011.

15 Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proc. 16th Ann. ACM-SIAM Symp. on Disc. Algo.*, pages 1038–1046, 2005.

16 Kyle Fox. Shortest non-trivial cycles in directed and undirected surface graphs. In *Proc. 24th Ann. ACM-SIAM Symp. Disc. Algo.*, pages 352–364, 2013.

17 Alexander Golynski and Joseph D. Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *Proc. 8th Scandinavian Workshop on Algo. Theory*, pages 200–209, 2002.

18 R. E. Gomory and T. C. Hu. Multi-terminal network flows. *J. SIAM*, 9(4):551–570, 1961.

19 Jonathan L. Gross and Thomas W. Tucker. *Topological graph theory.* Dover Publications, 2001.

20 D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM J. Disc. Math.*, 7(3):403–418, 1994.

21 J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16(2):358–366, 1987.

22 Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proc. 43rd Ann. ACM Symp. Theory Comput.*, pages 313–322, 2011.

23 Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E. Paluch. An $\tilde{O}(m^2n)$ algorithm for minimum cycle basis of graphs. *Algorithmica*, 52(3):333–349, 2008.

24 G. Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Poggendorf Ann. Physik*, 72:497–508, 1847. English transl. in Trans. Inst. Radio Engrs. CT-5 (1958), pp. 4–7.

25 Philip Klein. Multiple-source shortest paths in planar graphs. In *Proc. 16th Ann. ACM-SIAM Symp. Disc. Algo.*, pages 146–155, 2005.

26 Donald E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

27 K. Mehlhorn and D. Michail. Minimum cycle bases: Faster and simpler. *ACM Trans. Algo.*, 6(1):8, 2009.

28 S. Tazari and M. Müller-Hannemann. Shortest paths in linear time on minor-closed graph classes with an application to Steiner tree approximation. *Disc. Applied Math.*, 157(4):673–684, 2009.

29 Geetika Tewari, Craig Gotsman, and Steven J. Gortler. Meshing genus-1 point clouds using discrete one-forms. *Comput. Graph.*, 30(6):917–926, 2006.

30 C. Wulff-Nilsen. Minimum cycle basis and all-pairs min cut of a planar graph in subquadratic time. Technical Report arXiv:0912.1208, University of Copenhagen, 2009.

Finding Non-Orientable Surfaces in 3-Manifolds

Benjamin A. Burton[*1], **Arnaud de Mesmay**[†2], **and Uli Wagner**[3]

1 **School of Mathematics and Physics, The University of Queensland, Brisbane, Australia**
bab@maths.uq.edu.au
2 **CNRS, Gipsa-Lab, Grenoble, France**
arnaud.de-mesmay@gipsa-lab.fr
3 **IST Austria, Klosterneuburg, Austria**
uli@ist.ac.at

Abstract

We investigate the complexity of finding an embedded non-orientable surface of Euler genus g in a triangulated 3-manifold. This problem occurs both as a natural question in low-dimensional topology, and as a first non-trivial instance of embeddability of complexes into 3-manifolds.

We prove that the problem is NP-hard, thus adding to the relatively few hardness results that are currently known in 3-manifold topology. In addition, we show that the problem lies in NP when the Euler genus g is odd, and we give an explicit algorithm in this case.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases 3-manifold, low-dimensional topology, embedding, non-orientability, normal surfaces

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.24

1 Introduction

Since the foundational work of Haken [7] on unknot recognition, the past decades have witnessed a flurry of algorithms designed to solve decision problems in low-dimensional topology. Many of these results rely on the framework of *normal surfaces*, which provide a compact and algebraic way to analyze and enumerate the noteworthy surfaces embedded in a 3–manifold. In a nutshell, many low-dimensional problems can be seen as an instance of the following (intentionally vague) question, which encompasses the class of problems that normal surface theory has been designed to solve:

> Generic 3-manifold problem
> **Input:** A 3-manifold M.
> **Question:** Find an "interesting" surface in M.

For example, for unknot recognition [10], one triangulates the complement of the knot and looks for a spanning disk that the knot bounds, while for knot genus [1], one looks for a Seifert surface of minimal genus instead. To solve 3-sphere recognition [33, 39], one looks for

* The first author is supported by the Australian Research Council (project DP140104246).
† The second author has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no. 291734.


© Benjamin A. Burton, Arnaud de Mesmay, and Uli Wagner;
licensed under Creative Commons License CC-BY

a maximal collection of stable and unstable spheres [8]. Prime decomposition [20] and JSJ decomposition [17, 18] work by finding embedded spheres or tori in a 3-manifold – note that these decompositions are the first steps to test homeomorphism of 3-manifolds [22], which is often considered a holy grail of computational 3-manifold theory. Other examples include the computation of Heegard genus (and Heegard splittings) [24, 25], determining whether a manifold is Haken [15] or the crosscap number of a knot [5].

In this work, we investigate one of the most natural instances of this generic problem: since every 3-manifold contains every orientable surface, these (at least without further restrictions) can be considered uninteresting, and therefore the first non-trivial question is the following:

> Non Orientable Surface Embeddability
> **Input**: An integer g and a triangulation of a closed 3-manifold M.
> **Question**: Does the non-orientable surface of Euler genus g embed into M?

This question is not just a toy problem for computational 3-manifold theory: non-orientable surfaces embedded in a 3-manifold provide structural information about it. Following the foundational article of Bredon and Wood [2] classifying non-orientable surfaces in lens spaces and surfaces bundles, many works have been devoted to this study for specific 3-manifolds or specific surfaces (see for example [6, 14, 19, 23, 30, 31, 32]). Our work complements these by investigating the complexity of finding non-orientable surfaces in the most general setting.

Another motivation for studying this question comes from the higher dimensional analogues of graph embeddings. Graphs generalize naturally to simplicial complexes, and several recent efforts have been made to study higher dimensional versions of the classical notions of planar or surface-embedded graphs [26, 27, 40], see also Skopenkov [36] for some mathematical background. In particular, Matoušek, Sedgwick, Tancer and Wagner [27] recently showed that testing whether a given 2-complex embeds in $\mathbb{R}^3$ is decidable – the main algorithmic machinery underlying this result is yet another instance of the generic 3-manifold problem! In their paper, they ask what is the complexity of this problem for embeddings into other 3-manifolds (as opposed to $\mathbb{R}^3$), and since a non-orientable surface is a particular simple instance of a 2-complex, Non Orientable Surface Embeddability is the first problem to investigate in this direction.

Our results. Our first result is a proof of hardness.

▶ **Theorem 1.** *The problem* Non Orientable Surface Embeddability *is* **NP***-hard.*

As an immediate corollary, it is thus **NP**-hard to decide, given a 2-complex K and a 3-manifold M, whether K embeds into M[1]. This might not come as surprise: this is a higher-dimensional version of Graph Genus, which is already known to be **NP**-hard [38]. However, we would like to emphasize that non-orientable surfaces are among the simplest possible instances of 2-complexes, namely 2-manifolds, and by contrast deciding whether a 1-manifold, i.e., a circle graph, embeds on a surface is trivial. Furthermore, hardness results are well known to be elusive in 3-manifold topology, where iconic problems such as

[1] On the other hand this problem is not even known to be decidable. This places it in the same complexity limbo as testing embeddability of 2-complexes into $\mathbb{R}^4$ [26].

unknot recognition and 3-sphere recognition lie in **NP** ∩ co-**NP** assuming the Generalized Riemann Hypothesis [10, 21, 9, 35], and nothing is known for most other problems, the notable exception being 3-Manifold Knot Genus [1] which is known to be **NP**-complete. Our result can be seen as a hint that many three-dimensional problems are hard when the description of a 3-manifold is part of the input.

The proof of Theorem 1 starts similarly to the aforementioned one for 3-Manifold Knot Genus by Agol, Hass and Thurston: the idea is to encode an instance of One-in-Three SAT within the embeddability of a non-orientable surface inside a 2-complex. This complex is then turned into a 3-manifold by a *thickening* step and a *doubling* step. A key argument in the proof of the reduction of Agol, Hass and Thurston revolves around computing a *topological degree*, which is trivial in the case of knot genus. It turns out that this computation still works but is significantly harder in our setting, and this is the main technical hurdle in our case, for which we need to introduce (co-)homological ingredients.

Our second result provides an algorithm for this problem, provided that g is odd, proving that it is also in **NP**.

▶ **Theorem 2.** *Let g be an odd positive integer and M a triangulation of a 3-manifold. The problem* Odd Non Orientable Surface Embeddability *of testing whether M contains a non-orientable surface of Euler genus g is in* **NP**.

Observing that in the reduction involved in the proof of Theorem 1, the non-orientable surface that we use has odd Euler genus, we immediately obtain as a corollary that Odd Non Orientable Surface Embeddability is **NP**-complete.

As is the case with many problems in low-dimensional topology, proving membership in **NP** is not as trivial as most computer scientists might be accustomed to. As an illustration, our techniques fail for even values of g, and in these cases the problem is not even known to be decidable. A particularity of our proof is to leverage on the recent simplifications due to Burton [3] of the *crushing* procedure of Jaco and Rubinstein [16] to reduce the problem to the case of an irreducible 3-manifold. Then our proof relies on normal surface theory.

2 Preliminaries

We only recall here the definitions of the basic objects which we investigate in this article. The technical tools used in the proofs will be introduced when needed, and in general we will assume that the reader is familiar with the basic concepts of algebraic topology, as explained for example in Hatcher [11].

A *surface* (resp. a *surface with boundary*) is a topological space which is locally homeomorphic to the plane (resp. locally homeomorphic to the plane or the half-plane). By the theorem of classification of surfaces, these are classified up to homeomorphism by their *orientability* and their *genus* (and the number of boundaries if there are any). Since we will deal frequently with non-orientable surfaces, when we use the word genus we actually mean *Euler genus*, sometimes also called *non-orientable genus*, which equals twice the usual genus for orientable surfaces. In particular, any surface with odd genus is non-orientable.

A 3-*manifold* (resp. a *3-manifold with boundary*) is a topological space which is locally homeomorphic to $\mathbb{R}^3$, resp. to $\mathbb{R}^3$ or the half-space $\mathbb{R}^3_{|x\geq 0}$. To be consistent with the literature in low-dimensional topology, we will describe 3-manifolds not with simplicial complexes, but with the looser concept of (generalized) *triangulations*, which are defined as a collection of n abstract tetrahedra, all of whose $4n$ faces are glued together in pairs. In particular, we allow two faces of the same tetrahedron to be identified. Note that the underlying topological

space may not be a 3-manifold, but if each vertex of the tetrahedra has a neighborhood homeomorphic to $\mathbb{R}^3$ and no edge is identified to itself in the reverse direction, we obtain a 3-manifold [29].

A *simplicial complex* K is a set of simplices such that any face from a simplex in K is also in K, and the intersection of two simplices s_1 and s_2 of K is either empty or a face of both s_1 and s_2. In this article, we will only deal with 2-dimensional simplicial complexes, which are simplicial complexes where the maximal dimension of the simplices is 2 – these can be safely thought of as triangles glued together along their subfaces.

3 Hardness result

In this section we prove the following theorem.

▶ **Theorem 1.** *The problem* Non Orientable Surface Embeddability *is* **NP***-hard.*

Our reduction is inspired by the proof of Agol, Hass and Thurston [1] that Knot Genus in 3-manifolds is **NP**-hard. While the idea of the reduction is similar, the proof of its correctness is considerably more tricky. We use a reduction from the **NP**-complete [34] problem One-in-Three SAT, which we first recall. It is defined in terms of literals (boolean variables or their negations) gathered in clauses consisting of three literals.

> One-in-Three SAT
> **Input**: A set of variables U and a set of clauses over U such that each clause contains exactly 3 literals.
> **Question**: Does there exist a truth assignment for U such that each clause in C has exactly one true literal?

Starting from an instance I of One-in-Three SAT, we will build a non-orientable surface S and a 3-manifold M such that S embeds into M if and only if I is satisfiable.

3.1 The gadget

Let I be an instance of One-in-Three SAT, consisting of a set $U = \{u_1, \dots, u_n\}$ of variables and a set $C = \{c_1, \dots, c_m\}$ of clauses. The surface S is taken to be the non-orientable surface of Euler genus $2m + 2n + 1$. The construction of M is more intricate, and follows somewhat the construction of the 3-manifold of Agol, Hass and Thurston, but with a Möbius band glued on the boundary. We build M in three steps.

1. We first build a 2-dimensional complex K.
2. We *thicken* K into a 3-manifold N with boundary.
3. We *double* N, i.e., we glue two copies of N along their common boundary to obtain M.

We first describe how these spaces are defined topologically, and address in Lemma 3 the issue of computing an actual triangulation of M.

First step. The complex K is obtained in the following way. We start with a projective plane P with $n + m$ boundary curves, which we label by $u_1, \dots, u_n, c_1, \dots, c_m$. Let us denote by k_i the number of times that the variable u_i appears in the collection of clauses

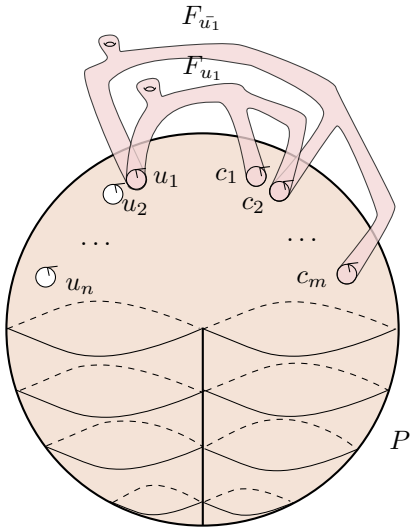


Figure 1 The projective plane P with its $n+m$ boundary curves, and examples of surfaces F_{u_1} and $F_{\bar{u_1}}$ glued to clauses containing u_1, respectively $\bar{u_1}$.

K, and $\bar{k_i}$ the number of times that the negation of u_i appears. Fix an orientation[2] of the boundary curves as in Figure 1. When gluing surfaces along curves, we will always use orientation-reversing homeomorphisms.

For $i = 1, \dots, n$, let F_{u_i} and $F_{\bar{u_i}}$ be genus one surfaces with $k_i + 1$ and $\bar{k_i} + 1$ boundaries. For each i, one boundary curve from the surface F_{u_i} is identified to u_i. The remaining k_i boundary components are identified with each of the curves c_j such that u_i appears in c_j. Similarly, $F_{\bar{u_i}}$ is attached to $\bar{u_i}$ and to every curve c_j for which $\bar{u_i}$ appears in c_j. In the end, three surfaces are attached along each u_i (F_{u_i}, $F_{\bar{u_i}}$ and P), and four surfaces are attached along each c_i (P and the surfaces corresponding to the three litterals in c_i). We call the curves $u_1, \dots, u_n, c_1, \dots c_m$ the *branching cycles* of K, and we refer to Figure 1 for an illustration.

Second step. A 3-manifold M is a *thickening* of a 2-dimensional complex K if there exists an embedding $f : K \to M$ such that M is a regular neighborhood of $f(K)$. Intuitively, a thickening corresponds to the idea of growing a 3-dimensional neighborhood around a 2-complex, but some care is needed, as not every 2-complex is thickenable – see for example Skopenkov [37] for more details on this operation.

In our case though, the complex K is always thickenable, and the process is exactly the same as in the proof of Agol, Hass and Thurston. When K is locally a surface, the thickening just amounts to taking a product with a small interval (Figure 2a.). Therefore, to define a thickening of K it suffices to describe how to thicken around its singular points, which by construction are the branching curves $u_1 \dots u_n, c_1 \dots c_m$. If $F_1, \dots F_k$ are the surfaces adjacent to a boundary curve, one can just pick a permutation of the surfaces around the curve and thicken the complex following this permutation, as in Figure 2b. This is akin to the fact that an embedding of a graph on a surface is described by a permutation of the

[2] Since P is not orientable, this is of course not well-defined. We mean an orientation "in the northern hemisphere" of P in Figure 1. Up to homeomorphism, it does not change anything, but this will be useful for the surgery arguments used throughout the proof.

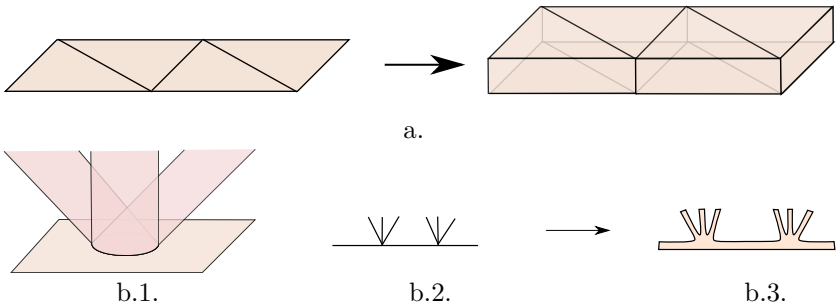


Figure 2 a. The thickening of a surface. b.1. Four surfaces adjacent to a boundary curve 2. A sectional drawing of these and 3. A sectional drawing of their thickening.

edges around each vertex. Applying this construction for every boundary curve, we obtain a 3-manifold with boundary N since every point close to the branching circles has now a neighborhood locally homeomorphic to $\mathbb{R}^3$.

Third step. In order to obtain a manifold without boundary, we *double* N, that is, we consider the disjoint union of two copies N_1 and N_2 of N, and glue them along the boundary $\partial N_1 = \partial N_2$ with the identity homeomorphism.

The following lemma shows that this construction can be computed in polynomial time. Its proof can be found in the full version of this article [4].

▶ **Lemma 3.** *A triangulation of the* 3*-manifold* M *can be computed in time polynomial in* $|I| = n + m$*, the complexity of the initial* One-in-Three SAT *instance* I*.*

Finally, let us fix some notation for the rest of the section. There is a natural projection $p : N \to K$ which corresponds to a deformation retraction of the thickening (since it is by definition a regular neighborhood). We define the continuous map $\tau : M \to N$ as being the identity on N_1 and sending every point of N_2 to its counterpart in N_1, and $\pi = \tau \circ p$.

3.2 Proof of the reduction: the easy direction

To prove Theorem 1, there remains to show how to build an embedding of S into M from a satisfying assignment for I and vice-versa. The first direction is straightforward.

▶ **Proposition 4.** *If there is a truth assignment for* I *such that each clause in* C *has exactly one true literal, then* S*, the non-orientable surface of Euler genus* $2m + 2n + 1$*, embeds in* M*.*

Proof. If there is a truth assignment for I such that each clause in C has exactly one true literal, we can embed S in K, and therefore in M, in the following way. Take the union of P and for every i, either F_{u_i} if u_i is true, or $F_{\bar{u}_i}$, if u_i is false. Then exactly two boundary components are identified along each boundary component of P, so we obtain a surface S'. Since S' contains P, it is non-orientable, and by construction S' has Euler genus $2m + 2n + 1$. Thus we have found an embedding of S. ◀

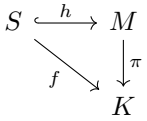
3.3 Proof of the reduction: the hard direction

The other direction will occupy us for the rest of the section.

▶ **Proposition 5.** *If* S *embeds in* M*, then there is a truth assignment for* I *such that each clause in* C *has exactly one true literal.*

Outline of the proof. Proving this proposition is the main technical step of this section, and it requires some tools from algebraic topology. Therefore, we first provide some intuition as to how the proof goes.

The natural idea would be to try to do the reverse of Proposition 4, that is, starting from an embedding of S into K, to find the truth assignment by looking at which tube the embedding chooses at every branching circle u_i. The difficulty is that we do not start with an embedding into K, but only into M. Composing this embedding h with the map $\pi = \tau \circ p : M \to K$ leads to a continuous map $f : S \to K$, but f has no reason to be an embedding.



However, this approach can be salvaged. Following Agol, Hass and Thurston [1], we can still look at the *topological degree mod 2* induced by the continuous map f at a point x in K, which roughly counts the parity of how many times f maps S to x. This number is constant where K is a surface (that is, outside of the branching circles $u_1, \dots, u_n, c_1 \dots, c_m$ of K) and the sum of the incoming degrees of the patches of K at a branching circle has to be 0: intuitively, every surface coming from one direction at a branching circle has to go somewhere. Therefore, **if the degree of f in P is 1**, exactly one of the surfaces F_{u_i} or $F_{\bar{u_i}}$ also has degree 1. We can use it to define a truth assignment for the variable in U, choosing u_i to be true if F_{u_i} has degree 1, and false in the other case. Then, the sum of the degrees also has to be 0 at the circles corresponding to the clauses. Since P has degree 1, this means that either one or three of the incoming surfaces also has degree 1. We show that this number is always one, otherwise the surface S can not have genus $2m + 2n + 1$. This will result from the fact that if we have a degree 1 map between two surfaces S_1 and S_2, then the genus of S_2 is not larger than the genus of S_1 (Lemma 8). This shows that every clause has exactly one true literal and concludes the proof.

This all hinges on the fact that the degree of f in P is one. This is where our proof diverges from the one of Agol, Hass and Thurston, as in their case this step is straightforward. Here, this will result from the non-orientability of S: morally, when embedding S into M and then mapping it into K, the only place where the non-orientability can go is P. To prove this fact formally is another matter and relies on three ingredients:

1. Since S is non-orientable and has odd genus, in the image of the embedding $h(S) \subseteq M$, there is a non-trivial homology cycle $h(\alpha)$, which has order 2 in the $\mathbb{Z}$-homology of M (Lemma 6).
2. The kernel of the map $\pi_\# : H_1(M) \to H_1(K)$ has no torsion (Lemma 7). In particular, $\pi \circ h(\alpha) = f(\alpha)$ is non-trivial in the $\mathbb{Z}$-homology of K. Intuitively, the reason is that the $\mathbb{Z}_2$-subgroup of $H_1(M)$ comes from P, which is preserved by π. To prove this formally, we split K and M at the "equator" and exploit the naturality of the Mayer-Vietoris sequence (Lemma 7).
3. Using cup-products, which provide an algebraic bridge between (co-)homology in dimensions 1 and 2, we leverage on this to prove that f has degree one on P.

Introductory lemmas. The notion of degree is conveniently expressed with the language of homology. In the following, we will rely extensively on the following notions: (relative) homology, Mayer-Vietoris sequence, cohomology, Kronecker pairing (which we denote with brackets), cup-products, and we will rely on Poincaré duality and the universal coefficient theorem. Alas, introducing (or even defining) these falls widely outside the scope of this paper, and we refer the reader to the textbook of Hatcher [11] to get acquainted with these concepts. For a map f, the induced maps in homology and cohomology are respectively denoted by $f_\#$ and $f^\#$.

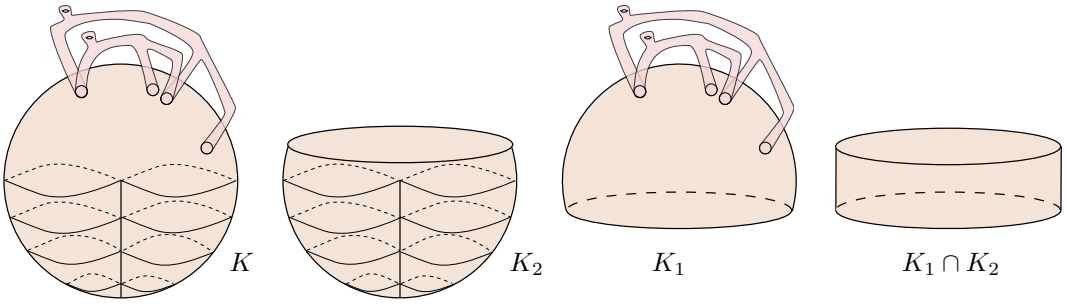


Figure 3 Decomposing the complex K around the equator.

Let us first prove the three aforementioned lemmas. The first one shows the non-triviality of maps from non-orientable surfaces of odd genus to 3-manifolds (see also Hempel [13, Lemma 5.1]). The second one shows that the map π only kills torsion-free elements and the third one shows that degree 1 maps between surfaces can only reduce the genus. To streamline the notations, when no module is indicated, homology and cohomology are taken with $\mathbb{Z}$ coefficients.

▶ **Lemma 6.** *Let S be a non-orientable surface of odd genus, and α be a simple closed curve on S, inducing an element of order 2 in $H_1(S)$. Let $f : S \to M$ be an embedding of S into a 3-manifold M. Then $f(\alpha)$ is not null-homologous in $H_1(M)$.*

Proof. We recall that a co-dimension 1 submanifold M_1 embedded in a manifold M_2 is two-sided if its normal bundle is trivial, otherwise it is one-sided. An embedded curve is orientation-preserving if it has an orientable neighborhood, otherwise it is orientation-reversing.

Since S has odd Euler characteristic, α is orientation-reversing on S. Now we distinguish two cases: either $f(S)$ is 2-sided in M, or it is 1-sided. In the first case, $f(\alpha)$ is orientation reversing in $f(S)$, and therefore also in M. Therefore it is non-trivial in $\mathbb{Z}_2$ homology. In the second case, a small generic perturbation of $f(\alpha)$ makes it have a single intersection point with $f(S)$. By Poincaré duality with $\mathbb{Z}_2$-coefficients, it is therefore non-trivial in $H_1(M, \mathbb{Z}_2)$. In both cases the result follows by the universal coefficient theorem. ◀

▶ **Lemma 7.** *Let K, M and π be as introduced in Section 3.1, then the kernel of the map $\pi_\# : H_1(M) \to H_1(K)$ has no torsion.*

Proof. Let K_1 and K_2 denote the lower and the upper hemispheres of K (see Figure 3), M_1 and M_2 be the corresponding subspaces of M. By naturality of the Mayer-Vietoris sequence with reduced homology, we obtain the following commutative diagram, where the horizontal lines are exact.

$$\begin{array}{ccccccccc}
\longrightarrow & H_1(M_1 \cap M_2) & \xrightarrow{(i_1, j_1)} & H_1(M_1) \oplus H_1(M_2) & \xrightarrow{k_1 - l_1} & H_1(M) & \xrightarrow{\partial} & \widetilde{H}_0(M_1 \cap M_2) & \longrightarrow \\
 & \downarrow{\scriptstyle \pi_\#} & & \downarrow{\scriptstyle (\pi_{1\#}, \pi_{2\#})} & & \downarrow{\scriptstyle \pi_\#} & & \downarrow{\scriptstyle \pi_\#} & \\
\longrightarrow & H_1(K_1 \cap K_2) & \xrightarrow{(i_2, j_2)} & H_1(K_1) \oplus H_1(K_2) & \xrightarrow{k_2 - l_2} & H_1(K) & \xrightarrow{\partial} & \widetilde{H}_0(K_1 \cap K_2) & \longrightarrow
\end{array}$$

We first remark that when applied to a surface, the process of thickening and doubling amounts to taking the product with S^1. Therefore, we know that K_2 is a Möbius band, M_2 is a Möbius band times a circle, $K_1 \cap K_2$ retracts to S^1 and $M_1 \cap M_2$ retracts to a torus T. Since

$M_1 \cap M_2$ and $K_1 \cap K_2$ are connected, their reduced 0-homologies are zero, thus the maps $k_1 - l_1$ and $k_2 - l_2$ are surjective. Furthermore, $\pi_{2\#}$ is the projection of the S^1 fiber, $\mathrm{Im}(i_1) = \mathbb{Z}^2$, $\mathrm{Im}(j_1) = \mathbb{Z} \oplus 2\mathbb{Z}$, $\mathrm{Im}(i_2) = \mathbb{Z}$ and $\mathrm{Im}(j_2) = 2\mathbb{Z}$. Therefore, $k_1(H_1(M_1) \oplus H_1(M_2))$ contains no torsion, and the torsion subgroup of $H_1(M)$ comes from $-l_1(H_1(M_2))$. Similarly, the torsion subgroup of $H_1(K)$ comes from $-l_2(H_1(K_2))$. By commutativity of the diagram, $(k_2 - l_2) \circ (\pi_{\#1}, \pi_{2\#}) = \pi_\# \circ (k_1 - l_1)$ and thus their image contains the $\mathbb{Z}_2$ torsion subgroup of $H_1(K)$. Therefore, the kernel of the map $\pi_\#$ has no torsion and the claim is proved. ◀

▶ **Lemma 8.** *Let $f : S_1 \to S_2$ a continuous map of degree one mod 2 between two surfaces S_1 and S_2. Then the genus of S_2 is not larger than the genus of S_1.*

The proof can be found in the full version of the paper [4].

Wrapping up the proof. We can now proceed with the proof.

Proof of Proposition 5. Let us denote by h the embedding from S into M, and by α a simple cycle of order 2 (in homology over $\mathbb{Z}$) in S. By Lemma 6, $h(\alpha)$ is not null-homologous in M, and it has order 2 in $H_1(M)$. By Lemma 7, $\pi_\# : H_1(M) \to H_1(K)$ does not have $h(\alpha)$ in its kernel, therefore we obtain that $\pi \circ h(\alpha) = f(\alpha)$ is not null-homologous in K, and since α has order 2 in $H_1(S)$, it also has order 2 in $H_1(K)$.

But there is a unique homology class of order 2 in $H_1(K)$, which is the one induced by the simple cycle β which has order 2 in P. Therefore $h(\alpha)$ is homologous to β.

We now switch to $\mathbb{Z}_2$ coefficients, in order to use the 2-dimensional homology despite the non-orientability. Since $\mathbb{Z}_2$ is a field, homology and cohomology with $\mathbb{Z}_2$ coefficients are dual to each other so we can take a cohomology class b in $H^1(K, \mathbb{Z}_2)$ which evaluates to 1 on $[\beta]$. The map $f^\# : H^1(K, \mathbb{Z}_2) \to H^1(S, \mathbb{Z}_2)$ maps b to a cohomology class $a \in H^1(S, \mathbb{Z}_2)$, and by naturality of the Kronecker pairing, we have

$$\langle a, [\alpha] \rangle = \langle f^\#(b), [\alpha] \rangle = \langle b, f_\#([\alpha]) \rangle = \langle b, [\beta] \rangle = 1,$$

where the brackets denote taking the representative in 1-dimensional homology with $\mathbb{Z}_2$ coefficients and the last equality follows from the definition of b. Now, let us denote by $(\alpha, \beta_1, \gamma_1, \beta_2, \gamma_2, \dots \beta_k, \gamma_k)$ a family of simple curves forming a basis of $H_1(S, \mathbb{Z}_2)$, such that each pair (β_i, γ_i) intersects once and there are no other intersections. We have that a is the Poincaré dual of $\alpha + \sum_I \beta_i + \sum_J \gamma_j$, for some subsets $I, J \subseteq [k]$, and since $\langle a, [\alpha] \rangle = 1$, a quick computation in the ring $H^*(S, \mathbb{Z}_2)$ shows that $a \cup a = \xi$, where ξ is the generator of $H^2(S, \mathbb{Z}_2)$.

Now, by naturality of the cup-product, we obtain $f^\#(b \cup b) = f^\#(b) \cup f^\#(b) = a \cup a = \xi$. Furthermore, once again by naturality of the Kronecker pairing, and writing $[S]$ for the fundamental class mod 2 of S, we have

$$1 = \langle \xi, [S] \rangle = \langle f^\#(b \cup b), [S] \rangle = \langle b \cup b, f_\#([S]) \rangle. \tag{1}$$

Let us open a parenthesis and recall how the notion of *degree* of a continuous map can be extended when the target is not a manifold, applied to our specific case. The map f induces a mapping $f_\#$ in *relative homology* between $H_2(S, \emptyset, \mathbb{Z}_2)$ and $H_2(K, B, \mathbb{Z}_2)$, where B is the set of branching circles of K. The group $H_2(K, B, \mathbb{Z}_2)$ is generated by the homology classes induced by the *pieces* P, F_{u_i} and $F_{\bar{u_i}}$, and therefore the image of $f_\#(S)$ associates to each piece a 0 or 1 number, the *topological degree mod 2* of f on this piece. An equivalent view of this number is the following. By standard transversality arguments, the map $f : S \to K$ can be homotoped so as to be a union of homeomorphisms of subsurfaces of S into one of

the pieces $P, F_{u_i}, F_{\bar{u_i}}$ forming K. The parity of the number of subsurfaces of S mapped to a piece P, F_{u_i} or $F_{\bar{u_i}}$ is also the topological degree mod 2 of the map f. This second point of view shows that the sum of the degrees of the pieces adjacent to a branching circle is 0, as S has no boundary.

Going back to the proof, we observe that, juggling between both interpretations of the degree, the geometric meaning of Equation (1) is that $f(S)$ covers the intersection point of two perturbed copies of β an odd number of times, and as this intersection point is in P, the topological degree mod 2 of f on P is 1.

The sum of the incoming degrees of 2-dimensional patches along a boundary curve u_i or c_i in K is 0. Therefore, around every boundary curve u_i, this allows us to pick a truth assignment for u_i, depending on whether f has degree 1 on F_{u_i} or $F_{\bar{u_i}}$. This will conclude the proof if we prove that this truth assignment φ is valid for the 1-in-3 SAT instance $|I|$.

For every clause c_i, there are exactly 4 surfaces adjacent to the boundary curve c_i, one of these being P, and we denote the others by F_1, F_2 and F_3. Since f has degree 1 on P, it has degree 1 either on one of the other surfaces or on all three. If we are in the former case for every clause, this shows that all the clauses are satisfied exactly by one of its variables under the truth assignment φ, and we are done. Otherwise, for every clause where f has degree one on all three surfaces F_1, F_2 and F_3, pick arbitrarily one, say F_1, and consider the surface S' obtained by gluing every such F_1 to P and every F_2 and F_3 together. We claim that this surface has genus strictly larger than $2n + 2m + 1$:

- The projective plane P contributes by 1.
- For every i, exactly one of the surfaces F_{u_i} or $F_{\bar{u_i}}$ is chosen. Since they have (Euler) genus two, they contribute by 2.
- For every clause, the gluing of F_1 to P increases the genus by 2. We have already reached $2n + 2m + 1$.
- Every time we glue F_2 and F_3 together, we increase the genus yet again.

But by definition of S', there is a degree one map from S to S', which is impossible by Lemma 8. This concludes the proof. ◀

The combination of Lemma 3 and Proposition 5 provides a polynomial reduction from 1-in-3 SAT to the problem of deciding the embeddability of a non-orientable surface into a 3-manifold, which concludes the proof of Theorem 1.

4 An algorithm to find non-orientable surfaces of odd Euler genus in 3-manifolds

In this section we prove the following theorem.

▶ **Theorem 2.** *Let g be an odd positive integer and M a triangulation of a 3-manifold. The problem* Odd Non Orientable Surface Embeddability *of testing whether M contains a non-orientable surface of Euler genus g is in* **NP**.

We first observe that if a non-orientable surface S of genus g embeds in a 3-manifold M, then all the non-orientable surfaces of genus $g + 2k$ for $k > 1$ also embed into M, since one can add orientable handles in a small neighborhood of S. Therefore, to prove Theorem 2 it is enough to find the non-orientable surface of **minimal** odd Euler genus which embeds into M, and this is what our algorithm will do.

Let us also note that if M is non-orientable, it contains a solid Klein bottle in the neighborhood of an orientation-reversing curve. Therefore it also contains every non-orientable

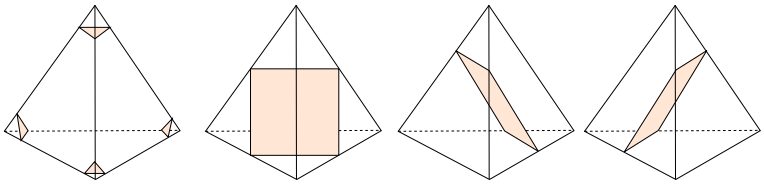


Figure 4 The seven types of normal disks within a given tetrahedron: Four triangles and three quadrilaterals.

surface of even genus, and the algorithm is trivial in this case. Thus, the only case not covered by our algorithm is the one of non-orientable surfaces of even Euler characteristic in orientable manifolds.

4.1 Background on low-dimensional topology and normal surfaces

We introduce here quickly the tools we are using from 3-dimensional topology and normal surfaces, and refer to Hass, Lagarias and Pippenger [10] or Matveev [28] for more background.

A 3-manifold M is *irreducible* if every sphere embedded in M bounds a ball in M. The *connected sum* $M_1 \# M_2$ of two 3-manifolds M_1 and M_2 is obtained by removing a small ball from both M_1 and M_2 and gluing together the resulting boundary spheres. A 3-manifold is *prime* if it can not be presented as a connected sum of more than one manifold, none of which is a sphere. It is well known [12, Proposition 1.4] that prime manifolds are irreducible, except for $S^2 \times S^1$ and the non-orientable bundle $S^2 \tilde{\times} S^1$.

Let S be a surface embedded in M. A *compressing disk* for S is an embedded disk $D \subset M$ whose interior is disjoint from S and whose boundary is a non-contractible loop in S. A surface is *compressible* if it has a compressing disk and *incompressible* if not. If a surface S is compressible, one can cut it along the boundary of a compressing disk and glue disks on the resulting boundaries, this reduces its genus by 2.

To introduce normal surfaces, we denote by T a triangulation of a 3–manifold M. A *normal isotopy* is an ambient isotopy of M that is fixed on the 2–skeleton of T. A *normal surface* in T is a properly embedded surface in T that meets each tetrahedron in a (possibly empty) disjoint collection of *normal disks*, each of which is either a *triangle* (separating one vertex of the tetrahedron from the other three) or a *quadrilateral* (separating two vertices from the other two). In each tetrahedron, there are 4 possible types of triangles and 3 possible types of quadrilaterals, pictured in Figure 4.

Normal surfaces are used to investigate combinatorially and computationally the surfaces embedded in a 3-manifold. In this endeavor, the first step is to prove that the surfaces we are interested in can be *normalized*, that is, represented by normal surfaces. The following theorem is due to Haken [7, Chapter 5], we refer to the book of Matveev for a proof.

▶ **Theorem 9** ([28, Corollary 3.3.25]). *Let M be an irreducible 3-manifold and S be an incompressible surface embedded in M. Then, if S is not a sphere, it is ambient isotopic to a normal surface.*

For S a normal surface, denote by $e(S)$ the *edge degree* of S, that is, the number of intersections of S with the 1-skeleton of the triangulation T. A normal surface is *minimal* if it has minimal edge degree over all the normal surfaces isotopic to it. Each embedded normal surface has associated *normal coordinates*: a vector in $\mathbb{Z}^{7t}_{\geq 0}$, where t is the number of tetrahedra in T, listing the number of triangles and quadrilaterals of each type in each tetrahedron. These coordinates provide an algebraic structure to normal surfaces: there

is a one-to one correspondence between normal surfaces up to normal isotopy and normal coordinates satisfying some constraints (which are called the matching equations and the quadrilateral constraints). In particular, one can add normal surfaces by adding their normal coordinates, this is called a *Haken sum*. Among normal surfaces, ones of particular interest are the *fundamental* normal surfaces, which are surfaces that can not be written as a sum of other non-empty normal surfaces. Every normal surface can be decomposed as a sum of fundamental normal surfaces, and the following theorem provides tools to understand these.

▶ **Theorem 10** ([28, Corollary 4.1.37], see also Jaco and Oertel [15]). *Let a minimal connected normal surface S in an irreducible 3-manifold M be presented as a sum $S = \sum_{i=1}^{n} S_i$ of $n > 1$ nonempty normal surfaces. If S is incompressible, so are the S_i. Moreover, no S_i is a sphere or a projective plane.*

4.2 Crushing

In order to rely on normal surface theory and apply the aforementioned theorems, we would like M to be irreducible. Therefore, the first step of the algorithm is to to simplify the 3-manifold M so as to make it irreducible. In order to do this, we rely on the operation of *crushing*, which was introduced by Jaco and Rubinstein [16], and extended to the non-orientable case (as well as simplified) by Burton [3]. In particular, Burton proves the following theorem [3, Algorithm 7].

▶ **Theorem 11.** *Given a 3-manifold M, there is an algorithm which either decomposes M into a connected sum of prime manifolds, or else proves that M contains an embedded two-sided projective plane.*

Furthermore, this algorithm is in **NP** *in the following sense: there exists a certificate of polynomial size (namely, the list of fundamental normal surfaces along which to crush) allowing to compute in polynomial time the triangulations of the summands or output that M contains an embedded two-sided projective plane.*

If this algorithm outputs an embedded projective plane, we are done, since in this case our 3-manifold M contains every non-orientable surface of odd genus. If not, if we are provided the aforementioned certificate we can proceed separately on every summand, thanks to the following easy lemma. The proof is available in the full version of this article [4].

▶ **Lemma 12.** *Let M be a connected sum of 3-manifolds $M_1, \dots M_k$. Then if a non-orientable surface S of odd genus g embeds into M, it also embeds into one of the M_i.*

If one of the summands is prime but not irreducible, then, as mentioned before, it is homeomorphic either to $S^2 \times S^1$ or the twisted bundle $S^2\tilde{\times}S^1$. One of the features [3, Algorithm 7] of the crushing algorithm that we use is that the $S^2 \times S^1$ and $S^2\tilde{\times}S^1$ summands in the prime decomposition are actually rebuilt afterwards based on the homology of the input 3-manifold. In particular, we know precisely if there are any and how many of them there are, without having to use some hypothetical recognition algorithm. Furthermore, the following lemma shows that these summands are uninteresting for our purpose.

▶ **Lemma 13.** *No non-orientable surface of odd genus embeds into $S^2 \times S^1$ or $S^2\tilde{\times}S^1$.*

Proof. By Lemma 6, if such an embedding existed, there would be an element of order 2 in $H_1(S^2 \times S^1)$ or $H_1(S^2\tilde{\times}S^1)$, which is a contradiction since both of these groups are equal to $\mathbb{Z}$. ◀

Therefore, the output of our algorithm is trivial for these summands, and in the rest of this section we assume that the manifold M is irreducible.

4.3 Fundamental normal surfaces

We now show that in order to find the non-orientable surface of minimal odd genus, it is enough to look at the fundamental normal surfaces.

▶ **Proposition 14.** *If a non-orientable surface of minimal odd genus embeds in an irreducible 3-manifold M, then it is witnessed by one of the fundamental normal surfaces. If none of the fundamental normal surfaces have odd genus, then no surface of odd genus embeds into M.*

Before proving this proposition, let us show how it implies Theorem 2.

Proof of Theorem 2. By applying the crushing procedure and following Lemma 12 and the discussion in Section 4.2, one can assume that M is irreducible if one is given the certificate of Theorem 11. Then, by Proposition 14, the non-orientable surface of minimal odd genus, if it exists, appears among one of the fundamental normal surfaces. By a now standard argument of Hass, Lagarias and Pippenger [10, Lemma 6.1], the coordinates of fundamental normal surfaces can be described with a polynomial number of bits. Since there are $7t$ coordinates for a triangulation of size T, we can therefore use this as a second half of the **NP** certificate. Now, if the input genus g is at least the minimal one witnessed by this certificate, then the non-orientable surface of genus g is embeddable in M, otherwise it is not. ◀

We now prove Proposition 14.

Proof of Proposition 14. Let S be a surface of minimal odd genus g embedded in M. We first claim that S is incompressible. Indeed, if it is not, let D be a compressing disk and $S \mid D$ be the surface obtained after the compression along D: then $S \mid D$ has genus $g - 2$ which contradicts the minimality of g.

The surface S being incompressible, then by Theorem 9, there exists a normal surface isotopic to it. Let us denote by S' a normal surface of genus g and of minimal edge degree among all of those. If S' is not fundamental, by Theorem 10, then it can be written as a sum of fundamental normal surfaces $S' = \sum_{i=1}^{n} S_i$ such that the S_i are incompressible and none of them are spheres or projective planes. In particular, none of the surfaces S_i have positive Euler characteristic. Since the Euler characteristic is additive on the space of normal coordinates, one of the surfaces S_i has odd genus at most g. By minimality of g, this surface S_i actually has genus g, and it has smaller edge degree by S', which is a contradiction. Therefore S' is fundamental, which concludes the proof. ◀

▶ Remark. The reason why the above proof fails in the case of even genus is that in general a non-orientable surface of genus g might be written as a Haken sum of orientable surfaces. In our case, this issue is avoided by the fact that a surface of odd Euler genus is necessarily non-orientable. For even Euler genus, the first problem that we do not solve is the one of deciding whether a given 3-manifold contains a Klein bottle. For this specific case, we believe that the problem should be decidable, by computing a JSJ decomposition and identifying in the geometric pieces which ones contain Klein bottles: hyperbolic pieces do not, and one can detect which Seifert fibered spaces do just based on their invariants. However, this technique does not seem to apply to higher genera.

Acknowledgements. We would like to thank Saul Schleimer and Eric Sedgwick for stimulating discussions, and the anonymous reviewers for helpful comments.

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.

2 Glen E. Bredon and John W. Wood. Non-orientable surfaces in orientable 3-manifolds. *Invent. Math.*, 7:83–110, 1969.

3 Benjamin A. Burton. A new approach to crushing 3-manifold triangulations. *Discrete & Computational Geometry*, 52(1):116–139, 2014.

4 Benjamin A. Burton, Arnaud de Mesmay, and Uli Wagner. Finding non-orientable surfaces in 3-manifolds. arXiv:1602.07907, 2016.

5 Benjamin A. Burton and Melih Ozlen. Computing the crosscap number of a knot using integer programming and normal surfaces. *ACM Trans. Math. Softw.*, 39(1):4:1–4:18, November 2012.

6 Werner End. Non-orientable surfaces in 3-manifolds. *Archiv der Mathematik*, 59(2):173–185, 1992.

7 Wolfgang Haken. Theorie der Normalflachen, ein Isotopiekriterium für den Kreisnoten. *Acta Mathematica*, 105:245–375, 1961.

8 Joel Hass. What is an almost normal surface. arXiv:1208.0568v1, 2012.

9 Joel Hass and Greg Kuperberg. New results on the complexity of recognizing the 3-sphere. In *Oberwolfach Reports*, volume 9, pages 1425–1426, 2012.

10 Joel Hass, Jeffrey C. Lagarias, and Nicholas Pippenger. The computational complexity of knot and link problems. *Journal of the ACM*, 46(2):185–211, 1999.

11 Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002. Available at `http://www.math.cornell.edu/~hatcher/`.

12 Allen Hatcher. Notes on basic 3-manifold topology. Notes available on the author's webpage, 2007.

13 John Hempel. *3-manifolds*. AMS Chelsea Publishing, Providence, RI, 2004. Reprint of the 1976 original.

14 Miwa Iwakura and Chuichiro Hayashi. Non-orientable fundamental surfaces in lens spaces. *Topology and its Applications*, 156(10):1753–1766, 2009.

15 William Jaco and Ulrich Oertel. An algorithm to decide if a 3-manifold is a Haken manifold. *Topology*, 23(2):195–209, 1984.

16 William Jaco and J. Hyam Rubinstein. 0-efficient triangulations of 3-manifolds. *Journal of Differential Geometry*, 65:61–168, 2003.

17 William H Jaco and Peter B Shalen. *Seifert fibered spaces in 3-manifolds*, volume 220. American Mathematical Society, 1979.

18 K Johannson. Homotopy equivalence of 3-manifolds with boundary. *Lecture Notes in Math*, 761, 1979.

19 Paik Kee Kim. Some 3-manifolds which admit Klein bottles. *Transactions of the American Mathematical Society*, 244:299–312, 1978.

20 Hellmuth Kneser. Geschlossene Flächen in dreidimensionalen Mannigfaltigkeiten. *Jahresbericht Math. Verein.*, 28:248–260, 1929.

21 Greg Kuperberg. Knottedness is in NP, modulo GRH. *Advances in Mathematics*, 256:493–506, 2014.

22 Greg Kuperberg. Algorithmic homeomorphism of 3-manifolds as a corollary of geometrization. arXiv:1508.06720, 2015.

23 Adam Levine, Daniel Ruberman, and Sašo Strle. Nonorientable surfaces in homology cobordisms. *Geometry & Topology*, 19(1):439–494, 2015.

24 Tao Li. Heegaard surfaces and measured laminations, I: the Waldhausen conjecture. *Inventiones mathematicae*, 167(1):135–177, 2007.

25 Tao Li. An algorithm to determine the Heegaard genus of a 3–manifold. *Geometry & Topology*, 15(2):1029–1106, 2011.

26 Jiří Matoussšek, Martin Tancer, and Uli Wagner. Hardness of embedding simplicial complexes in R^d. *Journal of the European Mathematical Society*, 13(2):259–295, 2011.

27 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Embeddability in the 3-sphere is decidable. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 78:78–78:84, New York, NY, USA, 2014. ACM.

28 Sergei V. Matveev. *Algorithmic topology and classification of 3-manifolds*, volume 9 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003.

29 Edwin E. Moise. Affine structures in 3-manifolds. V. The triangulation theorem and Hauptvermutung. *Ann. of Math. (2)*, 56:96–114, 1952.

30 Richard Rannard. Incompressible surfaces in Seifert fibered spaces. *Topology and its Applications*, 72(1):19–30, 1996.

31 Joachim Hyam Rubinstein. On 3-manifolds that have finite fundamental group and contain Klein bottles. *Transactions of the American Mathematical Society*, 251:129–137, 1979.

32 Joachim Hyam Rubinstein. Nonorientable surfaces in some non-Haken 3-manifolds. *Transactions of the American Mathematical Society*, 270(2):503–524, 1982.

33 Joachim Hyam Rubinstein. An algorithm to recognize the 3-sphere. In *Proceedings of the International Congress of Mathematicians, Vol. 1, 2 (Zürich, 1994)*, pages 601–611, Basel, 1995. Birkhäuser.

34 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

35 Saul Schleimer. Sphere recognition lies in NP. In Michael Usher, editor, *Low-dimensional and Symplectic Topology*, volume 82, pages 183–214. American Mathematical Society, 2011.

36 A. B. Skopenkov. Embedding and knotting of manifolds in euclidean spaces. In Nicholas Young and Yemon Choi, editors, *Surveys in Contemporary Mathematics*, pages 248–342. Cambridge University Press, 2007. Cambridge Books Online.

37 A.B. Skopenkov. A generalization of Neuwirth's theorem on thickening 2-dimensional polyhedra. *Mathematical Notes*, 58(5):1244–1247, 1995.

38 Carsten Thomassen. The graph genus problem is NP-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

39 Abigail Thompson. Thin position and the recognition problem for S^3. *Mathematical Research Letters*, 1:613–630, 1994.

40 Uli Wagner. Minors in random and expanding hypergraphs. In *Proceedings of the Twenty-Seventh Annual Sympoisum on Computational Geometry*, pages 351–360, 2011.

Structure and Stability of the 1-Dimensional Mapper*

Mathieu Carrière[1] and Steve Oudot[2]

1 **DataShape, Inria Saclay, Palaiseau, France**
`mathieu.carriere@inria.fr`
2 **DataShape, Inria Saclay, Palaiseau, France**
`steve.oudot@inria.fr`

Abstract

Given a continuous function $f : X \to \mathbb{R}$ and a cover $\mathcal{I}$ of its image by intervals, the Mapper is the nerve of a refinement of the pullback cover $f^{-1}(\mathcal{I})$. Despite its success in applications, little is known about the structure and stability of this construction from a theoretical point of view. As a pixelized version of the Reeb graph of f, it is expected to capture a subset of its features (branches, holes), depending on how the interval cover is positioned with respect to the critical values of the function. Its stability should also depend on this positioning. We propose a theoretical framework relating the structure of the Mapper to that of the Reeb graph, making it possible to predict which features will be present and which will be absent in the Mapper given the function and the cover, and for each feature, to quantify its degree of (in-)stability. Using this framework, we can derive guarantees on the structure of the Mapper, on its stability, and on its convergence to the Reeb graph as the granularity of the cover $\mathcal{I}$ goes to zero.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Mapper, Reeb Graph, Extended Persistence, Topological Data Analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.25

1 Introduction

The *Mapper*[1] was introduced in [22] as a new mathematical object to summarize the topological structure of a continuous map $f : X \to \mathbb{R}^d$. Its construction depends on the choice of a cover $\mathcal{I}$ of the image of f by open sets. Pulling back $\mathcal{I}$ through f^{-1} gives an open cover $\mathcal{U}$ of X. Splitting each element of $\mathcal{U}$ into its various connected components yields a connected cover $\mathcal{V}$, whose nerve is the Mapper (which thus has one k-simplex per non-empty $(k+1)$-fold intersection of elements of $\mathcal{V}$). The Mapper can be thought of as a *pixelized version* of the Reeb space, where the resolution is prescribed by the cover $\mathcal{I}$. In practice, its construction from point cloud data is easy to describe and implement, requiring only to build a neighborhood graph whose size is at worst quadratic in the size of the point cloud.

Since its introduction, the Mapper has aroused the interest of practitioners in the data sciences, with several success stories [1, 21], due to its ability to deal with very general functions and datasets. Nevertheless, little is known to date about the structure of the Mapper and its stability with respect to perturbations of the pair (X, f) or of the cover $\mathcal{I}$. Intuitively, when f is scalar, the Mapper is a pixelized version of the Reeb graph, so it should

* Note: The proofs are omitted in this extended abstract and can be found in the full version [9].
[1] In this article we call *Mapper* the mathematical object, not the algorithm used to build it.


© Mathieu Carrière and Steve Oudot;
licensed under Creative Commons License CC-BY

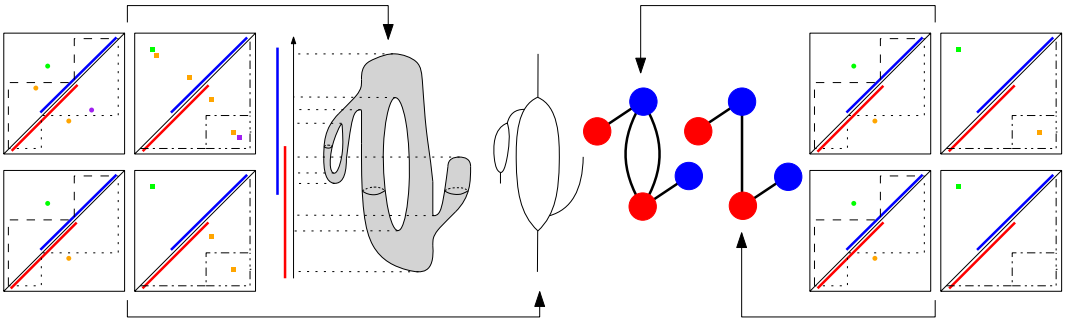


Figure 1 From left to right: a 2-manifold equipped with the height function; the corresponding Reeb graph, MultiNerve Mapper, and Mapper. For each object, we display the persistence diagrams of dimension 0 (green), 1 (orange) and 2 (purple). Extended points are squares while ordinary and relative points are disks (above and below the diagonal respectively). The staircases are represented with dashed ($Q^{\mathcal{I}}_{O}$), dotted ($Q^{\mathcal{I}}_{E-}$) dash-dotted ($Q^{\mathcal{I}}_{R}$) and dash-dot-dotted ($Q^{\mathcal{I}}_{E}$) lines.

capture some of its features (branches, holes) and miss others, depending on how the cover $\mathcal{I}$ is positioned with respect to the critical values of f. How can we formalize this phenomenon and quantify the stability of the structure of the Mapper when f is scalar? These are the questions addressed here.

Contributions. Assuming f is scalar, we draw an explicit connection between the Mapper and the Reeb graph, from which we derive guarantees on the structure of the Mapper and quantities to measure its stability. The connection happens through an intermediate object, called the *MultiNerve Mapper*, which we define as the *multinerve* [16] of the connected pullback cover.

Given a pair (X, f) with $f : X \to \mathbb{R}$ continuous, and an interval cover $\mathcal{I}$ of $\mathrm{im}(f)$, we show that the MultiNerve Mapper itself is a Reeb graph, for a perturbed pair (X', f') (Theorem 5.3). Furthermore, we are able to track the changes that occur in the structure of the Reeb graph as we go from the initial pair (X, f) to its perturbed version (X', f'). More precisely, we can match the quotient maps' persistence diagrams $\mathrm{Dg}(\tilde{f})$ and $\mathrm{Dg}(\tilde{f}')$ with each other (Theorem 5.2), and thus draw a correspondence between the features of the MultiNerve Mapper and the ones of the Reeb graph of (X, f). This correspondence is oblivious to the actual layouts of the features in the two graphs, which in principle could differ.

The previous connection allows us to derive a signature for the (MultiNerve) Mapper, which takes the form of a persistence diagram. The points in this diagram are in one-to-one correspondence with the features (branches, holes) in the (MultiNerve) Mapper. Thus, like $\mathrm{Dg}(\tilde{f})$, our diagram for the (MultiNerve) Mapper serves as a bag-of-features type descriptor.

An interesting property of our descriptor is to be predictable[2] given the persistence diagram of the quotient map $\tilde{f}$. Indeed, it is obtained from this diagram by removing the points lying in certain *staircases* that are defined solely from the cover $\mathcal{I}$ and that encode the mutual positioning of the intervals of the cover. Thus, the descriptor for the (MultiNerve) Mapper is a subset of the one for the Reeb graph, which provides theoretical evidence to the intuitive claim that the Mapper is a pixelized version of the Reeb graph. Then, one can easily derive sufficient conditions under which the bag-of-features structure of the Reeb graph is preserved in the (MultiNerve) Mapper, and when it is not, one can easily predict which features are preserved and which ones disappear (Corollary 5.4). See Figure 1.

[2] As a byproduct, we also clarify the relationship between the diagram of $\tilde{f}$ and the one of f (Theorem 2.5).

The staircases also play a role in the stability of the (MultiNerve) Mapper, since they prescribe which features will (dis-)appear as the function f is perturbed. Stability is then naturally measured by a slightly modified version of the bottleneck distance, in which the staircases play the role of the diagonal. Our stability guarantees (Theorem 6.1) follow easily from the general stability theorem for extended persistence [15]. Similar guarantees hold when the domain X or the cover $\mathcal{I}$ is perturbed. These stability guarantees can be exploited in practice to approximate the descriptors of the Mapper and MultiNerve Mapper from point cloud data efficiently. The details are given in Section 7 of the full version of the paper [9].

Our main proof technique consists in perturbing the so-called *telescope* [7] corresponding to the pair (X, f). We introduce a set of elementary perturbations and study their effects on the persistence diagram. By performing these perturbations in sequence, we can track the points in the diagram while the pair (X, f) is being modified. We believe these elementary perturbations are of an independent interest (see Section 4).

Related work. Reeb graphs are now well understood and have been used in a wide range of applications. We refer the interested reader to [4, 5, 6] for a comprehensive list of references. In a recent study, even more structure has been given to the Reeb graphs by categorifying them [17].

Several variants of these graphs have been studied in the last decade to face the common issues that come with the Reeb graphs (complexity and computational cost among others). The Mapper [22] is one of them. Chazal et al. [14] introduced the λ-*Reeb graph*, which is another type of Reeb graph pixelization with intervals. The authors can derive upper bounds on the Gromov-Hausdorff distance between the space and its Reeb or λ-Reeb graph. This is too much asking in general; as a result, the hypothesis are very strong.

Joint Contour Nets [8, 11] and Extended Reeb graphs [3] are Mapper-like objects. The former is the Mapper computed with the cover of the codomain given by rounding the function values, while the latter is the Mapper computed from a partition of the domain with no overlap. Munch and Wang [20] recently showed that, as the lengths of the intervals in the cover $\mathcal{I}$ go to zero uniformly, the Joint Contour Net and the Mapper itself converge to the continuous Reeb space in the so-called *interleaving distance* [17]. Their result holds in the general case of vector-valued functions. Here we restrict the focus to real-valued functions but are able to make non-asymptotic claims (Corollary 5.4).

On another front, Stovner [23] proposed a categorified version of the Mapper, seen as a covariant functor from the covered topological spaces to the simplicial complexes. Dey et al. [18] pointed out the inherent instability of the Mapper and proposed a multiscale variant that is built by taking the Mapper over a hierarchy of covers of the codomain. They derived a stable signature by considering the persistence diagram of this family. Unfortunately, their construction is hard to relate to the original Mapper. Babu [2] characterized the Mapper with zigzag persistent homology. Here, we do not coarsen a zigzag module but rather identify specific areas of an extended persistence diagram corresponding to features that disappear in the Mapper. By doing so, we answer two open questions from [18], introducing a signature that describes the set of features of the Mapper completely, together with a quantification of their stability and a provable way of approximating them from point cloud data.

2 Background

Throughout the paper we work with singular homology with coefficients in the field $\mathbb{Z}_2$, which we omit in our notations for simplicity. In the following, "connected" stands for "path-

connected", and "cc" stands for "connected component(s)". Given a real-valued function f on a topological space X, and an interval $I \subseteq \mathbb{R}$, we denote by X_f^I the preimage $f^{-1}(I)$. We omit the subscript f in the notation when there is no ambiguity in the function considered.

2.1 Morse-Type Functions

▶ **Definition 2.1.** A continuous real-valued function f on a topological space X is *of Morse type* if:

- **(i)** There is a finite set $\mathrm{Crit}(f) = \{a_1 < ... < a_n\} \subset \mathbb{R}$, called the set of *critical values*, s.t. over every open interval $(a_0 = -\infty, a_1), ..., (a_i, a_{i+1}), ..., (a_n, a_{n+1} = +\infty)$ there is a compact and locally connected space Y_i and a homeomorphism $\mu_i : Y_i \times (a_i, a_{i+1}) \to X^{(a_i,a_{i+1})}$ s.t. $\forall i = 0, ..., n, f|_{X^{(a_i,a_{i+1})}} = \pi_2 \circ \mu_i^{-1}$, where π_2 is the projection onto the second factor;
- **(ii)** $\forall i = 1, ..., n-1, \mu_i$ extends to a continuous function $\bar{\mu}_i : Y_i \times [a_i, a_{i+1}] \to X^{[a_i,a_{i+1}]}$; similarly, μ_0 extends to $\bar{\mu}_0 : Y_0 \times (-\infty, a_1] \to X^{(-\infty,a_1]}$ and μ_n extends to $\bar{\mu}_n : Y_n \times [a_n, +\infty) \to X^{[a_n,+\infty)}$;
- **(iii)** Each levelset X^t has a finitely-generated homology.

All Morse functions on a smooth manifold are of Morse type. However, the converse is not true. In fact, Morse-type functions do not have to be differentiable and their domain does not have to be a smooth manifold nor even a manifold at all.

2.2 Extended Persistence

Let f be a real-valued function on a topological space X. The family $\{X^{(-\infty,\alpha]}\}_{\alpha\in\mathbb{R}}$ of sublevel sets of f defines a *filtration*, that is, it is nested w.r.t. inclusion: $X^{(-\infty,\alpha]} \subseteq X^{(-\infty,\beta]}$ for all $\alpha \leq \beta \in \mathbb{R}$. The family $\{X^{[\alpha,+\infty)}\}_{\alpha\in\mathbb{R}}$ of superlevel sets of f is also nested but in the opposite direction: $X^{[\alpha,+\infty)} \supseteq X^{[\beta,+\infty)}$ for all $\alpha \leq \beta \in \mathbb{R}$. We can turn it into a filtration by reversing the real line. Specifically, let $\mathbb{R}^{\mathrm{op}} = \{\tilde{x} \mid x \in \mathbb{R}\}$, ordered by $\tilde{x} \leq \tilde{y} \Leftrightarrow x \geq y$. We index the family of superlevel sets by $\mathbb{R}^{\mathrm{op}}$, so now we have a filtration: $\{X^{[\tilde{\alpha},+\infty)}\}_{\tilde{\alpha}\in\mathbb{R}^{\mathrm{op}}}$, with $X^{[\tilde{\alpha},+\infty)} \subseteq X^{[\tilde{\beta},+\infty)}$ for all $\tilde{\alpha} \leq \tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$.

Extended persistence connects the two filtrations at infinity as follows. First, replace each superlevel set $X^{[\tilde{\alpha},+\infty)}$ by the pair of spaces $(X, X^{[\tilde{\alpha},+\infty)})$ in the second filtration. This maintains the filtration property since we have $(X, X^{[\tilde{\alpha},+\infty)}) \subseteq (X, X^{[\tilde{\beta},+\infty)})$ for all $\tilde{\alpha} \leq \tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$. Then, let $\mathbb{R}_{\mathrm{Ext}} = \mathbb{R} \cup \{+\infty\} \cup \mathbb{R}^{\mathrm{op}}$, where the order is completed by $\alpha < +\infty < \tilde{\beta}$ for all $\alpha \in \mathbb{R}$ and $\tilde{\beta} \in \mathbb{R}^{\mathrm{op}}$. This poset is isomorphic to $(\mathbb{R}, \leq)$. Finally, define the *extended filtration* of f over $\mathbb{R}_{\mathrm{Ext}}$ by:

$$F_\alpha = X^{(-\infty,\alpha]} \text{ for } \alpha \in \mathbb{R},\ F_{+\infty} = X \equiv (X, \emptyset) \text{ and } F_{\tilde{\alpha}} = (X, X^{[\tilde{\alpha},+\infty)}) \text{ for } \tilde{\alpha} \in \mathbb{R}^{\mathrm{op}},$$

where we have identified the space X with the pair of spaces $(X, \emptyset)$. The subfamily $\{F_\alpha\}_{\alpha\in\mathbb{R}}$ is called the *ordinary* part of the filtration, while $\{F_{\tilde{\alpha}}\}_{\tilde{\alpha}\in\mathbb{R}^{\mathrm{op}}}$ is called the *relative* part.

Applying the homology functor H_* to this filtration gives the so-called *extended persistence module* $\mathbb{V}$ of f, which is a sequence of vector spaces connected by linear maps induced by the inclusions in the extended filtration. For functions of Morse type, the extended persistence module can be decomposed as a finite direct sum of half-open *interval modules*—see e.g. [12]: $\mathbb{V} \simeq \bigoplus_{k=1}^{n} \mathbb{I}[b_k, d_k)$, where each summand $\mathbb{I}[b_k, d_k)$ is made of copies of the field of coefficients at every index $\alpha \in [b_k, d_k)$, and of copies of the zero space elsewhere, the maps between copies of the field being identities. Each summand represents the lifespan of a *homological feature* (cc, hole, void, etc.) within the filtration. More precisely, the *birth time* b_k and *death*

time d_k of the feature are given by the endpoints of the interval. Then, a convenient way to represent the structure of the module is to plot each interval in the decomposition as a point in the extended plane, whose coordinates are given by the endpoints. Such a plot is called the *extended persistence diagram* (PD) of f, denoted $\mathrm{Dg}(f)$. The distinction between ordinary and relative parts of the filtration allows us to classify the points in $\mathrm{Dg}(f)$ as follows:

- $p = (x, y)$ is called an *ordinary* point if $x, y \in \mathbb{R}$;
- $p = (x, y)$ is called a *relative* point if $x, y \in \mathbb{R}^{\mathrm{op}}$;
- $p = (x, y)$ is called an *extended* point if $x \in \mathbb{R}, y \in \mathbb{R}^{\mathrm{op}}$;

Note that ordinary points lie strictly above the diagonal $\Delta = \{(x, x) \mid x \in \mathbb{R}\}$ and relative points lie strictly below Δ, while extended points can be located anywhere, including on Δ (e.g. when a cc lies inside a single critical level, see Section 2.3). It is common to partition $\mathrm{Dg}(f)$ according to this classification: $\mathrm{Dg}(f) = \mathrm{Ord}(f) \sqcup \mathrm{Rel}(f) \sqcup \mathrm{Ext}^+(f) \sqcup \mathrm{Ext}^-(f)$, where by convention $\mathrm{Ext}^+(f)$ includes the extended points located on the diagonal Δ.

Stability. An important property of extended PDs is to be stable in the so-called *bottleneck distance* d_{b}^{∞}. Given two PDs D, D', a *partial matching* between D and D' is a subset Γ of $D \times D'$ where for every $p \in D$ there is at most one $p' \in D'$ such that $(p, p') \in \Gamma$, and conversely, for every $p' \in D'$ there is at most one $p \in D$ such that $(p, p') \in \Gamma$. Furthermore, Γ must match points of the same type (ordinary, relative, extended) and of the same homological dimension only. The *cost* of Γ is: $\mathrm{cost}(\Gamma) = \max\{\max_{p \in D} \delta_D(p), \max_{p' \in D'} \delta_{D'}(p')\}$, where $\delta_D(p) = \|p - p'\|_\infty$ if p is matched to some $p' \in D'$ and $\delta_D(p) = d_\infty(p, \Delta)$ if p is unmatched – same for $\delta_{D'}(p')$.

▶ **Definition 2.2.** Let D, D' be two PDs. The *bottleneck distance* between D and D' is $d_{\mathrm{b}}^{\infty}(D, D') = \inf_\Gamma \mathrm{cost}(\Gamma)$, where Γ ranges over all partial matchings between D and D'.

Note that d_{b}^{∞} is only a pseudo-metric, not a true metric, because points lying on Δ can be left unmatched at no cost.

▶ **Theorem 2.3** (Stability [15]). *For any Morse-type functions $f, g : X \to \mathbb{R}$,*

$$d_{\mathrm{b}}^{\infty}(\mathrm{Dg}(f), \mathrm{Dg}(g)) \leq \|f - g\|_\infty.$$

Moreover, as pointed out in [15], the theorem can be strengthened to apply to each subdiagram $\mathrm{Ord}, \mathrm{Ext}^+, \mathrm{Ext}^-, \mathrm{Rel}$ and to each homological dimension individually.

2.3 Reeb Graphs

▶ **Definition 2.4.** Given a topological space X and a continuous function $f : X \to \mathbb{R}$, we define the equivalence relation $\sim_f$ between points of X by $x \sim_f y$ if and only if $f(x) = f(y)$ and x, y belong to the same cc of $f^{-1}(f(x)) = f^{-1}(f(y))$. The *Reeb graph* $\mathrm{R}_f(X)$ is the quotient space $X/\sim_f$.

As f is constant on equivalence classes, there is an induced quotient map $\tilde{f} : \mathrm{R}_f(X) \to \mathbb{R}$ with $f = \tilde{f} \circ \pi$, where π is the projection $X \to \mathrm{R}_f(X)$ induced by $\sim_f$. If f is a function of Morse type, then the pair (X, f) is an $\mathbb{R}$-*constructible space* in the sense of [17]. This ensures that the Reeb graph is a multigraph, whose nodes are in one-to-one correspondence with the cc of the critical level sets of f. We can equip this multigraph with a metric by assigning the length $l(v_i, v_j) = |f(v_i) - f(v_j)|$ to each edge (v_i, v_j). In the following, the combinatorial version of the Reeb graph is denoted by $\mathcal{C}(\mathrm{R}_f(X))$.

Connection to the extended persistence. There is a nice interpretation of $\mathrm{Dg}(\tilde{f})$ in terms of the structure of $\mathrm{R}_f(X)$. We refer the reader to [4, 15] and the references therein for a full description as well as formal definitions and statements. Orienting the Reeb graph vertically so $\tilde{f}$ is the height function, we can see each cc of the graph as a trunk with multiple branches (some oriented upwards, others oriented downwards) and holes. Then, one has the following correspondences, where the *vertical span* of a feature is the span of its image by $\tilde{f}$:

- The vertical spans of the trunks are given by the points in $\mathrm{Ext}_0^+(\tilde{f})$;
- The vertical spans of the downward branches are given by the points in $\mathrm{Ord}_0(\tilde{f})$;
- The vertical spans of the upward branches are given by the points in $\mathrm{Rel}_1(\tilde{f})$;
- The vertical spans of the holes are given by the points in $\mathrm{Ext}_1^-(\tilde{f})$.

The rest of the diagram of $\tilde{f}$ is empty. These correspondences provide a dictionary to read off the structure of the Reeb graph from the PD of the quotient map $\tilde{f}$. Note that it is a bag-of-features type of descriptor, taking an inventory of all the features together with their vertical spans, but leaving aside the actual layout of the features. As a consequence, it is an incomplete descriptor: two Reeb graphs with the same PD may not be isomorphic.

The following theorem summarizes the known connections between $\mathrm{Dg}(\tilde{f})$ and $\mathrm{Dg}(f)$. It formalizes the intuition that the Reeb graph captures part of the topological structure of f, the missing features being either "inessential", or "horizontal", or "higer-dimensional". We provide a proof in the full version [9] for completeness.

▶ **Theorem 2.5.** *Let X be a topological space and $f : X \to \mathbb{R}$ a function of Morse type. Then, $\mathrm{Dg}(\tilde{f}) \subseteq \mathrm{Dg}(f)$. More precisely:*

$$\mathrm{Dg}_0(\tilde{f}) = \mathrm{Dg}_0(f); \quad \mathrm{Dg}_1(\tilde{f}) = \mathrm{Dg}_1(f) \setminus (\mathrm{Ext}_1^+(f) \cup \mathrm{Ord}_1(f)); \quad \mathrm{Dg}_p(\tilde{f}) = \emptyset \ \forall p \geq 2.$$

2.4 Covers and Nerves

Let Z be a topological space. A *cover* of Z is a family $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of subsets of Z, such that $Z = \bigcup_{\alpha \in A} U_\alpha$. It is *open* if all its elements are open subspaces of Z. It is *connected* if all its elements are connected subspaces of Z. Its *nerve* is the abstract simplicial complex $\mathcal{N}(\mathcal{U})$ that has a k-simplex per $(k+1)$-fold intersection of elements of $\mathcal{U}$:

$$\{\alpha_0, ..., \alpha_k\} \in \mathcal{N}(\mathcal{U}) \iff \bigcap_{i=0,...,k} U_{\alpha_i} \neq \emptyset.$$

When $\mathcal{V}$ itself is a cover of Z, it is called a *subcover* of $\mathcal{U}$. It is *proper* if it is not equal to $\mathcal{U}$. Finally, $\mathcal{U}$ is called *minimal* if it admits no proper subcover or, equivalently, if it has no element included in the union of the other elements. Given a minimal cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, for every $\alpha \in A$ we let $\tilde{U}_\alpha = U_\alpha \setminus \bigcup_{\alpha' \neq \alpha \in A} U_{\alpha'}$. The cc of $\tilde{U}_\alpha$ are called the *proper subsets* of U_α. $\mathcal{U}$ is called *generic* if no proper subset is a singleton.

Consider now the special case where Z is a subset of $\mathbb{R}$, equipped with the subspace topology. A subset $U \subseteq Z$ is an *interval of Z* if there is an interval I of $\mathbb{R}$ such that $U = I \cap Z$. Note that U is open in Z if and only if I can be chosen open in $\mathbb{R}$. A cover $\mathcal{U}$ of Z is an *interval cover* if all its elements are intervals. In this case, $\mathrm{End}(\mathcal{U})$ denotes the set of all of the interval endpoints. Finally, the *granularity* of $\mathcal{U}$ is the supremum of the lengths of its elements, i.e. it is the quantity $\sup_{U \in \mathcal{U}} |U|$ where $|U| := \sup(U) - \inf(U) \in \mathbb{R} \cup \{+\infty\}$.

▶ **Lemma 2.6.** *If $\mathcal{U}$ is a minimal open interval cover of $Z \subseteq \mathbb{R}$, then no more than two elements of $\mathcal{U}$ can intersect at a time. Moreover, if Z is $\mathbb{R}$ itself or a compact subset thereof, then any cover $\mathcal{U}$ of Z has a minimal subcover.*

From now on, unless otherwise stated, all covers of $Z \subseteq \mathbb{R}$ will be generic, open, minimal, interval covers (*gomic* for short). An immediate consequence of Lemma 2.6 is that every element U of a gomic $\mathcal{U}$ has exactly one proper subset $\tilde{U}$. More precisely, U partitions into three subintervals: $U = U_\cap^- \sqcup \tilde{U} \sqcup U_\cap^+$, where $U_\cap^-$ is the intersection of U with the element right below it in the cover ($U_\cap^- = \emptyset$ if that element does not exist), and where $U_\cap^+$ is the intersection of U with the element right above it ($U_\cap^+ = \emptyset$ if that element does not exist).

2.5 Mapper

Let $f : X \to Z$ be a continuous function. Consider a cover $\mathcal{U}$ of $\mathrm{im}(f)$, and pull it back to X via f^{-1}. Then, decompose every $V_\alpha = f^{-1}(U_\alpha) \subseteq X$ into its cc: $V_\alpha = \bigsqcup_{i \in \{1 \dots c(\alpha)\}} V_\alpha^i$, where $c(\alpha)$ is the number of cc of V_α. Then, $\mathcal{V} = \{V_\alpha^i\}_{\alpha \in A, i \in \{1 \dots c(\alpha)\}}$ is a connected cover of X. It is called the *connected pullback cover*, and its nerve $\mathcal{N}(\mathcal{V})$ is the Mapper.

▶ **Definition 2.7.** Let X, Z be topological spaces, $f : X \to Z$ a continuous function, $\mathcal{U}$ a cover of $\mathrm{im}(f)$ and $\mathcal{V}$ the associated cover of X. The *Mapper* of $(f, \mathcal{U})$ is $\mathrm{M}_f(X, \mathcal{U}) = \mathcal{N}(\mathcal{V})$.

See Figure 1 for an illustration. Note that, when $Z = \mathbb{R}$ and $\mathcal{U}$ is a gomic of $\mathrm{im}(f)$, the Mapper has a natural 1-dimensional stratification since no more than two intervals can intersect at a time by Lemma 2.6. Hence, in this case, it has the structure of a (possibly infinite) simple graph and therefore has trivial homology in dimension 2 and above. When $\mathcal{U}$ is not a gomic, the Mapper may not be a graph nor have trivial homology in dimension 2.

3 MultiNerve Mapper

Given a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of a topological space X, it is possible to extend the concept of nerve to a *simplicial poset* called the *multinerve*:

▶ **Definition 3.1** ([16])**.** The *multinerve* $\mathcal{M}(\mathcal{U})$ is the simplicial poset defined by:

$$\mathcal{M}(\mathcal{U}) = \{(\{\alpha_0, \cdots, \alpha_k\}, C) \;\mid\; \bigcap_{i=0,\cdots,k} U_{\alpha_i} \neq \emptyset \text{ and } C \text{ is a cc of } \bigcap_{i=0,\cdots,k} U_{\alpha_i}\}.$$

The proof that this set, together with the least element $(\emptyset, \bigcup_{\mathcal{U}})$ and equipped with the partial order $(F, C) \preceq (F', C') \iff F \subseteq F'$ and $C' \subseteq C$, is a simplicial poset, can be found in [16]. We extend the concept of Mapper by using the multinerve of the connected pullback cover instead of its nerve:

▶ **Definition 3.2.** Let X, Z be topological spaces, $f : X \to Z$ a continuous function, $\mathcal{U}$ a cover of $\mathrm{im}(f)$ and $\mathcal{V}$ the associated cover of X. The *MultiNerve Mapper* of X is $\overline{\mathrm{M}}_f(X, \mathcal{U}) = \mathcal{M}(\mathcal{V})$.

See Figure 1 for an illustration. Again, when $Z = \mathbb{R}$ and $\mathcal{U}$ is a gomic of $\mathrm{im}(f)$, the MultiNerve Mapper is a (possibly infinite) multigraph and therefore has trivial homology in dimension 2 and above. Contrary to the Mapper, it also takes the cc of the intersections into account. As we shall see in Section 5, the MultiNerve Mapper is able to capture the same features as the Mapper, even with coarser gomics, and is more naturally related to the Reeb graph.

The connection between the Mapper and the MultiNerve Mapper is induced by the connection between nerves and multinerves [16]: $\mathrm{M}_f(X, \mathcal{U}) = \pi_1(\overline{\mathrm{M}}_f(X, \mathcal{U}))$, where $\pi_1 : (F, C) \mapsto F$ is the projection of the simplices $(\{\alpha_0, \dots, \alpha_k\}, C)$ of the multinerve $\overline{\mathrm{M}}_f(X, \mathcal{U})$ onto their first coordinate. Thus, when $Z = \mathbb{R}$ and $\mathcal{U}$ is a gomic, the Mapper is the simple graph obtained by gluing the edges that have the same endpoints in the MultiNerve Mapper. In this special case, π_1 induces a surjective homomorphism in homology.

4 Telescope

In this section we introduce the telescopes, which are our main objects of study when we relate the structure of the MultiNerve Mapper to the one of the Reeb graph of a perturbation of the pair (X, f).

▶ **Definition 4.1** (Telescope [7])**.** A *telescope* is an adjunction space of the following form:

$$T = (Y_0 \times (a_0, a_1]) \cup_{\psi_0} (X_1 \times \{a_1\}) \cup_{\phi_1} (Y_1 \times [a_1, a_2]) \cup_{\psi_1} \ldots \cup_{\phi_n} (Y_n \times [a_n, a_{n+1})),$$

where $a_0 = -\infty$ and $a_{n+1} = +\infty$ by convention and where the $\phi_i : Y_i \times \{a_i\} \to X_i \times \{a_i\}$ and $\psi_i : Y_i \times \{a_{i+1}\} \to X_{i+1} \times \{a_{i+1}\}$ are continuous maps. The a_i are called the *critical values* of T and their set is denoted by $\mathrm{Crit}(T)$, the ϕ_i and ψ_i are called *attaching maps*, the Y_i are compact and locally connected spaces called the *cylinders* and the X_i are topological spaces called the *critical slices*. Moreover, all Y_i and X_i have finitely-generated homology.

A telescope comes equipped with π_1 and π_2, which are the projections onto the first factor and second factor respectively. Given any interval I, we let $T^I = \pi_1 \circ \pi_2^{-1}(I)$.

A function of Morse type $f : X \to \mathbb{R}$ naturally induces a telescope T_X defined with $\mathrm{Crit}(T) = \mathrm{Crit}(f)$, $X_i = f^{-1}(\{a_i\})$, $Y_i = \pi_1 \circ \mu_i^{-1} \circ f^{-1}(a_i, a_{i+1})$, $\phi_i = (\bar{\mu}_i|_{Y_i \times \{a_i\}}, \mathrm{id})$ and $\psi_i = (\bar{\mu}_i|_{Y_i \times \{a_{i+1}\}}, \mathrm{id})$. One can define a homeomorphism $\mu : X \to T_X$ such that $f = \pi_2 \circ \mu$, so that $\mathrm{Dg}(\pi_2) = \mathrm{Dg}(f)$. We refer the reader e.g. to [17] for more details.

Operations

We will now present three kinds of perturbations on telescopes that preserve the MultiNerve Mapper, namely: *Merge*, *Split*, and *Shift*. For this we will use generalized attaching maps:

$$\phi_i^a : Y_i \times \{a\} \to X_i \times \{a\};\ (y, a) \mapsto (\pi_1 \circ \phi_i(y, a_i), a),$$
$$\psi_i^a : Y_i \times \{a\} \to X_{i+1} \times \{a\};\ (y, a) \mapsto (\pi_1 \circ \psi_i(y, a_{i+1}), a).$$

▶ **Definition 4.2** (Merge)**.** Let T be a telescope. Let $a \leq b$. If $[a, b]$ contains at least one critical value, i.e. $a_{i-1} < a \leq a_i \leq a_j \leq b < a_{j+1}$, then the Merge on T between a, b is the telescope $T' = \mathrm{Merge}_{a,b}(T)$ given by:

$$\ldots (Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} \ldots \cup_{\psi_{j-1}} (X_j \times \{a_j\}) \cup_{\phi_j} (Y_j \times [a_j, a_{j+1}]) \ldots$$
$$\downarrow$$
$$\ldots (Y_{i-1} \times [a_{i-1}, \bar{a}]) \cup_{f_{i-1}} (T^{[a,b]} \times \{\bar{a}\}) \cup_{g_j} (Y_j \times [\bar{a}, a_{j+1}]) \ldots$$

where $\bar{a} = \frac{a+b}{2}$, where $f_{i-1} = \psi_{i-1}^{\bar{a}}$ if $a = a_i$ and $f_{i-1} = \mathrm{id}_{Y_{i-1} \times \{\bar{a}\}}$ otherwise, and where $g_j = \phi_j^{\bar{a}}$ if $b = a_j$ and $g_j = \mathrm{id}_{Y_j \times \{\bar{a}\}}$ otherwise.

If $[a, b]$ contains no critical value, i.e. $a_{i-1} < a \leq b < a_i$, then $\mathrm{Merge}_{a,b}(T)$ is given by:

$$\ldots (X_{i-1} \times \{a_{i-1}\}) \cup_{\phi_{i-1}} (Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \ldots$$
$$\downarrow$$
$$\ldots (X_{i-1} \times \{a_{i-1}\}) \cup_{\phi_{i-1}} (Y_{i-1} \times [a_{i-1}, \bar{a}]) \cup_{f_{i-1}} (T^{[a,b]} \times \{\bar{a}\}) \cup_{g_{i-1}} (Y_{i-1} \times [\bar{a}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \ldots$$

where $\bar{a} = \frac{a+b}{2}$, and where $f_{i-1} = g_{i-1} = \mathrm{id}_{Y_{i-1} \times \{\bar{a}\}}$.

See Figure 2 for an illustration.

Similarly, we define the Merge between a, b on a diagram D as the diagram $\mathrm{Merge}_{a,b}(D)$ given by:

$$\mathrm{Merge}_{a,b}(x, y) = (\bar{x}, \bar{y}) \text{ where } \bar{x} = \begin{cases} x \text{ if } x \notin [a, b] \\ \bar{a} \text{ otherwise} \end{cases} \text{ and similarly for } y.$$

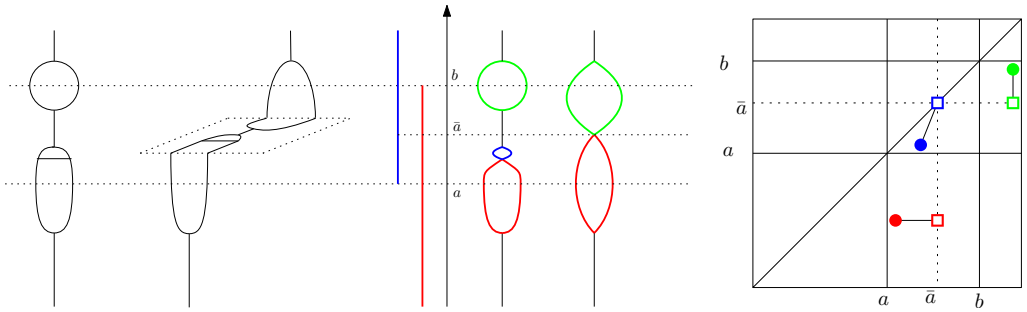


Figure 2 Left: Effect of a Merge. Middle: Effect on the corresponding Reeb graph. Right: Effect on the corresponding extended PD of dimension 1.

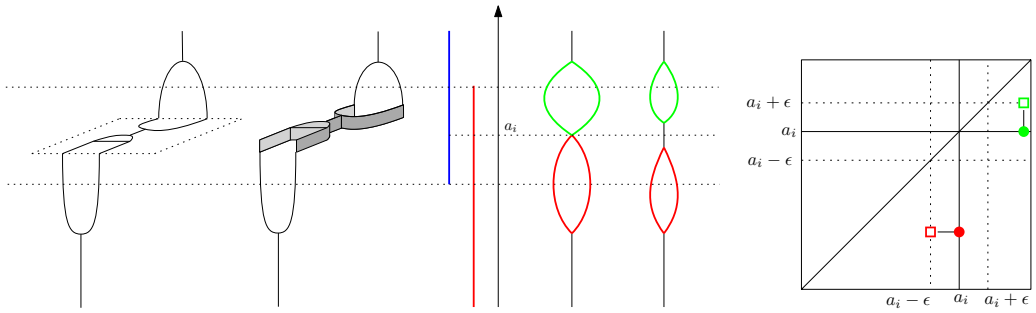


Figure 3 Left: Effect of a Split. Middle: Effect on the corresponding Reeb graph. Right: Effect on the corresponding extended PD of dimension 1.

▶ **Lemma 4.3.** *Let $a \leq b$ and $T' = \mathrm{Merge}_{a,b}(T)$. Let $\pi_2' : T' \to \mathbb{R}$ be the projection onto the second factor. Then, $\mathrm{Dg}(\pi_2') = \mathrm{Merge}_{a,b}(\mathrm{Dg}(\pi_2))$.*

▶ **Definition 4.4** (Split)**.** Let T be a telescope. Let $a_i \in \mathrm{Crit}(T)$ and ϵ s.t. $0 \leq \epsilon < \min\{a_{i+1} - a_i, a_i - a_{i-1}\}$. The ϵ-Split on T at a_i is the telescope $T' = \mathrm{Split}_{\epsilon,a_i}(T)$ given by:

$$\begin{array}{c} \ldots (Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} (Y_i \times [a_i, a_{i+1}]) \ldots \\ \downarrow \\ \ldots (Y_{i-1} \times [a_{i-1}, a_i - \epsilon]) \cup_{\psi_{i-1}^{a_i-\epsilon}} (X_i \times \{a_i - \epsilon\}) \cup_{\mathrm{id}} (X_i \times [a_i - \epsilon, a_i + \epsilon]) \cup_{\mathrm{id}} (X_i \times \{a_i + \epsilon\}) \cup_{\phi_i^{a_i+\epsilon}} (Y_i \times [a_i + \epsilon, a_{i+1}]) \ldots \end{array}$$

See Figure 3 for an illustration.

Similarly, we define the ϵ-Split at a_i on a diagram D as the diagram $\mathrm{Split}_{\epsilon,a_i}(D)$ given by:

$$\mathrm{Split}_{\epsilon,a_i}(x,y) = (\bar{x}, \bar{y}) \text{ where } \bar{x} = \begin{cases} x \text{ if } x \neq a_i \\ a_i + \epsilon \text{ if } (a_i, y) \in \mathrm{Rel} \\ a_i - \epsilon \text{ otherwise} \end{cases} \text{ and } \bar{y} = \begin{cases} y \text{ if } y \neq a_i \\ a_i - \epsilon \text{ if } (x, a_i) \in \mathrm{Ord} \\ a_i + \epsilon \text{ otherwise} \end{cases}$$

Note that the definition of $\mathrm{Split}_{\epsilon,a_i}(D)$ assumes implicitly that D contains no point within the horizontal and vertical bands $[a_i - \epsilon, a_i) \times \mathbb{R}$, $(a_i, a_i + \epsilon] \times \mathbb{R}$, $\mathbb{R} \times [a_i - \epsilon, a_i)$ and $\mathbb{R} \times (a_i, a_i + \epsilon]$, which is the case under the assumptions of Definition 4.4.

A critical value $a_i \in \mathrm{Crit}(T)$ is called an *up-fork* if ψ_{i-1} is an homeomorphism, and it is called a *down-fork* if ϕ_i is an homeomorphism. The new attaching maps introduced by the Split are identity maps, hence the following lemma:

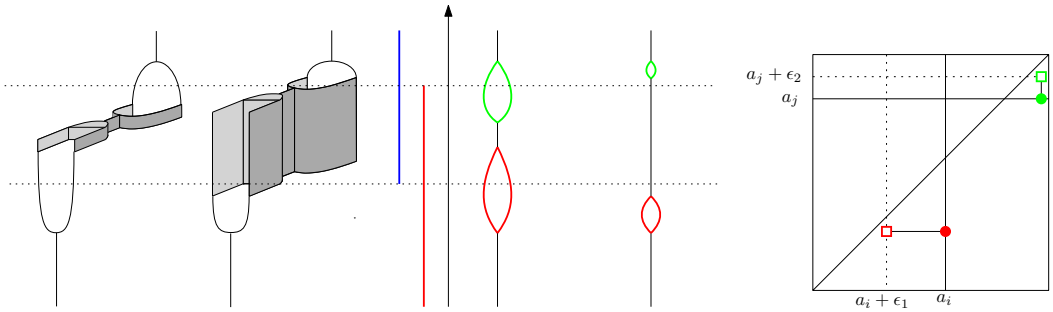


Figure 4 Left: Effect of a double Shift with amplitudes $\epsilon_1 < 0 < \epsilon_2$. Middle: Effect on the corresponding Reeb graph. Right: Effect on the corresponding extended PD of dimension 1.

▶ **Lemma 4.5.** *The new critical values $a_i - \epsilon$ and $a_i + \epsilon$ created after a Split are down- and up-forks respectively.*

▶ **Lemma 4.6.** *Let $a_i \in \mathrm{Crit}(T)$. Let $0 < \epsilon < \min\{a_{i+1} - a_i, a_i - a_{i-1}\}$, $T' = \mathrm{Split}_{\epsilon,a_i}(T)$ and $\pi_2' : T' \to \mathbb{R}$ the projection onto the second factor. Then, $\mathrm{Dg}(\pi_2') = \mathrm{Split}_{\epsilon,a_i}(\mathrm{Dg}(\pi_2))$.*

▶ **Definition 4.7** (Shift)**.** Let T be a telescope. Let $a_i \in \mathrm{Crit}(T)$ and ϵ s.t. $0 \leq |\epsilon| < \min\{a_{i+1} - a_i, a_i - a_{i-1}\}$. The ϵ-Shift on T at a_i is the telescope $T' = \mathrm{Shift}_{\epsilon,a_i}(T)$ given by:

$$\begin{array}{c} \ldots(Y_{i-1} \times [a_{i-1}, a_i]) \cup_{\psi_{i-1}} (X_i \times \{a_i\}) \cup_{\phi_i} (Y_i \times [a_i, a_{i+1}])\ldots \\ \downarrow \\ \ldots(Y_{i-1} \times [a_{i-1}, a_i + \epsilon]) \cup_{\psi_{i-1}^{a_i+\epsilon}} (X_i \times \{a_i + \epsilon\}) \cup_{\phi_i^{a_i+\epsilon}} (Y_i \times [a_i + \epsilon, a_{i+1}])\ldots \end{array}$$

See Figure 4 for an illustration. Similarly, we define the ϵ-Shift at a_i on a diagram D as the diagram $\mathrm{Shift}_{\epsilon,a_i}(D)$ given by:

$$\mathrm{Shift}_{\epsilon,a_i}(x,y) = (\bar{x}, \bar{y}) \text{ where } \bar{x} = \begin{cases} x \text{ if } x \neq a_i \\ a_i + \epsilon \text{ otherwise} \end{cases} \quad \text{and similarly for } y$$

Note that the definition of $\mathrm{Shift}_{\epsilon,a_i}(D)$ assumes implicitly that D contains no point within the horizontal and vertical bands delimited by a_i and $a_i + \varepsilon$, which is the case under the assumptions of Definition 4.7.

▶ **Lemma 4.8.** *Let $a_i \in \mathrm{Crit}(T)$, ϵ s.t. $0 < |\epsilon| < \min\{a_{i+1} - a_i, a_i - a_{i-1}\}$, $T' = \mathrm{Shift}_{\epsilon,a_i}(T)$ and $\pi_2' : T' \to \mathbb{R}$ the projection onto the second factor. Then, $\mathrm{Dg}(\pi_2') = \mathrm{Shift}_{\epsilon,a_i}(\mathrm{Dg}(\pi_2))$.*

Invariance

The above operations leave the (MultiNerve) Mapper unchanged under certain conditions:

▶ **Proposition 4.9.** *Let T be a telescope and $\mathcal{I}$ be a gomic of $\mathrm{im}(\pi_2)$.*

(i) *Let $a \leq b$ s.t. a, b belong to the same intersection $I \cap J$ or proper interval $\tilde{I}$. Then, $\overline{\mathrm{M}}_{\pi_2}(\mathrm{Merge}_{a,b}(T), \mathcal{I})$ is isomorphic to $\overline{\mathrm{M}}_{\pi_2}(T, \mathcal{I})$.*

(ii) *Let $a_i \in \mathrm{Crit}(T) \setminus \mathrm{End}(\mathcal{I})$, and $a < a_i < b$ with a, b consecutive in $\mathrm{End}(\mathcal{I})$. If $\epsilon < \min\{a_i - a, b - a_i\}$, then $\overline{\mathrm{M}}_{\pi_2}(\mathrm{Split}_{\epsilon,a_i}(T), \mathcal{I})$ is isomorphic to $\overline{\mathrm{M}}_{\pi_2}(T, \mathcal{I})$.*

(iii) *Let $a_i \in \mathrm{Crit}(T) \setminus \mathrm{End}(\mathcal{I})$, and $b < a_i < c < d$ with b, c, d consecutive in $\mathrm{End}(\mathcal{I})$. If a_i is an up-fork, $(b, c) = I \cap J$ is an intersection, and $c - a_i < \epsilon < \min\{d, a_{i+1}\} - a_i$, then $\overline{\mathrm{M}}_{\pi_2}(\mathrm{Shift}_{\epsilon,a_i}(T), \mathcal{I})$ is isomorphic to $\overline{\mathrm{M}}_{\pi_2}(T, \mathcal{I})$.*

(iv) *Let $a_i \in \mathrm{Crit}(T) \setminus \mathrm{End}(\mathcal{I})$, and $a < b < a_i < c$ with a, b, c consecutive in $\mathrm{End}(\mathcal{I})$. If a_i is a down-fork, $(b, c) = I \cap J$ is an intersection, and $\max\{a, a_{i-1}\} - a_i < \epsilon < b - a_i$, then $\overline{\mathrm{M}}_{\pi_2}(\mathrm{Shift}_{\epsilon,a_i}(T), \mathcal{I})$ is isomorphic to $\overline{\mathrm{M}}_{\pi_2}(T, \mathcal{I})$.*

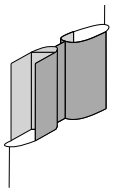
5 Structure of the MultiNerve Mapper

Let $f : X \to \mathbb{R}$ be of Morse type, and let $\mathcal{I}$ be a gomic of $\mathrm{im}(f)$. Let T_X be the corresponding telescope. In this section, we move out all critical values of the intersection preimages $f^{-1}(I \cap J)$, so that the MultiNerve Mapper and the Reeb graph become isomorphic. For any interval $I \in \mathcal{I}$, we let $a_{\tilde{I}} < b_{\tilde{I}}$ be the endpoints of its proper subinterval $\tilde{I}$, so we have $\tilde{I} = [a_{\tilde{I}}, b_{\tilde{I}}]$. For any non-empty intersection $I \cap J$, we fix a subinterval $[a_{I\cap J}, b_{I\cap J}] \subset I \cap J$ such that every critical value within $I \cap J$ falls into $[a_{I\cap J}, b_{I\cap J}]$. We define

$$T'_X := \mathrm{Merge}'_{\mathcal{I}} \circ \mathrm{Shift}_{\mathcal{I}} \circ \mathrm{Split}_{\mathcal{I}} \circ \mathrm{Merge}_{\mathcal{I}}(T_X), \tag{1}$$

where each operation is defined individually as follows:

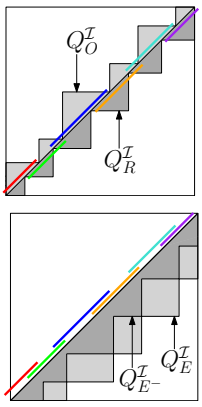
- $\mathrm{Merge}_{\mathcal{I}}$ is the composition of all the $\mathrm{Merge}_{a_{\tilde{I}}, b_{\tilde{I}}}$, $I \in \mathcal{I}$, and of all the $\mathrm{Merge}_{a_{I\cap J}, b_{I\cap J}}$, $I, J \in \mathcal{I}$ and $I \cap J \neq \emptyset$. All these functions commute, so their composition is well-defined. The same holds for the following compositions.
- $\mathrm{Split}_{\mathcal{I}}$ is the composition of all the $\mathrm{Split}_{\epsilon,\bar{a}}$ with $\bar{a}$ a critical value after the first $\mathrm{Merge}_{\mathcal{I}}$ (therefore not an interval endpoint) and $\epsilon > 0$ satisfying the assumptions of Prop. 4.9 (ii).
- $\mathrm{Shift}_{\mathcal{I}}$ is the composition of all the $\mathrm{Shift}_{\epsilon,\bar{a}_+}$ with $\bar{a}_+$ an up-fork critical value after the $\mathrm{Split}_{\mathcal{I}}$ and $\epsilon > 0$ such that the assumptions of Prop. 4.9 (iii) are satisfied, and of all the $\mathrm{Shift}_{\epsilon,\bar{a}_-}$ with $\bar{a}_-$ a down-fork critical value after the $\mathrm{Split}_{\mathcal{I}}$ and $\epsilon < 0$ such that the assumptions of Prop. 4.9 (iv) are satisfied. After $\mathrm{Shift}_{\mathcal{I}}$ there are no more critical values located in the intersections of consecutive intervals of $\mathcal{I}$.
- $\mathrm{Merge}'_{\mathcal{I}}$ is the composition of all the $\mathrm{Merge}_{a_{\tilde{I}}, b_{\tilde{I}}}$, $I \in \mathcal{I}$.



Combine Figures 2, 3 and 4 with the figure opposite for an illustration of this sequence of transformations. We let $\pi'_2 : T'_X \to \mathbb{R}$ be the projection onto the second factor. In the following, we identify the pair (T_X, π_2) with (X, f). since they are isomorphic in the category of $\mathbb{R}$-constructible spaces. We also rename π'_2 into f' for convenience. Let $\tilde{f}' : \mathrm{R}_{f'}(T'_X) \to \mathbb{R}$ be the induced quotient map.

▶ **Lemma 5.1.** *For T'_X defined as in (1), $\overline{\mathrm{M}}_{f'}(T'_X, \mathcal{I})$ is isomorphic to $\overline{\mathrm{M}}_f(X, \mathcal{I})$ as a combinatorial multigraph.*

The effect of (1) on the extended PD of f is illustrated in Figure 5. There are two grids in this figure: the one with solid lines is defined by the interval endpoints, while the one with dotted lines is defined by the critical values $\bar{a}$ introduced by the $\mathrm{Merge}_{\mathcal{I}}$. In the following, we use the term *cell* to designate a rectangle of the first grid. Cells are closed if they correspond to proper subintervals for both coordinates, they are open if they correspond to intersections for both coordinates, and they are neither closed nor open otherwise. Blue and green cells in Figure 5 correspond to squares associated to a proper subinterval (blue) or intersection (green).



Our first structure theorem (Theorem 5.2 below) involves certain unions of colored cells from Figure 5, called the *staircases* and defined as follows. Given an interval I (indifferently open, closed or half-open),

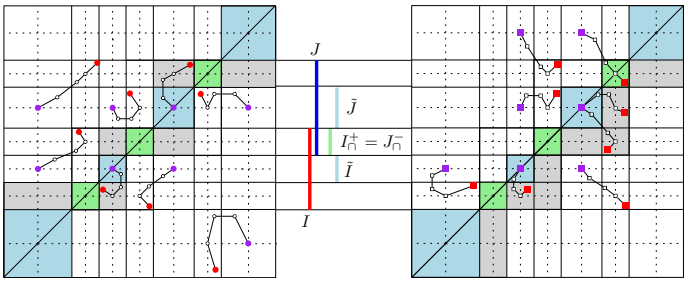


Figure 5 The left panel displays the trajectories of points in Ord (disks above the diagonal) and Rel (disks under the diagonal) while the right panel displays the trajectories of points in Ext. For both diagrams, the original point is red, the final point is purple and intersection and proper intervals are colored in green and light blue respectively.

let $Q_I^+ = \{(x,y) \in \mathbb{R}^2 \mid x \leq y \in I\}$ be the half-square above the diagonal, and $Q_I^- = \{(x,y) \in \mathbb{R}^2 \mid y < x \in I\}$ the half-square strictly below the diagonal. Decompose now each interval $I \in \mathcal{I}$ as $I = I_\cap^- \sqcup \tilde{I} \sqcup I_\cap^+ \in \mathcal{I}$, then let $Q_O^{\mathcal{I}} = \bigcup_{I\in\mathcal{I}} Q^+_{\tilde{I}\cup I_\cap^+}$, $Q_R^{\mathcal{I}} = \bigcup_{I\in\mathcal{I}} Q^-_{\tilde{I}\cup I_\cap^-}$ and $Q_{E^-}^{\mathcal{I}} = \bigcup_{I\in\mathcal{I}} Q_I^-$ be the staircases obtained by taking the unions of these half-squares, as illustrated in the figures on the right. Our structure theorem is stated as follows (recall that we have identified the pairs (X,f) and (T_X,π_2) and renamed π_2' into f'):

▶ **Theorem 5.2.** *For T_X' defined as in (1), for every $p \geq 0$ there is a perfect matching between:*

(i) $\mathrm{Ord}_p(f')$ *and* $\mathrm{Ord}_p(f) \setminus Q_O^{\mathcal{I}}$,
(ii) $\mathrm{Rel}_p(f')$ *and* $\mathrm{Rel}_p(f) \setminus Q_R^{\mathcal{I}}$,
(iii) $\mathrm{Ext}_p^-(f')$ *and* $\mathrm{Ext}_p^-(f) \setminus Q_{E^-}^{\mathcal{I}}$,
(iv) $\mathrm{Ext}_p^+(f')$ *and* $\mathrm{Ext}_p^+(f) \cup (\mathrm{Ext}_p^-(f) \cap Q_{E^-}^{\mathcal{I}})$.

The proof follows the tracking strategy illustrated in Figure 5. For each point of $\mathrm{Dg}(f)$, the results of Section 4 allow us to recreate its trajectory through the various operations of (1). Then, a series of simple observations (detailed in [9]) leads to the conclusion.

Theorems 2.5 and 5.2 together induce the following perfect matching between the persistence diagrams of the quotient maps[3]:

$$\mathrm{Ord}(\tilde{f}') \text{ and } \mathrm{Ord}(\tilde{f}) \setminus Q_O^{\mathcal{I}}; \quad \mathrm{Ext}(\tilde{f}') \text{ and } \mathrm{Ext}(\tilde{f}) \setminus Q_{E^-}^{\mathcal{I}}; \quad \mathrm{Rel}(\tilde{f}') \text{ and } \mathrm{Rel}(\tilde{f}) \setminus Q_R^{\mathcal{I}}. \quad (2)$$

Now we relate $\tilde{f}'$ to the MultiNerve Mapper through the Reeb graph of f', using the property that f' has exactly one critical value inside each proper interval and none outside:

▶ **Theorem 5.3.** *For T_X' defined as in (1), $\overline{\mathrm{M}}_f(X,\mathcal{I})$ is isomorphic to $\mathcal{C}(\mathrm{R}_{f'}(T_X'))$ as a combinatorial multigraph.*

A signature for MultiNerve Mapper. Theorem 5.3 means that the dictionary introduced in Section 2.3 can be used to describe the structure of the MultiNerve Mapper from the extended persistence diagram of the perturbed quotient function $\tilde{f}'$. $\mathrm{Dg}(\tilde{f}')$ is obtained from $\mathrm{Dg}(\tilde{f})$ as in (2). This suggests using the off-staircase part of $\mathrm{Dg}(\tilde{f})$ as a descriptor for the

[3] Note that $\mathrm{Ext}_0^-(g) = \emptyset$ for any Morse-type function g, including $g = \tilde{f}$, $g = f$, $g = f'$, and $g = \tilde{f}'$.

structure of the MultiNerve Mapper:

$$\begin{aligned}\mathrm{Dg}(\overline{\mathrm{M}}_f(X,\mathcal{I})) &= \mathrm{Ord}(\tilde{f}) \setminus Q_O^{\mathcal{I}} \cup \mathrm{Ext}(\tilde{f}) \setminus Q_{E^-}^{\mathcal{I}} \cup \mathrm{Rel}(\tilde{f}) \setminus Q_R^{\mathcal{I}} \\ &= \mathrm{Ord}_0(f) \setminus Q_O^{\mathcal{I}} \cup (\mathrm{Ext}_0^+(f) \cup \mathrm{Ext}_1^-(f)) \setminus Q_{E^-}^{\mathcal{I}} \cup \mathrm{Rel}_1(f) \setminus Q_R^{\mathcal{I}}, \end{aligned} \tag{3}$$

where the second equality comes from Theorem 2.5. We call this descriptor the *persistence diagram* of the MultiNerve Mapper. Note that this descriptor is not computed by applying persistence to some function defined on the multinerve, but it is rather a pruned version of the persistence diagram of $\tilde{f}$. As for Reeb graphs, it serves as a bag-of-features descriptor of the structure of $\overline{\mathrm{M}}_f(X,\mathcal{I})$. The fact that $\mathrm{Dg}(\overline{\mathrm{M}}_f(X,\mathcal{I})) \subseteq \mathrm{Dg}(\tilde{f})$ formalizes the intuition that the MultiNerve Mapper is a *pixelized version* of the Reeb graph, in which some of the features disappear due to the staircases (prescribed by the cover). The following convergence result is a direct consequence of our theorems:

▶ **Corollary 5.4.** *Suppose the granularity of the gomic $\mathcal{I}$ is at most ε. Then,*

$$\mathrm{Dg}(\tilde{f}) \setminus \{(x,y) \mid |y-x| \leq \varepsilon\} \subseteq \mathrm{Dg}(\overline{\mathrm{M}}_f(X,\mathcal{I})) \subseteq \mathrm{Dg}(\tilde{f}).$$

Thus, the features (branches, holes) of the Reeb graph that are missing in the MultiNerve Mapper have spans at most ε. Moreover, the two signatures become equal when ε is smaller than the smallest ℓ^∞-distance of the points of $\mathrm{Dg}(\tilde{f})$ to the diagonal. For even smaller ε, $\overline{\mathrm{M}}_f(X,\mathcal{I})$ and $\mathcal{C}(\mathrm{R}_f(X))$ become isomorphic as combinatorial multigraphs, up to vertex splits and edge subdivisions – see [9]. Note that convergence occurs before ε goes to zero.

Induced signature for Mapper. Recall from Section 3 that the projection $\pi_1 : \overline{\mathrm{M}}_f(X,\mathcal{I}) \to \mathrm{M}_f(X,\mathcal{I})$ induces a surjective homomorphism in homology. Thus, the Mapper has a simpler structure than the MultiNerve Mapper. To be more specific, π_1 identifies all the edges connecting the same pair of vertices. This eliminates the corresponding holes in $\overline{\mathrm{M}}_f(X,\mathcal{I})$. Since the two vertices lie in successive intervals of the cover, the corresponding diagram points lie in the following extended staircase: $Q_E^{\mathcal{I}} = \bigcup_{I \cup J \text{ s.t. } I,J \in \mathcal{I} \text{ and } I \cap J \neq \emptyset} Q_{I \cup J}^-$. The other staircases remain unchanged. Hence the following descriptor for the Mapper:

$$\begin{aligned}\mathrm{Dg}(\mathrm{M}_f(X,\mathcal{I})) &= \mathrm{Ord}(\tilde{f}) \setminus Q_O^{\mathcal{I}} \cup \mathrm{Ext}(\tilde{f}) \setminus Q_E^{\mathcal{I}} \cup \mathrm{Rel}(\tilde{f}) \setminus Q_R^{\mathcal{I}} \\ &= \mathrm{Ord}_0(f) \setminus Q_O^{\mathcal{I}} \cup (\mathrm{Ext}_0^+(f) \cup \mathrm{Ext}_1^-(f)) \setminus Q_E^{\mathcal{I}} \cup \mathrm{Rel}_1(f) \setminus Q_R^{\mathcal{I}}. \end{aligned} \tag{4}$$

The interpretation of this signature in terms of the structure of the Mapper follows the same rules as for the MultiNerve Mapper and Reeb graph. Moreover, the convergence result stated in Corollary 5.4 holds the same with the Mapper. See Figure 1 for an example that summarizes the different spaces and signatures.

6 Stability

Intuitively, for a point in the descriptor $\mathrm{Dg}(\overline{\mathrm{M}}_f(X,\mathcal{I}))$, the ℓ^∞-distance to its corresponding staircase ($Q_O^{\mathcal{I}}$, $Q_{E^-}^{\mathcal{I}}$ or $Q_R^{\mathcal{I}}$, depending on the type of the point) measures the amount by which the function f or the cover $\mathcal{I}$ must be perturbed in order to eliminate the corresponding feature (branch, hole) in the MultiNerve Mapper. Conversely, for a point in the Reeb graph's descriptor $\mathrm{Dg}(\tilde{f})$ that is not in the MultiNerve Mapper's descriptor (i.e. that lies inside the staircase), the ℓ^∞-distance to the boundary of the staircase measures the amount by which f or $\mathcal{I}$ must be perturbed in order to create a corresponding feature in the MultiNerve Mapper. Our aim here is to translate this intuition into stability results. For this we adapt the bottleneck distance so that it takes the staircases into account. Our results are stated for the MultiNerve Mapper, they hold as well for the Mapper (with $Q_{E^-}^{\mathcal{I}}$ replaced by $Q_E^{\mathcal{I}}$).

Stability w.r.t. perturbations of the function. Let Θ be a subset of $\mathbb{R}^2$. Given a partial matching Γ between two persistence diagrams D, D', the Θ-*cost* of Γ is: $\text{cost}_\Theta(\Gamma) = \max\{\max_{p\in D}\ \delta_D(p),\ \max_{p'\in D'}\ \delta_{D'}(p')\}$, where $\delta_D(p) = \|p - p'\|_\infty$ if $\exists p' \in D'$ s.t. $(p, p') \in \Gamma$ and $\delta_D(p) = d_\infty(p, \Theta)$ otherwise — same for $\delta_{D'}(p')$. The bottleneck distance becomes: $d^\infty_{\text{b},\Theta}(D, D') = \inf_\Gamma\ \text{cost}_\Theta(\Gamma)$, where Γ ranges over all partial matchings between D and D'. This is again a pseudometric and not a metric. To avoid heavy notations, we write d_Θ instead of $d^\infty_{\text{b},\Theta}$. Note that the usual bottleneck distance is obtained by taking Θ to be the diagonal $\Delta = \{(x, x) \mid x \in \mathbb{R}\}$. Given a gomic $\mathcal{I}$, we choose different sets Θ depending on the types of the points in the two diagrams. More precisely, we define the distance between descriptors as follows, where the notation D_* stands for the subdiagram of D of the right type (Ordinary, Extended or Relative):

$$d_\mathcal{I}(D, D') = \max\left\{d_{Q^\mathcal{I}_O}(D_O, D'_O),\ d_{Q^\mathcal{I}_{E^-}}(D_E, D'_E),\ d_{Q^\mathcal{I}_R}(D_R, D'_R)\right\}. \tag{5}$$

▶ **Theorem 6.1.** *Given a gomic $\mathcal{I}$, for any Morse-type functions $f, g : X \to \mathbb{R}$,*

$$d_\mathcal{I}(\text{Dg}(\overline{\text{M}}_f(X, \mathcal{I})), \text{Dg}(\overline{\text{M}}_g(X, \mathcal{I}))) \leq \|f - g\|_\infty.$$

This result follows easily from the definition of the signature in (3) and from the stability of extended persistence diagrams (Theorem 2.3). It can be extended likewise to perturbations of the domain using the framework of [10] and the stability result therein, see [9]. Note that the classical bottleneck distance d_Δ is unstable in this context.

Stability w.r.t. perturbations of the cover. Let us now fix the pair (X, f) and consider varying gomics. We aim for a quantification of the extent to which the structure of the (MultiNerve) Mapper may change as the gomic is perturbed. For this we adopt the dual point of view: for any two choices of gomics, we want to use the points of the diagram $\text{Dg}(f)$ to assess the degree by which the gomics differ.

The diagram points that discriminate between the two gomics are the ones located in the symmetric difference of the staircases, since they witness that the symmetric difference is non-empty. Given a persistence diagram D and two gomics $\mathcal{I}, \mathcal{J}$, we consider the quantity:

$$d_D(\mathcal{I}, \mathcal{J}) = \max_{*\in\{O,E^-,R\}}\ \sup_{p\in D_*\cap(Q^\mathcal{I}_*\triangle Q^\mathcal{J}_*)} \max\left\{d_\infty(p, Q^\mathcal{I}_*),\ d_\infty(p, Q^\mathcal{J}_*)\right\}, \tag{6}$$

where $\triangle$ denotes the symmetric difference, where D_* stands for the subdiagram of D of the right type (Ordinary, Extended or Relative), and where we adopt the convention that $\sup_{p\in\emptyset} \ldots$ is zero instead of infinite. Deriving an upper bound on $d_D(\mathcal{I}, \mathcal{J})$ in terms of the Hausdorff distance between the staircases is straightforward, since the supremum in (6) is taken over points that lie in the symmetric difference between the staircases:

▶ **Theorem 6.2.** *Given a Morse-type function $f : X \to \mathbb{R}$, for any gomics $\mathcal{I}, \mathcal{J}$,*

$$d_{\text{Dg}(\tilde{f})}(\mathcal{I}, \mathcal{J}) \leq \max_{*\in\{O,E^-,R\}} d^\infty_\text{H}(Q^\mathcal{I}_*,\ Q^\mathcal{J}_*),$$

where $\tilde{f}$ is the quotient map defined on the Reeb graph $\text{R}_f(X)$.

Moreover, we have

$$\text{Dg}_*(\tilde{f})\cap(Q^\mathcal{I}_*\triangle Q^\mathcal{J}_*) = (\text{Dg}_*(\tilde{f})\cap Q^\mathcal{I}_*)\triangle(\text{Dg}_*(\tilde{f})\cap Q^\mathcal{J}_*) = \text{Dg}_*(\overline{\text{M}}_f(X, \mathcal{I}))\triangle\text{Dg}_*(\overline{\text{M}}_f(X, \mathcal{J})),$$

where the second equality follows from the definition of the descriptor of the MultiNerve Mapper given in (3). Thus, $d_{\text{Dg}(\tilde{f})}(\mathcal{I}, \mathcal{J})$ quantifies the proximity of each descriptor to the

other staircase. In particular, having $d_{\mathrm{Dg}(\tilde{f})}(\mathcal{I},\mathcal{J}) = 0$ means that there are no diagram points in the symmetric difference, so the two gomics are equivalent from the viewpoint of the structure of the MultiNerve Mapper. Differently, having $d_{\mathrm{Dg}(\tilde{f})}(\mathcal{I},\mathcal{J}) > 0$ means that the structures of the two MultiNerve Mappers differ, and the value of $d_{\mathrm{Dg}(\tilde{f})}(\mathcal{I},\mathcal{J})$ quantifies by how much the covers should be perturbed to make the two multigraphs isomorphic.

7 Discrete Case

Building the signatures of $\overline{\mathrm{M}}_f(X,\mathcal{I})$ and $\mathrm{M}_f(X,\mathcal{I})$ requires to compute the critical values of f exactly, which may not always be possible. However, the stability properties presented in Section 6 can be exploited in practice to approximate the signatures from point cloud data. The approach boils down to applying known scalar field analysis techniques [13], then pruning the obtained persistence diagrams using the staircases. If one wants to further guarantee that the approximate signatures do correspond to some perturbed Mapper or MultiNerve Mapper, then one can use known techniques for Reeb graph approximation instead [19]. The details are given in the full version of the paper [9], together with a glimpse at other questions related to the discrete approximation of the Mapper (such as the statistical computation of confidence intervals and convergence rates to the Reeb graph) which we are currently working on.

Acknowledgements. We thank the anonymous referees for their insightful comments. This work was supported by ERC grant Gudhi and by ANR project TopData.

References

1 M. Alagappan. From 5 to 13: Redefining the Positions in Basketball. MIT Sloan Sports Analytics Conference, 2012.
2 A. Babu. Zigzag Coarsenings, Mapper Stability and Gene-network Analyses. 2013.
3 V. Barra and S. Biasotti. 3D Shape Retrieval and Classification using Multiple Kernel Learning on Extended Reeb Graphs. *The Visual Computer*, 30(11):1247–1259, 2014.
4 U. Bauer, X. Ge, and Y. Wang. Measuring Distance Between Reeb Graphs. In *Proc. 30th Sympos. Comput. Geom.*, pages 464–473, 2014.
5 U. Bauer, E. Munch, and Y. Wang. Strong Equivalence of the Interleaving and Functional Distortion Metrics for Reeb Graphs. In *Proc. 31st Sympos. Comput. Geom.*, 2015.
6 S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno. Reeb Graphs for Shape Analysis and Applications. *Theor. Comput. Sci.*, 392(1-3):5–22, 2008.
7 G. Carlsson, V. de Silva, and D. Morozov. Zigzag Persistent Homology and Real-valued Functions. In *Proc. 25th Sympos. Comput. Geom.*, pages 247–256, 2009.
8 H. Carr and D. Duke. Joint Contour Nets. *IEEE Trans. Vis. Comput. Graph.*, 20(8):1100–1113, 2014.
9 M. Carrière and S. Oudot. Structure and Stability of the 1-Dimensional Mapper. arXiv 1511.05823, 2015.
10 M. Carrière, S. Oudot, and M. Ovsjanikov. Stable Topological Signatures for Points on 3D Shapes. In *Proc. 13th Sympos. Geom. Proc.*, 2015.
11 A. Chattopadhyay, H. Carr, D. Duke, Z. Geng, and O. Saeki. Multivariate Topology Simplification. arXiv 1509.04465, 2015.
12 F. Chazal, V. de Silva, M. Glisse, and S. Oudot. The Structure and Stability of Persistence Modules. arXiv 1207.3674, 2012.
13 F. Chazal, L. Guibas, S. Oudot, and P. Skraba. Analysis of scalar fields over point cloud data. In *Proc. 20th Sympos. Discr. Algo.*, pages 1021–1030, 2009.

14 F. Chazal and J. Sun. Gromov-Hausdorff Approximation of Filament Structure Using Reeb-type Graph. In *Proc. 30th Sympos. Comput. Geom.*, 2014.

15 D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Found. Comput. Math.*, 9(1):79–103, 2009.

16 É. Colin de Verdière, G. Ginot, and X. Goaoc. Multinerves and helly numbers of acyclic families. In *Proc. 28th Sympos. Comput. Geom.*, pages 209–218, 2012.

17 V. de Silva, E. Munch, and A. Patel. Categorified Reeb Graphs. arXiv 1501.04147, 2015.

18 T. Dey, F. Mémoli, and Y. Wang. Mutiscale Mapper: A Framework for Topological Summarization of Data and Maps. arXiv 1504.03763, 2015.

19 T. Dey and Y. Wang. Reeb graphs: Approximation and persistence. *Discr. Comput. Geom.*, 49(1):46–73, 2013.

20 E. Munch and B. Wang. Convergence between Categorical Representations of Reeb Space and Mapper. In *Proc. 32nd Sympos. Comput. Geom.*, 2016.

21 M. Nicolau, A. Levine, and G. Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proc. National Acad. Sci.*, 108(17):7265–7270, 2011.

22 G. Singh, F. Mémoli, and G. Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In *Sympos. PB Graphics*, 2007.

23 R. B. Stovner. On the Mapper Algorithm. 2012.

Max-Sum Diversity Via Convex Programming*

Alfonso Cevallos[1], Friedrich Eisenbrand[2], and Rico Zenklusen[3]

1 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**
`alfonso.cevallosmanzano@epfl.ch`
2 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**
`friedrich.eisenbrand@epfl.ch`
3 **Swiss Federal Institute of Technology in Zürich (ETH Zürich), Zürich, Switzerland**
`ricoz@math.ethz.ch`

Abstract

Diversity maximization is an important concept in information retrieval, computational geometry and operations research. Usually, it is a variant of the following problem: Given a ground set, constraints, and a function $f(\cdot)$ that measures diversity of a subset, the task is to select a feasible subset S such that $f(S)$ is maximized. The *sum-dispersion* function $f(S) = \sum_{x,y \in S} d(x,y)$, which is the sum of the pairwise distances in S, is in this context a prominent diversification measure. The corresponding diversity maximization is the *max-sum* or *sum-sum diversification*. Many recent results deal with the design of constant-factor approximation algorithms of diversification problems involving sum-dispersion function under a matroid constraint.

In this paper, we present a PTAS for the max-sum diversification problem under a matroid constraint for distances $d(\cdot,\cdot)$ of *negative type*. Distances of negative type are, for example, metric distances stemming from the ℓ_2 and ℓ_1 norms, as well as the cosine or spherical, or Jaccard distance which are popular similarity metrics in web and image search.

Our algorithm is based on techniques developed in geometric algorithms like metric embeddings and convex optimization. We show that one can compute a fractional solution of the usually non-convex relaxation of the problem which yields an upper bound on the optimum integer solution. Starting from this fractional solution, we employ a deterministic rounding approach which only incurs a small loss in terms of objective, thus leading to a PTAS. This technique can be applied to other previously studied variants of the max-sum dispersion function, including combinations of diversity with linear-score maximization, improving the previous constant-factor approximation algorithms.

1998 ACM Subject Classification F.2.2 Geometrical problems and computations, I.3.5 Geometric algorithms, languages, and systems

Keywords and phrases Geometric Dispersion, Embeddings, Approximation Algorithms, Convex Programming, Matroids

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.26

1 Introduction

Diversification is an important concept in many areas of computing such as information retrieval, computational geometry or optimization. When searching for news on a particular subject, for example, one is usually confronted with several relevant search results. A news reader might be interested in news from various sources and viewpoints. Thus, on the one

* This work was partially supported by the Swiss National Science Foundation, grant number 146660.

© Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen;
licensed under Creative Commons License CC-BY

hand, the articles should be relevant to his search and, on the other hand, they should be significantly diverse.

The so-called *max-sum diversification* or *max-sum dispersion* is a diversity-measure that has been subject of study in operations research [19, 28, 5, 13] for a while and it is currently receiving considerable attention in the information retrieval literature [16, 4, 7]. It is readily described. Given a ground set X together with a distance function $d : X \times X \to \mathbb{R}_{\geq 0}$. The diversity, or dispersion, of a subset $S \subseteq X$ is the sum of the pairwise distances

$$f(S) = \sum_{i,j \in S} d(i,j).$$

Since documents are often represented as vectors in a high-dimensional space, their similarity is measured by norms in $\mathbb{R}^n$ and their induced distances, see, e.g., [24, 29]. Among the most frequent distances are the ones induced by the ℓ_1 and ℓ_2 norms, the *cosine distance* or the *Jaccard distance* [26]. These norms are also used to measure similarity via much lower-dimensional bit-vectors stemming from sketching techniques [9]. So, usually, the ground set X is a finite set of vectors in $\mathbb{R}^d$ and $d(\cdot,\cdot)$ is a metric on $\mathbb{R}^d$ which makes diversity maximization a *geometric optimization problem*.

Before we go on, we state the general version of the *maximum sum diversification* or *maximum sum dispersion* (MSD) problem which generalizes many previously studied variants. It is in the focus of this paper, and is described by the following quadratic integer programming problem:

$$\begin{aligned} \text{maximize} \quad & x^T D x \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0,1\} \text{ for } i \in [n], \end{aligned} \tag{1}$$

where the symmetric matrix $D \in \mathbb{R}^{n \times n}$ represents the distances between the points of a *distance space* and $Ax \leq b$ is a set of additional linear constraints where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. If $x \in \{0,1\}^n$ is the characteristic vector of the set $S \subseteq X$, then $x^T D x = \sum_{i,j \in S} d(i,j)$. We recall the notion of a distance space; see, e.g., [11]. It is a pair (X,d) where X is a finite set and $d(\cdot,\cdot)$ is the *distance* function $d : X \times X \to \mathbb{R}_{\geq 0}$. The function d satisfies $d(i,i) = 0$ and $d(i,j) = d(j,i)$ for all $i,j \in X$. If in addition d satisfies the triangle inequality $d(i,j) \leq d(i,k) + d(j,k)$ for all $i,j,k \in X$, then d is a *(semi) metric* and (X,d) a finite *metric space*.

We now mention some previous algorithmic work on diversity maximization with this objective function. For the case where $Ax \leq b$ represents one cardinality constraint, $\sum_{i=1}^n x_i \leq k$, and $d(\cdot,\cdot)$ is a metric, the problem is also coined MSD_k [7]. Constant-factor approximation algorithms for MSD_k have been developed in [28, 19]. Birnbaum and Goldman [5] presented an algorithm with approximation factor converging to $1/2$. This is tight under the assumption that the *planted clique* problem [3] is hard, see [7]. Fekete and Meijer [13] have shown that this problem has a PTAS for $X \subset \mathbb{R}^d$ and $d(\cdot,\cdot)$ being the ℓ_1-distance, provided that the dimension d is *fixed*. Bhattacharya et al. [4] developed a $1/2$-approximation algorithm for MSD_k where the objective function is replaced by $x^T D x + c^T x$ for some $c \in \mathbb{R}^n$. This has been useful in accommodating also *scores of documents* in the objective function.

Recently, Abbassi et al. [1] have shown that MSD has a $1/2$-approximation algorithm if $d(\cdot,\cdot)$ is a metric and $Ax \leq b$ models the independent sets of a *matroid*. This is particularly relevant in situations where documents are partitioned into subsets $D_1, \ldots, D_\ell$ and only p_i

results should be returned from partition D_i for each i. The possible sets are then *independent sets* of a *partition matroid*. The case of one cardinality constraint only is subsumed by $\ell = 1$. Thus, the tightness of their result also follows from the planted clique assumption as described in [7].

Contributions of this paper

The $\frac{1}{2} + \epsilon$ hardness of MSD_k [7] is based on a metric that does not play a prominent role as a similarity measure. Are there better approximation algorithms, possibly polynomial-time approximation schemes, for other relevant distance metrics?

We give a positive answer to this question for the case where $d(\cdot, \cdot)$ is a distance of *negative type*. We review the notion of negative-type distances in Section 2 and here only note that the previously mentioned distances, like the ones stemming from the ℓ_2 and ℓ_1 norms, as well as the *cosine-distance* and the *Jaccard distance* are, among many other relevant distance functions, of negative type. Our main result is the following.

▶ **Theorem 1.** *There exists a polynomial-time approximation scheme for* MSD *for the case that $d(\cdot, \cdot)$ is of negative type and $Ax \leq b$ is a matroid constraint. In particular, there is a polynomial-time approximation scheme for* MSD_k *for negative-type distances.*

Theorem 1 is shown by following these two steps.

1. We show that one can compute a fractional solution $x^* \in \mathbb{R}^n$ that fulfills all the constraints $Ax \leq b$ and satisfies ${x^*}^T D x^* \geq OPT$, where OPT is the objective-function value of the optimal solution. More precisely, our algorithm does not optimize over the natural relaxation, but only over a family of slices of it, one for each possible ℓ_1 norm of the solution vector. The key property we show and exploit is that the optimization problem on each slice is a convex optimization problem that can be attacked by standard techniques, despite the fact that the natural relaxation is not convex. This allows us to obtain an optimal solution to the relaxation via the ellipsoid method for a wide family of constraints $Ax \leq b$, even if only a separation oracle is given; in particular, this includes matroid polytopes [18].
2. For the case in which $Ax \leq b$ describes the convex hull of independent sets of a matroid, we furthermore describe a polynomial-time *rounding algorithm* that computes an integral feasible solution $\bar{x}$ which satisfies $\bar{x}^T D \bar{x} \geq \left(1 - c \cdot \frac{\log k}{k}\right) {x^*}^T D x^*$, for some universal constant c and where k is the *rank* of the matroid.

Thus, step 1) is via a suitable convexification of a non-convex relaxation. Convexifications have proved useful in the design of approximation algorithms before, see for example [33].

We also want to mention that we obtain similar results for the case where the objective function is a combination of max-sum dispersion and linear scores, a scenario that has been considered in [4]. Finally, to complement our results, we prove strong NP-hardness of MSD_k for negative-type distances. We prove as well that, for the rounding algorithm mentioned in point b), the approximation factor of $1 - O\left(\frac{\log k}{k}\right)$ almost matches the integrality gap of our relaxation, which we can show to be at least $1 - \frac{1}{k}$.

2 Preliminaries

In this section, we review some preliminaries that are required for the understanding of this paper. A *polynomial-time approximation scheme (PTAS)* for an optimization problem in

which the objective function is maximized, is an algorithm that, given an instance and any $\epsilon > 0$, computes a solution that has an objective function value of at least $(1-\epsilon) \cdot OPT$, in time polynomial in the size of the instance, where OPT denotes the objective-function value of the optimal solution. If the running time is also polynomial in ϵ^{-1}, the algorithm is called a *fully polynomial-time approximation scheme (FPTAS)*. Clearly, our rounding algorithm is a PTAS (but not FPTAS) since we can compute the optimal solution in the case where $\epsilon < c \cdot \frac{\log k}{k}$ by brute force.

Norms and embeddings

Our results rely heavily on the theory of embeddings. We review some notions that are relevant for us and refer to [11, 25] for a thorough account. For a vector $v = (v^1, \cdots, v^t)^T \in \mathbb{R}^t$, we define the ℓ_p norm in the usual way, $\|v\|_\infty := \max_{1 \le i \le t} |v^i|$, and $\|v\|_p := (\sum_{i=1}^t |v^i|^p)^{1/p}$ for $p \ge 1$, and we extend this last definition to $0 < p < 1$, even if these are not proper norms as they do not respect the triangle inequality. For $0 < p \le \infty$, the space (X, d) is *ℓ_p-embeddable* if there is a dimension t and a function $v : X \to \mathbb{R}^t$ (the isometric embedding), such that for all $i, j \in X$ we have $d(i,j) = \|v_j - v_i\|_p$. Any finite metric space is ℓ_∞-embeddable with the *Fréchet embedding* [14] $v_i^j = d(i,j)$ for $i, j \in X$.

For the remainder of this paper, we assume that $X = \{1, \ldots, n\}$ and $n \ge 2$. Let $b_1, \ldots, b_n$ be real coefficients. The inequality

$$\sum_{1 \le i,j \le n} b_i b_j x_{ij} \le 0$$

with variables x_{ij} is a *negative-type* inequality if $\sum_{i=1}^n b_i = 0$. The distance space (X, d) is of negative type if $d(\cdot,\cdot)$ satisfies all negative-type inequalities, i.e., $\sum_{1 \le i,j \le n} b_i b_j d(i,j) \le 0$ holds for all $b_1, \ldots, b_n \in \mathbb{R}$ with $\sum_{i=1}^n b_i = 0$. Schoenberg [30, 31] characterized the metric spaces that are ℓ_2-embeddable as those whose square distance is of negative type.

▶ **Theorem 2** ([30, 31]). *A finite distance space (X, d) is of negative type if and only if $(X, \sqrt{d})$ is ℓ_2-embeddable.*

The following assertions, which help identifying distance spaces of negative type, can be found in [11].

1. If (X, d) is a metric space, then $\left(X, d^{\log_2(\frac{n}{n-1})}\right)$ is of negative type. [12]
2. For any $0 < \alpha \le p \le 2$, if (X, d) is ℓ_p-embeddable, then (X, d^α) is of negative type. [31]
3. If (X, d) is of negative type, then $(X, f(d))$ is also of negative type for any of the following functions: $f(x) = \frac{x}{1+x}$, $f(x) = \ln(1+x)$, $f(x) = 1 - e^{-\lambda x}$ for $\lambda > 0$, and $f(x) = x^\alpha$ for $0 \le \alpha \le 1$. [30]

We now list some distance functions that are of negative type and which are often used in information retrieval and web search. The ℓ_1-metric is of negative type. This follows from the assertion 2 above with $\alpha = p = 1$. In fact, any ℓ_p-metric with $1 \le p \le 2$ is of negative type. The ℓ_1-metric is a prominent similarity measure in information retrieval [24] in particular when using sketching techniques [22] where data points are represented by small-dimensional bit-vectors whose Hamming-distance approximates the distance of the corresponding points.

Also, the *spherical* and *cosine* distances, which measure the distance of two points on the sphere $S^{(t-1)}$ by the angle Θ that they enclose and by $1 - \cos\Theta$, respectively, are of negative type. This follows from a result of Blumenthal [6], see also [11].

For subsets A, B of a finite ground set U, the *Jaccard distance* $d(A,B) = \frac{|A \triangle B|}{|A \cup B|}$ is of negative type [17]. Similarly, many other distances of sets are of negative type, such as *Simple Matching* $\frac{|A \triangle B|}{|U|}$, *Russell and Rao* $1 - \frac{|A \cap B|}{|U|}$, and *Dice* $\frac{|A \triangle B|}{|A|+|B|}$ distances (see [26, Table 5.1] for a more complete list). Assertion 3 above presents some examples of transformations of distance spaces that preserve this property, and thus permit to construct new spaces of negative type from existing ones.

It is important to remark that distance spaces of negative type are in general not metric, or vice-versa. Hence results for these two families of distance spaces are not directly comparable. For instance, the cosine distance and the Dice distance mentioned before are not metric.

Matroids and the matroid polytope

We mention some basic definitions and results on matroid theory, see, e.g., [32, Volume B] for a thorough account. A matroid $\mathcal{M}$ over a finite ground set X is a tuple $\mathcal{M} = (X, \mathcal{I})$, where $\mathcal{I} \subseteq 2^X$ is a family of *independent* sets with the following properties.

(M1) $\emptyset \in \mathcal{I}$.

(M2) If $A \subseteq B$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$.

(M3) If $A, B \in \mathcal{I}$ and $|A| > |B|$, then there exists an element $e \in A \setminus B$ such that $B \cup \{e\} \in \mathcal{I}$.

In particular, the family of all subsets of X of cardinality at most k, that is, $\mathcal{I} = \{S \subseteq X \colon \mid S \mid \leq k\}$, forms a matroid known as *uniform matroid of rank* k, often denoted by U_n^k. Thus, MSD_k can be understood as picking an independent set $S \in \mathcal{I}$ from the uniform matroid that maximizes the sum of the pairwise distances.

The *rank* $r(A)$ of $A \subseteq X$ is the maximum cardinality of an independent set contained in A. Any inclusion-wise maximal independent set B is called a *basis*, and a direct consequence of the definition of matroids is that all bases have the same cardinality $r(X)$, called the *rank of the matroid*.

The *matroid polytope* $P(\mathcal{M})$ is the convex hull of the characteristic vectors of the independents sets of the matroid $\mathcal{M}$. It can be described by the following inequalities: $P(\mathcal{M}) = \left\{x \in \mathbb{R}^n_{\geq 0} \colon \sum_{i \in A} x_i \leq r(A)\ \forall A \subseteq X\right\}$. The base polytope of a matroid $\mathcal{M}$ of rank k is the convex hull of all characteristic vectors of bases of $\mathcal{M}$, and is given by $P(\mathcal{M}) \cap \{x \in \mathbb{R}^n \colon \sum_{i=1}^n x_i = k\}$.

Convex quadratic programming

A *quadratic program* is an optimization problem of the form

$$\min\{x^T Q x + c^T x \colon x \in \mathbb{R}^n,\ Ax \leq b\},$$

where $Q \in \mathbb{R}^{n \times n}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. If Q is positive semidefinite, then it is a *convex quadratic program*. Convex quadratic programs can be solved in polynomial time with the ellipsoid method [20], see, e.g., [21]. This also holds if $Ax \leq b$ is not explicitly given but the *separation problem* for $Ax \leq b$ can be solved in polynomial time [18]. In this case the running time is polynomial in the input size of Q and c and the largest binary encoding length of the coefficients in A and numbers in b. The separation problem for the matroid polytope $P(\mathcal{M})$ can be solved in polynomial time, provided that one can efficiently decide whether a set $S \subseteq X$ is an independent set, see [18]. Moreover, the largest encoding length of the numbers in the above-mentioned description of the matroid polytope is $O(\log n)$. Thus a convex quadratic program over the matroid polytope can be solved in polynomial time.

3 A relaxation that can be solved by convex programming

We now describe how to efficiently compute a fractional point x^* f or the relaxation of (1) with an objective value ${x^*}^T D x^* \geq OPT$. The function $f : \mathbb{R}^n \to \mathbb{R}$, $f(x) = x^T D x$ is in general non-concave, even if the distances $d(i,j)$ are ℓ_2-embeddable or of negative type. However, we have the following useful observation.

▶ **Lemma 3.** *Let (X,d) be a finite distance space of negative type, then*

$$f(x) = x^T D x$$

is a concave function over the domain $\{x \in \mathbb{R}^n \colon \sum_{i=1}^n x_i = \alpha\}$ for each fixed $\alpha \in \mathbb{R}$.

Proof. The statement is equivalent to saying that, for any two distinct points $x, y \in \mathbb{R}^n$ such that $\sum_i x_i = \sum_i y_i$, the function $f(\cdot)$ is concave over the line connecting x and y. Or in other words, for any point $x \in \mathbb{R}^n$ and vector b with $\sum_i b = 0$, the function $g(\lambda) := f(x + \lambda b)$, $\lambda \in \mathbb{R}$, is concave. Now, since the distance is of negative type, and $\sum_i b = 0$, we obtain the negative-type inequality $b^T D b = \sum_{i,j} b_i b_j d(i,j) \leq 0$. The function $g(\lambda)$ can be written as

$$g(\lambda) = f(x + \lambda b) = (x + \lambda b)^T D (x + \lambda b) = x^T D x + 2\lambda b^T D x + \lambda^2 b^T D b;$$

and hence its second derivative is $\frac{d^2}{(d\lambda)^2} g(\lambda) = 2 b^T D b \leq 0$. This proves the lemma. ◀

Using Lemma 3, we can efficiently determine a relaxed solution for MSD on distance spaces of negative type for a wide class of constraints, by solving a family of convex problems, one for each possible ℓ_1 norm of the solution vector. There is a rich set of algorithms for convex optimization problems as we encounter here. In the following theorem, for simplicity, we focus on consequences stemming from the ellipsoid algorithm. The ellipsoid algorithm has the advantage that it only needs a separation oracle, and often allows us to obtain an optimal solution without any error. As a technical requirement, we need that the coefficients of the underlying linear constraints have small encoding lengths, which holds for most natural constraints.

▶ **Theorem 4.** *Consider the max-sum dispersion problem with general linear constraints, for which the separation problem can be solved in polynomial time*

$$\begin{aligned} &\textit{maximize} && x^T D x \\ &\textit{subject to} && A x \leq b \\ & && x_i \in \{0,1\} \ \textit{for} \ i \in X. \end{aligned} \tag{2}$$

If $d(\cdot,\cdot)$ is of negative type, then one can compute a fractional point $x^* \in [0,1]^n$ satisfying $Ax \leq b$ with ${x^*}^T D x^* \geq OPT$ in time polynomial in the input size and the maximal binary encoding length of any coefficient or right-hand side of $Ax \leq b$.

Before proving the theorem, we briefly discuss the above-mentioned dependence of the running time on the encoding length. Notice that if $Ax \leq b$ is given explicitly, then the claimed point x^* is always obtained in polynomial time because the encoding length of $Ax \leq b$ is part of the input. Similarly, Theorem 4 implies that we can obtain x^* efficiently for matroid polytopes, since the inequality-description mentioned in Section 2 has only $\{0,1\}$-coefficients and right-hand sides within $\{1,\ldots,n\}$. We highlight that our techniques can often be used even if the encoding length condition is not fulfilled by accepting a small additive error.

Proof. Using the ellipsoid method, we solve each of the following n convex quadratic programming problems that are parameterized by $\alpha \in \{1, \dots, n\}$

$$\begin{aligned} \text{maximize} \quad & x^T D x \\ \text{subject to} \quad & Ax \leq b \\ & \textstyle\sum_{i=1}^{n} x_i = \alpha \\ & 0 \leq x_i \leq 1, \text{ for } i \in X, \end{aligned} \tag{3}$$

with optimum solutions $x^{*,1}, \dots, x^{*,n}$ respectively; see [21, 18] for details on why an optimal solution can be obtained (without any additive error which is typical for many convex optimization techniques). Since each feasible $\bar{x} \in \{0,1\}^n$ satisfies one of these constraints, x^* being one of these solutions with largest objective function value is a point satisfying the claim. Clearly, x^* can be computed in polynomial time. ◀

▶ Remark. We notice that for very simple constraints, like a cardinality constraint where at most k elements can be picked, a randomized rounding approach [27] can now be employed. More precisely, when working with a cardinality constraint one can simply scale down the fractional solution x^* satisfying $\sum_{i=1}^{n} x_i^* = k$ by a $(1-\epsilon)$-factor to obtain $y = (1-\epsilon)x^*$, and then round each component of y independently. Independent rounding preserves the objective in expectation, and the number of picked elements is $(1-\epsilon)k$ in expectation and sharply concentrates around this value due to Chernoff-type concentration bounds. However, this simple rounding approach fails for more interesting constraint families like matroid constraints.

4 Negative-type MSD under matroid constraint

Consider the MSD problem for the case that the distance space is of negative type and $Ax \leq b$ is a matroid constraint. The main result of this section is the following theorem which immediately implies Theorem 1.

▶ **Theorem 5.** *There exists a deterministic algorithm for the* MSD *problem in distance spaces of negative type with a matroid constraint, which outputs in polynomial time a basis B with $\left(\chi^B\right)^T D \chi^B \geq \left(1 - c\frac{\log k}{k}\right)$ OPT, where k is the rank of the matroid and c is an absolute constant.*

We solve the relaxation to obtain an optimal fractional point over the matroid polytope, as in Theorem 4, and perform a deterministic rounding algorithm. The suggested rounding procedure has similarities with pipage rounding for matroid polytopes (see [8], which is based on work in [2]) and swap rounding [10], in the sense that it iteratively changes at most two components of the fractional point until an integral point is obtained. However, contrary to these previous procedures we need to judiciously choose the two coordinates. Also, our analysis differs substantially from the above-mentioned prior rounding procedures on matroids, since we deal with a quadratic objective function where we must accept a certain loss in the objective value due to rounding, because there is a strictly positive integrality gap. (Pipage rounding and swap rounding are typically applied in settings where the objective function is preserved in expectation.) Makarychev, Schudy, and Sviridenko [23] build up on the swap rounding procedure and show how to obtain concentration bounds

for polynomial objective functions. Their concentration results apply to general polynomial objective functions with coefficients in $[0,1]$; however, they are not strong enough for our purposes.

Our deterministic rounding algorithm exploits the fact that we are dealing with negative-type distance spaces, and shows that only a very small loss in the objective value is necessary to obtain an integral solution. In order to bound this loss we use a very general inequality, stemming from the definition of distance spaces of negative type, that compares the (fractional) dispersion of two sets to that of its union. Given a vector $x \in \mathbb{R}^n$ and a set $S \subseteq \{1, \cdots, n\}$, we define the restricted vector x^S as $x^S_i = x_i$ if $i \in S$, and 0 otherwise.

▶ **Lemma 6.** *Let $D \in \mathbb{R}^{n\times n}$ be the matrix representing a negative-type distance space. Given a vector $x \in \mathbb{R}^n_{\geq 0}$ of coefficients, and two disjoint sets $A, B \subseteq [n]$ such that $\|x^A\|_1 > 0$ and $\|x^B\|_1 > 0$, we have*

$$\frac{\left(x^{A\cup B}\right)^T Dx^{A\cup B}}{\|x^{A\cup B}\|_1} \geq \frac{\left(x^A\right)^T Dx^A}{\|x^A\|_1} + \frac{\left(x^B\right)^T Dx^B}{\|x^B\|_1}.$$

Proof. Define the vector $b \in \mathbb{R}^n$ as $b = \frac{x^A}{\|x^A\|_1} - \frac{x^B}{\|x^B\|_1}$. Since $\sum_{i=1}^n b_i = 0$, the inequality $b^T Db \leq 0$ is of negative type. Expanding it yields

$$\begin{aligned} 0 &\geq \left(\frac{x^A}{\|x^A\|_1} - \frac{x^B}{\|x^B\|_1}\right)^T D \left(\frac{x^A}{\|x^A\|_1} - \frac{x^B}{\|x^B\|_1}\right) \\ &= \frac{\left(x^A\right)^T Dx^A}{\|x^A\|_1^2} + \frac{\left(x^B\right)^T Dx^B}{\|x^B\|_1^2} - \frac{2\left(x^A\right)^T Dx^B}{\|x^A\|_1\|x^B\|_1}. \end{aligned}$$

Hence $2\left(x^A\right)^T Dx^B \geq \frac{\|x^B\|_1}{\|x^A\|_1}\left(x^A\right)^T Dx^A + \frac{\|x^A\|_1}{\|x^B\|_1}\left(x^B\right)^T Dx^B$. Finally,

$$\begin{aligned} \left(x^{A\cup B}\right)^T Dx^{A\cup B} &= \left(x^A + x^B\right)^T D(x^A + x^B) \\ &= \left(x^A\right)^T Dx^A + \left(x^B\right)^T Dx^B + 2\left(x^A\right)^T Dx^B \\ &\geq \left(\|x^A\|_1 + \|x^B\|_1\right)\left(\frac{\left(x^A\right)^T Dx^A}{\|x^A\|_1} + \frac{\left(x^B\right)^T Dx^B}{\|x^B\|_1}\right) \\ &= \|x^{A\cup B}\|_1\left(\frac{\left(x^A\right)^T Dx^A}{\|x^A\|_1} + \frac{\left(x^B\right)^T Dx^B}{\|x^B\|_1}\right). \end{aligned}$$

◀

Consider an MSD instance consisting of a distance space of negative type represented by a matrix $D \in \mathbb{R}^{n\times n}$, and a matroid $\mathcal{M}$ over the ground set $X = \{1, \ldots, n\}$ of rank k. We assume that one can efficiently decide whether a set $S \subseteq X$ is independent. We apply Theorem 4 to find a fractional vector x^* over the matroid polytope $P(\mathcal{M}) = \left\{x \in \mathbb{R}^n_{\geq 0} \colon \sum_{i\in A} x_i \leq r(A)\ \forall A \subseteq X\right\}$, with $\left(x^*\right)^T Dx^* \geq \mathrm{OPT}$. Due to the monotonicity of the diversity function $x^T Dx$, we can assume that $\|x^*\|_1 = k$, i.e., x^* is on the base polytope of the matroid $\mathcal{M}$.[1] We describe now a deterministic rounding algorithm that takes x^* as input, and outputs in polynomial time a basis B of $\mathcal{M}$ with $\left(\chi^B\right)^T D\chi^B \geq \left(1 - O\left(\frac{\log k}{k}\right)\right)\left(x^*\right)^T Dx^* \geq \left(1 - O\left(\frac{\log k}{k}\right)\right)\mathrm{OPT}$.

[1] Indeed, using standard techniques from matroid optimization (see [32, Volume B]), one can, for any point $y \in P(\mathcal{M})$, determine a point $z \in P(\mathcal{M}) \cap \{x \in \mathbb{R}^n \colon \|x\|_1 = k\}$ satisfying $z \geq y$ component-wise. Hence, if the fractional point x^* we obtain is not on the base polytope, we can replace it efficiently with a point on the base polytope that dominates it and therefore has no worse objective value than x^* due to monotonicity of the considered objective.

In the remainder of the section, for any vector $x \in P(\mathcal{M})$ we ignore the elements i with $x_i = 0$ and assume without loss of generality that x has no zero components. We call an element $i \in X$ *integral* or *fractional* (with respect to x), respectively, if $x_i = 1$ or $x_i < 1$, and we call a set $S \subseteq X$ *tight* or *loose*, respectively, if $\|x^S\|_1 = r(S)$ or $\|x^S\|_1 < r(S)$. We will need the following result about faces of the matroid polytope, which is a well-known consequence of combinatorial uncrossing (see [15], or [32, Section 44.6c in Volume B]).

▶ **Lemma 7.** *Let $x \in P(\mathcal{M})$ with $x_i \neq 0$ for $i \in \{1, \dots, n\}$, and let $\emptyset = S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_p = X$ be a (inclusion-wise) maximal chain of tight sets with respect to x, i.e., $\sum_{i \in S_l} x_i = r(S_l)$ for $l \in \{1, \dots, p\}$. Then the polytope $P(\mathcal{M}) \cap \{y \in \mathbb{R}^n_{\geq 0} \colon \sum_{i \in S_l} y_i = r(S_l) \text{ for } l \in \{1, \dots, p\}\}$ defines the minimal face of $P(\mathcal{M})$ that contains x. (In other words, all other x-tight sets are implied by the ones in the chain.)*

Also, given a point $x \in P(\mathcal{M})$, one can efficiently find a maximal chain of tight sets as described in Lemma 7. Our algorithm starts with such a chain $\emptyset = S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_p = X$ for the vector x^*. For $1 \leq l \leq p$ define the set $R_l = S_l \setminus S_{l-1}$; we call these sets *rings*. The rings form a partition of X, their weights $\| (x^*)^{R_l} \|_1 = r(S_l) - r(S_{l-1})$ are strictly positive integers whose sum is k, and each ring R_l either consists of a single integral element, or of at least 2 elements, all fractional. This is because whenever $i \in R_l$ is integral, the set $S_{l-1} \cup \{i\}$ is tight, hence it can be added to the chain. We call the rings integral or fractional, accordingly. We start with $x = x^*$, we iteratively change two coordinates of x without leaving the minimal face of the matroid polytope on which x lies; one coordinate will be increased and the other one decreased by the same amount.

The rounding procedure

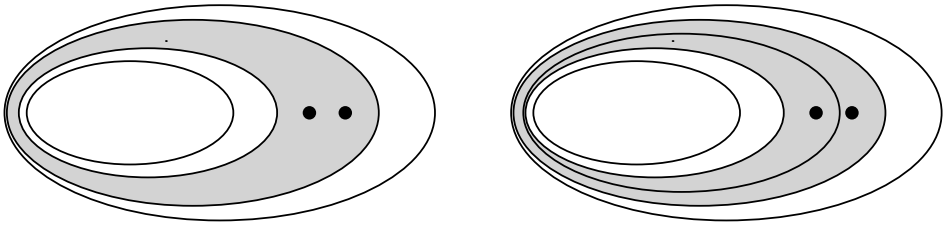
Starting with $x = x^*$, the rounding of x proceeds in iterations, and stops when all elements are integral. Among all fractional rings, and all pairs of fractional elements within the same ring, select the pair i, j that minimizes the term $x_i x_j d(i, j)$. We perturb vector x by adding to x_i and subtracting from x_j a certain quantity ϵ. The dispersion $x^T D x$ is linear in ϵ except for the term $2x_i x_j d(i, j)$, hence we can select the sign of ϵ so that the value of $x^T D x - 2x_i x_j d(i, j)$ does not decrease. We assume without loss of generality that this choice is $\epsilon > 0$, hence x_i is increasing and x_j decreasing. We increment ϵ until a new tight constraint appears. If the constraint corresponds to x_j becoming zero, we erase that element and end the iteration step. Otherwise, a previously loose set $S \subseteq X$ becomes tight, and S must contain i but not j, else its weight $\|x^S\|_1$ would not increase during this process. If the ring containing i and j is $R_l = S_l \setminus S_{l-1}$, then the set $S' = (S \cup S_{l-1}) \cap S_l$ is also tight,[2] and it also contains i but not j, hence $S_{l_1} \subsetneq S' \subsetneq S_l$ (see Figure 1). We add S' to the chain, update the list of rings, and end the iteration step.

We now argue that the above-described rounding procedure runs in polynomial time and computes the characteristic vector χ^B of a basis B of the matroid $\mathcal{M}$ with

$$\left(\chi^B\right)^T D \chi^B \geq \left(1 - c\frac{\log k}{k}\right) x^{*T} D x^* \tag{4}$$

where k is the rank of the matroid and c is an absolute constant.

[2] This follows from the uncrossing property: if A and B are tight sets then $A \cup B$ and $A \cap B$ are also tight. This property is a consequence of the submodularity of the matroid rank function.



(a) The fractional ring containing the pair i, j with minimal value $x_i^m x_j^m d(i,j)$.

(b) Elements i and j are separated by a new tight set that fits in the chain structure.

Figure 1 The refinement of a fractional ring in an iteration of the rounding procedure.

At any stage of the algorithm, if q is the number of fractional rings and f is the number of fractional elements, the number of iterations remaining is at most $f - q$. This is because the value $f - q$ can never be negative, and it decreases in each iteration. Either f decreases, or q increases, or q decreases by 1 but f decreases by at least 2 (any disappearing fractional ring has at least 2 fractional elements that become integral).

Suppose there are M iterations. We enumerate them in reverse order, and add a superscript m to all variables to signify their value when there are m iterations remaining. Hence $x^0 = \chi^B$ is the integral output vector, x^1 is the vector at the beginning of the last iteration, and so on until $x^M = x^*$. Clearly, all vectors x^m are in $P(\mathcal{M})$, and their weights $\|x^m\|_1 = k$ remain unchanged, hence each x^m is on the base polytope, and x^0 will be the characteristic vector of a basis in $\mathcal{M}$. From the previous claim we know that $m \le f^m - q^m$, and in particular $M \le n$, hence the algorithm runs in polynomial time. For $1 \le m \le M$, define $\text{loss}^m = (x^m)^T Dx^m - \left(x^{m-1}\right)^T Dx^{m-1}$, hence the total additive loss incurred in the rounding algorithm is $\sum_{m=1}^{M} \text{loss}^m$. We postpone for a moment the proof of the following inequality.

▶ **Lemma 8.** *The loss in iteration m is bounded by*

$$\text{loss}^m \le \min\left\{\frac{2}{m \cdot k}, \frac{2}{m^2}\right\} (x^*)^T Dx^*.$$

The total additive loss incurred by the algorithm is

$$\begin{aligned}(x^*)^T Dx^* - \left(x^0\right)^T Dx^0 = \sum_{m=1}^{M} \text{loss}^m &\le (x^*)^T Dx^* \left(\sum_{m=1}^{k} \frac{2}{m \cdot k} + \sum_{m>k} \frac{2}{m^2}\right) \\ &\le (x^*)^T Dx^* \cdot 2\left(\frac{1+\ln k}{k} + \frac{1}{k}\right) = (x^*)^T Dx^* \cdot \left(\frac{4+2\ln k}{k}\right) \ ,\end{aligned}$$

where the second inequality follows from $\sum_{m=1}^{k} \frac{1}{m} \le 1 + \ln k$ and $\sum_{m>k} \frac{1}{m^2} \le \frac{1}{k}$. In summary, the algorithm finds a basis with dispersion

$$\left(x^0\right)^T Dx^0 \ge (x^*)^T Dx^* \left(1 - \frac{4+2\ln k}{k}\right) \ge \text{OPT}\left(1 - O\left(\frac{\log k}{k}\right)\right)$$

which is our main result.

Proof of Lemma 8. If the pair i, j of fractional elements is chosen during the m-th iteration, then $\text{loss}^m \le 2x_i^m x_j^m d(i,j)$. We bound this term from above, using the inequality in Lemma 6 multiple times over the vector x^m and its partition into rings. We skip the superscript m to simplify notation and obtain

$$\frac{x^T Dx}{k} \geq \sum_{\text{ring } R} \frac{\left(x^R\right)^T Dx^R}{\|x^R\|_1} \geq \sum_{\text{fractional } R} \frac{\left(x^R\right)^T Dx^R}{\|x^R\|_1} \geq \sum_{\text{fractional } R} \frac{\binom{|R|}{2}\text{loss}}{\|x^R\|_1} \ ,$$

where the last inequality comes from $\text{loss}^m \leq 2x_i^m x_j^m d(i,j)$, and the choice of elements i and j that minimizes this quantity over all pairs of elements within any fractional ring. We complete the above inequality in two ways. First, for every fractional ring R, $|R| \geq \|x^R\|_1$, hence $\frac{\binom{|R|}{2}}{\|x^R\|_1} \geq \frac{|R|-1}{2}$ and thus

$$\frac{x^T Dx}{k} \geq \frac{\text{loss}}{2} \sum_{\text{frac. } R} (|R|-1) = \frac{\text{loss}}{2}(f-q) \geq \frac{m}{2}\text{loss} \ ,$$

which implies

$$\text{loss}^m \leq \frac{2}{m \cdot k} \left(x^m\right)^T Dx^m \leq \frac{2}{m \cdot k} \left(x^*\right)^T Dx^* \ .$$

Next, $\binom{|R|}{2} \geq (|R|-1)^2/2$, and using a Cauchy-Schwarz inequality (for the third inequality below), we get:

$$\begin{aligned} x^T Dx \geq & \frac{k \cdot \text{loss}}{2} \sum_{\text{frac. } R} \frac{(|R|-1)^2}{\|x^R\|_1} \geq \frac{\text{loss}}{2} \left(\sum_{\text{frac. } R} \|x^R\|_1 \right) \left(\sum_{\text{frac. } R} \frac{(|R|-1)^2}{\|x^R\|_1} \right) \\ \geq & \frac{\text{loss}}{2} \left(\sum_{\text{frac.} R} (|R|-1) \right)^2 = \frac{\text{loss}}{2}(f-q)^2 \geq \frac{m^2}{2}\text{loss} \ , \end{aligned}$$

and hence,

$$\text{loss}^m \leq \frac{2}{m^2} \left(x^m\right)^T Dx^m \leq \frac{2}{m^2} \left(x^*\right)^T Dx^* \ . \qquad \blacktriangleleft$$

▶ Remark. The integrality gap of the convex program $\max\{x^T Dx \mid x \in P(\mathcal{M})\}$ above is at least $1 - \frac{1}{k}$, which matches the approximation factor of our rounding algorithm up to a logarithmic term. Consider the matrix D with $D_{i,j} = 1$ for all $i \neq j$, which defines a distance space of negative type, and a cardinality constraint corresponding to the polytope $\{x \in [0,1]^n \mid \sum_i x_i = k\}$. An optimal solution is any k-set $B \subseteq X$, with value $\text{OPT} = (\chi^B)^T D \chi^B = k(k-1)$; but the fractional vector $x^* = (k/n, \cdots, k/n)$ is feasible and has value $(x^*)^T Dx^* = \frac{k^2}{n^2} n(n-1)$. Hence, $\frac{\text{OPT}}{(x^*)^T Dx^*} = \frac{k-1}{k} \frac{n}{n-1} \to 1 - \frac{1}{k}$, as $n \to \infty$.

▶ Remark. The previous approximation extends to the more general case of a combination of dispersion and linear scores, as follows. Consider the problem of maximizing the objective function $g(x) = x^T Dx + w^T x$, with a matroid constraint of rank k, and where D represents a distance space of negative type. The vector w here corresponds to non-negative *scores* on the elements of the ground set, and the objective is to find a feasible set with both high dispersion and high scores. The extra linear term does not change the concavity of $x^T Dx$, hence Lemma 3 and Theorem 5 are valid for this problem and provide a fractional vector $x^* \in P(\mathcal{M})$ with $g(x^*) \geq \text{OPT}$. Moreover, $g(x)$ is still monotone, which means we can assume that $\|x^*\|_1 = k$. In each iteration of this section's rounding algorithm, $g(x) - 2x_i x_j d(i,j)$ is linear in ϵ, so we can bound the loss of value of $g(x)$ during this iteration by $2x_i x_j d(i,j)$, as before. Hence, the above analysis still holds and shows that the total loss is very small, even

when comparing it only to the contribution of the quadratic term $(x^*)^T Dx^*$ to the objective, and ignoring the additional nonnegative term $w^T x^*$. Thus, we get the same approximation guarantee for this setting.

To conclude, we prove that MSD remains strongly NP-hard on distance spaces of negative type, even for a cardinality constraint. For this, we give a reduction from Densest k-Subgraph (DkS), which is strongly NP-hard. An instance of DkS consists of a graph $G = (V, E)$ and a number $k \leq n = |V|$, and the object is to find a k-set $W \subseteq V$ whose induced subgraph $G[W]$ contains the largest number of edges. Now, the distance function $d' : V^2 \to \mathbb{R}_{\geq 0}$ defined by $d'(i, j) = 2$ if $\{i, j\} \in E$, 1 if $\{i, j\} \notin E$ is metric.[3] Thus, by assertion 1) (below Theorem 2), we conclude that the distance $d = (d')^{\log_2 \frac{n}{n-1}}$ is of negative type, where $d(i, j) = \frac{n}{n-1}$ if $\{i, j\} \in E$, 1 otherwise. Finally, it is evident that an exact solution to the MSD instance (V, d) with cardinality constraint k corresponds to an exact solution to the DkS instance. This completes the proof. As is typical for many strongly NP-hard problems, one can easily deduce that an FPTAS for MSD for distance spaces of negative type subject to a cardinality constraint could be transformed into an exact algorithm for the same problem by choosing the error parameter ϵ sufficiently small. We therefore obtain the following.

▶ **Theorem 9.** *Max-sum dispersion* MSD *is strongly* NP*-hard, even for distance spaces of negative type and $Ax \leq b$ representing one cardinality constraint. In particular, this problem does not admit a fully polynomial-time approximation scheme unless* P = NP*.*

Finally, we can show that a randomized version of our rounding algorithm allows for dealing with a combination of a matroid and a constant number of knapsack constraints. Due to space constraints, we defer the proof of the following result to a long version of this paper.

▶ **Theorem 10.** *There exists a randomized polynomial-time approximation scheme for* MSD *for the case that $d(\cdot, \cdot)$ is of negative type and $Ax \leq b$ describes one matroid constraint and $O(1)$ knapsack constraints.*

References

1. Z. Abbassi, V. S. Mirrokni, and M. Thakur. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–40. ACM, 2013.
2. A. A. Ageev and M. I. Sviridenko. Pipage rounding: a new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(23):307–328, 2004.
3. N. Alon, S. Arora, R. Manokaran, D. Moshkovitz, and O. Weinstein. Inapproximability of densest κ-subgraph from average case hardness. *Unpublished manuscript*, 2011.
4. S. Bhattacharya, S. Gollapudi, and K. Munagala. Consideration set generation in commerce search. In *Proceedings of the 20th international conference on World wide web*, pages 317–326. ACM, 2011.
5. B. Birnbaum and K. J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 55(1):42–59, 2009.
6. L. M. Blumenthal. *Theory and Applications of Distance Geometry*, volume 347. Oxford, 1953.

[3] Any distance space, where the distance between distinct points is either 1 or 2, is metric.

7 A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st Symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.

8 G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

9 M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388. ACM, 2002.

10 C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science*, pages 575–584, 2010.

11 M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer-Verlag, Berlin, 1997.

12 M. M. Deza and H. Maehara. Metric transforms and euclidean embeddings. *Transactions of the American Mathematical Society*, 317(2):661–671, 1990.

13 S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2004.

14 M. Fréchet. Les dimensions d'un ensemble abstrait. *Mathematische Annalen*, 68(2):145–168, 1910.

15 F. R. Giles. *Submodular Functions, Graphs and Integer Polyhedra*. PhD thesis, University of Waterloo, 1975.

16 S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, pages 381–390. ACM, 2009.

17 J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3(1):5–48, 1986.

18 M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

19 R. Hassin, S. Rubinstein, and A. Tamir. Approximation algorithms for maximum dispersion. *Operations Research Letters*, 21(3):133–137, 1997.

20 L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1097, 1979.

21 M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228, 1980.

22 Q. Lv, M. Charikar, and K. Li. Image similarity search with compact data structures. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 208–217. ACM, 2004.

23 K Makarychev, W Schudy, and M Sviridenko. Concentration inequalities for nonlinear matroid intersection. *Random Structures & Algorithms*, 46(3):541–571, 2015.

24 C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*, volume 1. Cambridge university press Cambridge, 2008.

25 J. Matoušek. Lecture notes on metric embeddings. `kam.mff.cuni.cz/~matousek/ba-a4.pdf`, 2013.

26 E. Pekalska and R. P. W. Duin. *The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence)*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2005.

27 P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

28 S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

29 G. Salton and M. J. MacGill. Introduction to modern information retrieval. *McGraw-Hill computer science series*, 1983.

30 I. J. Schoenberg. Metric spaces and completely monotone functions. *Annals of Mathematics*, pages 811–841, 1938.

31 I. J. Schoenberg. Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536, 1938.

32 A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

33 M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2):206–242, 2001.

Dynamic Streaming Algorithms for ε-Kernels

Timothy M. Chan*

Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
tmchan@uwaterloo.ca

Abstract

Introduced by Agarwal, Har-Peled, and Varadarajan [J. ACM, 2004], an *ε-kernel* of a point set is a coreset that can be used to approximate the width, minimum enclosing cylinder, minimum bounding box, and solve various related geometric optimization problems. Such coresets form one of the most important tools in the design of linear-time approximation algorithms in computational geometry, as well as efficient insertion-only streaming algorithms and dynamic (non-streaming) data structures. In this paper, we continue the theme and explore dynamic streaming algorithms (in the so-called turnstile model).

Andoni and Nguyen [SODA 2012] described a dynamic streaming algorithm for maintaining a $(1+\varepsilon)$-approximation of the width using $O(\text{polylog}\, U)$ space and update time for a point set in $[U]^d$ for any constant dimension d and any constant $\varepsilon > 0$. Their sketch, based on a *polynomial method*, does not explicitly maintain an ε-kernel. We extend their method to maintain an ε-kernel, and at the same time reduce some of logarithmic factors. As an application, we obtain the first randomized dynamic streaming algorithm for the width problem (and related geometric optimization problems) that supports k *outliers*, using $\text{poly}(k, \log U)$ space and time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases coresets, streaming algorithms, dynamic algorithms, polynomial method, randomization, outliers

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.27

1 Introduction

Given a set S of n points in a constant dimension d, define the *width* (or *extent*) of S along a given vector ξ to be $w(S, \xi) = \max_{s \in S} s^T \xi - \min_{s \in S} s^T \xi$. The *width* of S is the minimum width over all unit vectors ξ.

The seminal paper by Agarwal, Har-Peled, and Varadarajan [1] introduced the concept of *ε-kernels*, which are *coresets* for the width along all directions simultaneously: an ε-kernel of S is a subset $Q \subset S$ such that $w(Q, \xi) \geq (1-\varepsilon) w(S, \xi)$ for all vectors ξ. Agarwal et al. showed that an ε-kernel of constant $O((1/\varepsilon)^{(d-1)/2})$ size always exists and can be constructed efficiently in $O(n)$ time for any constant $\varepsilon > 0$ (the hidden dependency of the running time on ε was subsequently improved by the author [9]). Using ε-kernels, we can immediately obtain efficient $(1+\varepsilon)$-factor approximation algorithms not only for computing the width of a point set, but for numerous other geometric optimization problems, such as convex hull volume, minimum enclosing cylinder, minimum-volume bounding box, minimum-volume enclosing ellipsoid, minimum-width enclosing annulus, and minimum-width cylindrical shell, as well as variants of all these problems for moving points; see [1, 2]. For this reason, ε-kernels

* This work is supported by an NSERC Discovery Grant. This work was started during the author's visit to the Hong Kong University of Science and Technology.


© Timothy M. Chan;
licensed under Creative Commons License CC-BY

have been recognized as one of the most powerful techniques in geometric approximation algorithms.

Using ε-kernels, Agarwal et al. [1] also obtained one-pass *streaming algorithms* for all these problems with $O(\text{polylog}\, n)$ space and $O(\text{polylog}\, n)$ time per insertion of a new point. This was later improved to $O(1)$ space and time for any constant $\varepsilon > 0$ by the author [9] in the real-RAM model (the hidden ε-dependencies were further improved by Zarrabi-Zadeh [16] and by Arya and the author [6]).

Agarwal et al. [1] also used ε-kernels to design *dynamic data structures* for all these problems to support both insertions and deletions in $O(\text{polylog}\, n)$ time, using $O(n)$ space. The update time was later improved to $O(\log n)$ by the author [10] (the ε-dependencies were further improved by Agarwal, Phillips, and Yu [4]).

These results naturally led to the question of whether these fully dynamic data structures can be made to use small space, as in those insertion-only streaming algorithms. The typical model for *dynamic streaming* algorithms in geometry is the *turnstile* model [15]: points are assumed to have integer coordinates in $[U] = \{0, \ldots, U-1\}$ (with $n \leq U^{O(1)}$); the update sequence is given as a stream; during both insertion and deletion of a point s, we are given its d coordinate values. We allow insertions of multiple copies of a point, but a point cannot be deleted more times than it is inserted (otherwise, the algorithm is not guaranteed to produce meaningful output). Several geometric results in this model were known, for example, on k-medians and Euclidean minimum spanning tree weight [12, 11].

However, finding a dynamic streaming algorithm for approximating the width of a point set, even in the two-dimensional case, turned out to be a challenge, and was posed by Agarwal and Indyk in an open problem list.[1] The difficulty is that unlike other problems such as k-medians whose objective function involves *sums*, the width function along a fixed direction involves *maximums/minimums*; for example, it is unclear if sampling techniques help. Furthermore, standard techniques for insertion-only streaming such as "merge-and-reduce" fail in the dynamic setting.

Andoni and Nguyen's method. This open problem was eventually resolved by Andoni and Nguyen [5] in SODA'12, who presented a method with $O(\text{polylog}\, U)$ space and update time for width of a point set in any constant dimension d with approximation factor $1+\varepsilon$ for any constant $\varepsilon > 0$.

The basic idea is remarkably simple: approximate the maximum with a sum of p-th powers, for a small integer $p = O(\log n)$. In order to apply this idea to the width function for all directions simultaneously, we treat the sum as a polynomial function; the sketch then consists of the coefficients of a degree-$O(p)$ polynomial in d variables. However, the number of logarithmic factors is large: the space/time bound is $O(\log^{Cd} U)$ for some unspecified absolute constant C. This constant C comes from the use of a "hammer"—namely, a solver for a system of polynomial inequalities over multiple real variables [8]—needed in the last step to minimize the width function over all possible directions. (Andoni and Nguyen showed how to avoid the polynomial-inequality solver using randomization, but only in the two-dimensional case.)

Though not claimed by Andoni and Nguyen, their dynamic streaming method can likely solve many of the other applications of ε-kernels, such as minimum-volume bounding box. This is because the polynomial maintained by Andoni and Nguyen encodes an approximation of the width along all directions simultaneously, and thus "acts" like an ε-kernel. The

[1] `http://sublinear.info/23`

invocation of the polynomial-inequality solver gets more complex for other problems such as minimum-volume bounding box, but in theory should still work.

Our method. The question of maintaining explicitly an ε-kernel in the dynamic streaming setting still remains. In some applications, it is desirable to find not only the value approximating the optimum, but an actual subset of the point set that realizes the approximate optimum (i.e., a coreset). With deterministic algorithms, this is impossible to do with sublinear space, since an adversary could delete the elements of the coreset and repeat to retrieve the entire point set.

Our main result is a *randomized* (Monte Carlo) dynamic streaming algorithm to maintain an ε-kernel with $O(\text{polylog}\, U)$ space and update time for any constant $\varepsilon > 0$. Thus, we can now obtain coresets for the width, the minimum-volume bounding box, and all the other applications. The key is an oracle to find not just an approximation of the width along a given direction, but approximate extreme points that realize this width. We follow Andoni and Nguyen's idea of approximating the maximum by a sum of p-th powers, but extend it to find a *witness* for an approximate maximum: in Section 2.1, we describe an interesting, simple randomized method to isolate such a witness via sampling. The parameter p is increased by a second logarithmic factor, and $O(\log n)$ polynomials are maintained, but at the end we get fewer logarithmic factors: the space and update bound is about $O(\log^{2d+3} U)$ (see Theorems 4 and 6). The reason for the improvement is that with explicit maintenance of an ε-kernel, we no longer need the polynomial-inequality solver. The avoidance of the "hammer" is also an advantage from the practical perspective.

More broadly speaking, although the "polynomial method" has seen wide-ranging applications in theoretical computer science and has been used in streaming algorithms even before Andoni and Nguyen's paper,[2] its occurrence in computational geometry is rarer (not counting a different "polynomial method" or the use of algebraic geometry in combinatorial geometry, advocated by Sharir and others in recent years). It is therefore worthy of more exposure in the SoCG community, in the author's opinion.

An alternative method. If an ε-kernel need not be reported after every update time, it is desirable to distinguish between update time (time to do an insertion/deletion) and query time (time to report an ε-kernel). For example, although Andoni and Nguyen's method has $O(\log^{Cd} U)$ query time, it has a better update time of about $O(\log^{d+1} U)$. We give an alternative method with an even better update time of about $O(\log^3 U)$, and with a query time of about $O(\log^{3d+4} U)$ (see Theorems 5 and 6).

In this method, we do not even need to store the polynomials explicitly. Instead, the sketch consists of just a few counters, namely, the number of data points (in various samples) that lie at each possible coordinate values modulo q, for each q in a family of small primes. As it turns out, such counters contain as much information as an ε-kernel! (This alternative method is an instance of a *linear sketch* [13].) The Chinese remainder theorem is invoked during the query algorithm.

Although the use of the Chinese remainder theorem is hardly original in the context of streaming algorithms, its occurrence in computational geometry is rarer, and noteworthy.

Outliers. One major application where explicit maintenance of an ε-kernel appears critical is in solving variants of geometric optimization problems that tolerate a certain number

[2] For example, just see Puzzle 1 in Muthukrishnan's survey [15].

k of *outliers*. For example, instead of the width of S, we may want to find the smallest width of $S \setminus T$ over all subsets T of size k. Such variants are well-motivated, since objective functions involving maximums/minimums rather than sums are particularly sensitive to outliers. The new problems can be solved by an extension of ε-kernels [3]: let $w_{a,b}(S, \xi) =$ (a-th largest of $\{s^T\xi\}_{s\in S}$)$-$(b-th smallest of $\{s^T\xi\}_{s\in S}$); a (k, ε)-*kernel* of S is a subset $Q \subset S$ such that $w_{a,b}(Q, \xi) \geq (1-\varepsilon)w_{a,b}(S, \xi)$ for every vector ξ and every $a, b \leq k$.

Agarwal, Har-Peled, and Yu [3] showed that a (k, ε)-kernel of $O((1/\varepsilon)^{(d-1)/2}k)$ size exists and can be constructed in $O(n + k^2)$ time for any constant $\varepsilon > 0$; the running time was improved by the author [10] to $O(n + k \log n)$ (ignoring ε-dependencies). Their algorithm is simple and is by *repeated peeling*: compute a standard ε-kernel, delete its points, and repeat $O(k)$ times.

Andoni and Nguyen's dynamic streaming algorithm cannot be used to solve the problem with k outliers. The key oracle concerns not just the maximum but the *top* k values, but the power-sum approach does not seem immediately applicable, at least without witness finding. With witness finding and repeated deletions though, the task becomes straightforward: since our method explicitly maintains an ε-kernel and supports deletions, Agarwal, Har-Peled, and Yu's peeling algorithm can be directly implemented, and we obtain the first dynamic streaming method for (k, ε)-kernels, with $O(k \operatorname{polylog} U)$ space and update time and $O(k^2 \operatorname{polylog} U)$ query time for any constant $\varepsilon > 0$ (see Corollary 7).

2 An Oracle for Approximate Extreme Points

The key to constructing ε-kernels is an oracle for answering approximate extreme-point queries:

▶ **Problem 1.** *Let $f(s_1, \ldots, s_d, x_1, \ldots, x_{d+1}) = s_1x_1 + \cdots + s_dx_d + x_{d+1}$. Maintain a small-space sketch for a set $S \subset [U]^d$ that supports insertions and deletions in S and can answer queries of the following kind: for any query vector $x \in (\pm[U])^{d+1}$, find an element $\widetilde{m} \in S$ with $|f(\widetilde{m}, x)| \geq (1-\varepsilon) \max_{s\in S} |f(s, x)|$.*

Andoni and Nguyen [5] suggested a simple way to approximate $\max_{s\in S} |f(s, x)|$, namely, by $\left(\sum_{s\in S} f(s, x)^p\right)^{1/p}$ for an even integer p. The approximation factor is at least $(\frac{1}{n})^{1/p}$, which can be made at least $1-\varepsilon$ by setting $p = 2\lceil \log_{1/(1-\varepsilon)} n \rceil = O((1/\varepsilon) \log n)$. The main observation is that $\sum_{s\in S} f(s, x)^p$ is a low-degree polynomial in x; we can maintain the coefficients of this polynomial easily in the dynamic streaming model. However, this solution gives only the value of an approximate maximum, not a "witness" $\widetilde{m} \in S$ that attains such an approximate maximum value.

2.1 Witness for Approximate Maximum

We start with a more basic version of the problem in one dimension:

▶ **Problem 2.** *Maintain a small-space sketch for a set $S \subset [U]$ of at most n elements, where each element s has a nonnegative value y_s, that supports insertions and deletions in S and can answer queries of the following kind: find an element $\widetilde{m} \in S$ with $y_{\widetilde{m}} \geq (1-\varepsilon) \max_{s\in S} y_s$.*

In what follows, we let $m \in S$ denote the element with the true maximum value $y_m = \max_{s\in S} y_s$, which we may assume is nonzero.

There are different ways to solve Problem 2, but not all methods can be generalized to solve Problem 1 later when we replace the y_s's with functions in x (for example, one solution

involves bucketing with exponential spacing, but it is unclear how one would assign functions to buckets). We will propose a solution based purely on maintaining power sums, akin to Andoni and Nguyen's, that can be generalized.

We warm up with a special case of Problem 2 where the maximum is known to have a much larger value than all other elements. Formally, we say that a set is α-*separated* if the ratio of the second largest to the largest in the set is at most α. Consider the following simple algorithm:

Algorithm 1: (preliminary version)
1. let $A = \sum_{s\in S} y_s$ and $C = \sum_{s\in S} s y_s$
2. if $\{y_s\}_{s\in S}$ is α-separated then return $\lfloor C/A \rceil$

Here, $\lfloor x \rceil$ denotes $\lfloor x + 1/2 \rfloor$ (the nearest integer to x). Both sums A and C are easy to maintain dynamically. Intuitively, as α approaches 0, either sum will be dominated by its maximum term and the algorithm would solve the problem *exactly*. An issue remains though: the α-separation condition is not easy to certify dynamically. We will replace it with a condition involving one more sum:

Algorithm 2: (second version)
1. let $A = \sum_{s\in S} y_s$, $B = \sum_{s\in S} y_s^2$, and $C = \sum_{s\in S} s y_s$
2. if $A^2 < (1 + 3n\alpha)B$ then return $\lfloor C/A \rceil$

Intuitively, as α approaches 0, the condition that the largest value is much larger than all other values roughly aligns with the condition that the square of the sum is close to the sum of the squares—this little trick is the key. We formally verify correctness for a concrete choice of α below:

▶ **Lemma 1.** *Let* $\alpha = \frac{1}{3n^2U}$.
(a) *If Algorithm 2 returns a number, then the returned number is exactly* m.
(b) *If* $\{y_s\}_{s\in S}$ *is* α*-separated, then Algorithm 2 returns a number.*

Proof. The proof involves just straightforward algebra. Let $A' = \sum_{s\in S\setminus\{m\}} y_s$. Then

$$\left(\sum_{s\in S} y_s\right)^2 \geq \sum_{s\in S} y_s^2 + 2\sum_{s\in S\setminus\{m\}} y_m y_s \qquad \Longrightarrow \qquad A^2 \geq B + 2y_m A'.$$

If $A^2 < (1 + 3n\alpha)B$, we then have

$$A' < \frac{3n\alpha B}{2y_m} \leq \frac{3n\alpha \cdot n y_m^2}{2y_m} = \frac{1}{2U} y_m$$

and thus

$$\frac{C}{A} \leq \frac{m y_m + U A'}{y_m} < m + \frac{1}{2}$$
$$\frac{C}{A} \geq \frac{m y_m}{y_m + A'} > \frac{m}{1 + \frac{1}{2U}} \geq m\left(1 - \frac{1}{2U}\right) \geq m - \frac{1}{2}.$$

This proves (a). For (b), just note that α-separation implies

$$A^2 \leq (y_m + n\alpha y_m)^2 < (1 + 3n\alpha) y_m^2 \leq (1 + 3n\alpha)B.$$

◀

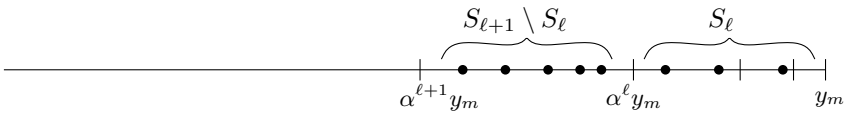


Figure 1 If exactly one element of S_ℓ is chosen and none of $S_{\ell+1} \setminus S_\ell$ is chosen, then the subset is α-separated.

In general, an arbitrary input would not satisfy the α-separation property. The next lemma shows that a *random sample* of a certain size can guarantee separation and at the same time contains an approximate maximum with good probability. The idea of sampling to isolate a unique solution is of course standard [14], but to adapt the idea in the approximate setting, we combine sampling with exponentially spaced bucketing in a nontrivial way—this is arguably the most interesting part of the entire section:

▶ **Lemma 2.** *Let R_j be a random sample of S where each element is selected independently with probability $1/2^j$. Fix any $\alpha > 0$. With probability $\Omega(1)$, for some $j \le \lceil \log n \rceil + 3$ the set $\{y_s\}_{s \in R_j}$ is α-separated and has maximum at least $\alpha^{\lceil \log n \rceil} y_m$.*

Proof. Let $S_\ell = \{s \in S : y_s \ge \alpha^\ell y_m\}$. Pick an index $\ell \le \lceil \log n \rceil$ with $|S_{\ell+1}| \le 2|S_\ell|$. Such an index exists: if not, $|S_{\lceil \log n \rceil}| > 2^{\lceil \log n \rceil} |S_0| \ge n$, which would be a contradiction. Set $j = \lceil \log |S_\ell| \rceil + 3$, so that $2^{j-3} \le |S_\ell| < 2^{j-2}$.

Let E be the event that exactly one element of S_ℓ is chosen in R_j and none of the elements of $S_{\ell+1} \setminus S_\ell$ is chosen in R_j (see Figure 1). If E is true, then $\{y_s\}_{s \in R_j}$ is α-separated, and furthermore has maximum at least $\alpha^\ell y_m$.

Let E_s denote the event that s is chosen in R_j. We can write $E = \bigcup_{s \in S_\ell} (E_s \cap \overline{\bigcup_{s' \in S_{\ell+1} - \{s\}} E_{s'}})$, which holds with probability at least

$$|S_\ell| \cdot \frac{1}{2^j} \cdot \left(1 - \frac{|S_{\ell+1}| - 1}{2^j}\right) > \frac{|S_\ell|}{2^j} \cdot \left(1 - \frac{2|S_\ell|}{2^j}\right) > \frac{1}{16}. \qquad \blacktriangleleft$$

Note that we use exponentially spaced bucketing only in the proof above, not in the algorithm itself.

There is still one complaint: the approximation factor $\alpha^{\lceil \log n \rceil}$ in Lemma 2 appears very weak, especially as the parameter α in Lemma 1 is very small. This can be fixed with one last trick: we can refine the approximation factor by simply replacing each y_s with its p-th power y_s^p for a sufficiently large even integer p. An approximation of $\max_{s \in S} y_s^p$ to within factor $\alpha^{\lceil \log n \rceil}$ yields an approximation of $\max_{s \in S} y_s$ to within factor $\alpha^{\lceil \log n \rceil / p}$, which can be made at least $1 - \varepsilon$ by setting $p = 2 \lceil \log n \rceil \lceil \log_{1/(1-\varepsilon)}(1/\alpha) \rceil$.

Putting everything together, we obtain our final algorithm to solve Problem 2:

Algorithm 3: (final version)

1. for $j = 0$ to $\lceil \log n \rceil + 3$ do
2. let R_j be a random sample of S where each element is selected independently with probability $1/2^j$
3. let $A_j = \sum_{s \in R_j} y_s^p$, $B_j = \sum_{s \in R_j} y_s^{2p}$, and $C_j = \sum_{s \in R_j} s y_s^p$
4. if $A_j^2 < (1 + 3n\alpha) B_j$ and ($\widetilde{m}$ is undefined or $y_{\lfloor C_j / A_j \rfloor} > y_{\widetilde{m}}$) then $\widetilde{m} = \lfloor C_j / A_j \rceil$

▶ **Theorem 3.** *Let $\alpha = \frac{1}{3n^2 U}$ and $p = 2 \lceil \log n \rceil \lceil \log_{1/(1-\varepsilon)}(1/\alpha) \rceil = O((1/\varepsilon) \log^2 U)$. With probability $\Omega(1)$, Algorithm 3 finds an element $\widetilde{m}$ with $y_{\widetilde{m}} \ge (1 - \varepsilon) y_m$. Furthermore, we always have $\widetilde{m} \in S$ whenever $\widetilde{m}$ is not undefined.*

Proof. The theorem follows immediately by combining Lemmas 1 and 2. $\blacktriangleleft$

2.2 First Solution via Polynomials

We are now ready to design a dynamic streaming algorithm to implement the oracle in Problem 1:

▶ **Theorem 4.** *For any constant d, we can solve Problem 1 with*
- $O((1/\varepsilon)^{d+1+o(1)}(\log U)^{2d+3+o(1)}\log(1/\delta))$ *words of space and update/query time,*

with error probability at most δ.

Proof. Our sketch consists of the coefficients of the following polynomials for each $j = 0, \ldots, \lceil \log n \rceil + 3$:

$$A_j(x) = \sum_{s \in R_j} f(s,x)^p, \quad B_j(x) = \sum_{s \in R_j} f(s,x)^{2p}, \quad \text{and} \quad C_j(x) = \sum_{s \in R_j} \lambda(s) f(s,x)^p,$$

where λ is a bijection from $[U]^d$ to $[U^d]$, e.g., $\lambda(s_1, \ldots, s_d) = s_1 U^{d-1} + s_2 U^{d-2} + \cdots + s_d$. Each such polynomial has degree $O(p)$ and consists of $O(p^d)$ monomials (since there are $O(p)$ choices for the degree of each of the $d+1$ variables, and the sum of these degrees is fixed). Each of the $O(p^d)$ coefficients of each of these $O(\log n)$ polynomials requires $O(p \log U)$ bits; the total space usage in words is $O(p^{d+1} \log n)$, where $p = O((1/\varepsilon)\log^2 U)$ (which remains true even after replacing U with U^d).

Insertion/deletion of an element s requires adding/subtracting a term to/from these polynomials. Each of the $O(p^d)$ coefficients of each of the $O(\log n)$ polynomials can be updated using $O(1)$ arithmetic operations on $O(p \log U)$-bit numbers; the total time is $O(p^{d+1+o(1)} \log n)$ in the $(\log U)$-bit word RAM model.

A query for a given vector x can be answered by evaluating these polynomials at x, via Algorithm 3 and Theorem 3. Each of the $O(\log n)$ evaluations requires $O(p^d)$ arithmetic operations on $O(p \log U)$-bit numbers; the total time is $O(p^{d+1+o(1)} \log n)$.

One technical issue concerns how the samples R_j are generated and stored. We can pick a random hash function $h : [N] \to [N]$ from a pairwise independent family[3] for some $N \in [U^d, 2U^d]$, and set $R_j = \{s \in S : h(\lambda(s)) \le \lfloor N/2^j \rfloor\}$. The proof of Lemma 2 only requires pairwise independence of the events E_s. Such a hash function h can be encoded in $O(\log U)$ bits and evaluated in constant time.

To lower the error probability from constant to δ, we can use $O(\log(1/\delta))$ independent versions of the data structure, increasing all bounds by an $O(\log(1/\delta))$ factor. ◀

2.3 Alternative Solution via Chinese Remainders

We also propose an alternative dynamic streaming algorithm for Problem 1 which does not require explicitly storing the polynomials $A_j(x), B_j(x), C_j(x)$ but instead uses the Chinese remainder theorem. The new version has better update time (which is desirable when there are more updates than queries).

▶ **Theorem 5.** *For any constant d, we can solve Problem 1 with*
- $O((1/\varepsilon)^{d+1}(\log U)^{3d+4}\log(1/\delta)/\log\log U)$ *words of space and query time, and*
- $O((1/\varepsilon)(\log U)^3 \log(1/\delta)/\log\log U)$ *expected update time,*

with error probability at most δ.

[3] For example, $h(i) = (r_1 + r_2 i) \bmod N$ for two random numbers $r_1, r_2 \in [N]$, assuming that N is prime.

Proof. Let $W = \max\{n(dU^2+U)^{2p}, nU^d(dU^2+U)^p\}$. Find a set Π of $O(\log W/\log\log W)$ primes, each at most $O(\log W)$, whose product exceeds $2W$. Our sketch simply consists of a counter

$$n_{j,q,(\sigma_1,\ldots,\sigma_d)} \;=\; |\{(s_1,\ldots,s_{d_0}) \in R_j : s_1 \equiv \sigma_1,\ \ldots,\ s_d \equiv \sigma_d \pmod q\}|$$

for each $j = 0,\ldots,\lceil\log n\rceil+3$, each $q \in \Pi$, and each $(\sigma_1,\ldots,\sigma_d) \in [q]^d$; the total space usage in words is $O(\log n \cdot (\log W/\log\log W)\cdot \log^d W)$, where $\log W = O(p\log U) = O((1/\varepsilon)\log^3 U)$.

Insertion/deletion of a point s affects an expected $O(1)$ number of samples R_j, each requiring increments/decrements of $O(\log W/\log\log W)$ counters of $O(\log n)$ bits and $O(\log W/\log\log W)$ divisions on $O(\log U)$-bit numbers; the total expected time is $O(\log W/\log\log W)$ in the $(\log U)$-bit word RAM model.

A query for a given point x can be answered by computing $A_j(x), B_j(x), C_j(x)$ as in the proof of Theorem 4, for each $j = 0,\ldots,\lceil\log n\rceil+3$. To this end, we compute, for each $q \in \Pi$,

$$A_j(x) \equiv \sum_{\sigma\in[q]^d} n_{j,q,\sigma} f(\sigma,x)^p \pmod q.$$

From these values, we can then reconstruct $A_j(x)$ by the Chinese remainder theorem, since $|A_j(x)| < W$. Similarly, we can compute $B_j(x)$ and $C_j(x)$. For each of the $O(\log n)$ indices j and each of the $O(\log W/\log\log W)$ primes q, the computation requires $O(\log^d W)$ evaluations of f and $O(\log^d W)$ arithmetic operations on $O(\log U)$-bit numbers; the total time is $O(\log n \cdot (\log W/\log\log W)\cdot \log^d W)$.

The technical issue of storing the samples R_j can be addressed as before. The error probability can be lowered to δ by increasing the bounds by an $O(\log(1/\delta))$ factor as before. ◀

3 Applications

3.1 ε-Kernels

We now apply the oracle from the previous section to compute an ε-kernel for a point set $S \subset [U]^d$ in a constant dimension d. This part is where geometry finally comes into play, although as it turns out, not much originality is required here. We adopt the following variant of one of Agarwal, Har-Peled, and Varadarajan's ε-kernel algorithms [1]:

0. $s_0 =$ any point in S
1. for $i = 1$ to d do
2. $\quad s_i =$ a point $s \in S$ that approximately maximizes the distance to the $(i-1)$-flat through $s_0,\ldots,s_{i-1}$, with constant approximation factor $1/c$
3. $\phi =$ the affine transformation that maps $s_0, s_1, \ldots, s_d$ to $e_0 = (0,\ldots,0), e_1 = (1,0,\ldots,0), \ldots, e_d = (0,\ldots,0,1)$
4. for each grid point ξ of a grid over $\partial[-1,1]^d$ of side length ε do
5. $\quad s_\xi =$ a point $s \in S$ that approximately maximizes $\phi(s)\cdot\xi$ with additive error $O(\varepsilon)$
6. return an ε-kernel of $O((1/\varepsilon)^{(d-1)/2})$ size for all these s_ξ's

Correctness. Because the algorithm is not original, we will only outline the correctness proof and point to references for the details. The algorithm consists of two phases. In the first phase (lines 0–3), we compute a transformation ϕ that makes $\phi(S)$ *fat*, by a simple iterative procedure with d steps (based on an idea of Barequet and Har-Peled [7]). Specifically, $\phi(S)$ satisfies the following properties:

- $\phi(S) \subset [-c, c]^d$. A proof can be found, for example, in [10, Lemma 2.2].
- $w(\phi(S), \xi) \geq 1/\sqrt{d}$ for all unit vectors ξ. This follows since $\phi(S)$ contains $\{e_0, \ldots, e_d\}$ by line 3.

In the second phase (lines 4–6), we compute an $O(\varepsilon)$-kernel of this fat point set $\phi(S)$, simply by finding extreme points along $O((1/\varepsilon)^{d-1})$ roughly equally spaced directions. A correctness proof can be found, for example, in [9, Theorem 2.5].

As noted in previous papers [1], an $O(\varepsilon)$-kernel of $\phi(S)$ yields an $O(\varepsilon)$-kernel of S for any invertible affine transformation ϕ, and an ε-kernel of an $O(\varepsilon)$-kernel is an $O(\varepsilon)$-kernel, and so the final output is an $O(\varepsilon)$-kernel of S.

Analysis. A naive implementation of lines 2 and 5 would require linear time. To achieve sublinear query time, we observe how both lines can be implemented using the extreme-point oracle for Problem 1.

For line 2, let A_{i-1} be the $d \times (i-1)$ matrix with column vectors $s_1 - s_0, \ldots, s_{i-1} - s_0$. Define the projection matrix $P_{i-1} = A_{i-1}(A_{i-1}^T A_{i-1})^{-1} A_{i-1}^T$ and let $P'_{i-1} = I - P_{i-1}$. The subproblem is equivalent to finding an $s \in S$ that approximately maximizes the expression $\|P'_{i-1}(s - s_0)\|$. It suffices to approximately maximize the expression $|P'_{i-1}(s - s_0) \cdot e_j|$ for each $j = 1, \ldots, d$ and take the maximum of the results; the approximation worsens by a $\sqrt{d}$ factor. Since

$$P'_{i-1}(s - s_0) \cdot e_j = s^T (P'_{i-1})^T e_j - s_0^T (P'_{i-1})^T e_j,$$

we can consult the oracle for the query vector $x = ((P'_{i-1})^T e_j, -s_0^T (P'_{i-1})^T e_j) \in \mathbb{R}^{d+1}$ (using a constant ε).

One issue is that this choice of x involves rational rather than integer coordinates, but we can convert the coordinates to integers by multiplying by the lowest common denominator. It can be checked that all integers involved are bounded by $U^{O(d)}$ (for example, by using the standard formula for matrix inverse in terms of determinants and co-factors); replacing U with $U^{O(d)}$ will not affect the space/time bounds.

For line 5, first note that $\phi(s) = A_d^{-1}(s - s_0)$. Since fatness implies that $|\phi(s) \cdot \xi| \leq dc$, it suffices to maximize $\phi(s) \cdot \xi + dc$ (which is nonnegative) with approximation factor $1 - \varepsilon$. Since

$$\phi(s) \cdot \xi + dc = s^T (A_d^{-1})^T \xi - s_0^T (A_d^{-1})^T \xi + dc,$$

we can consult the oracle for the query vector $x = ((A_d^{-1})^T \xi, -s_0^T (A_d^{-1})^T \xi + dc) \in \mathbb{R}^{d+1}$. Again, we can make the coordinates of x integers, bounded by $U^{O(d)}$.

Line 0 is easy (we can just query the oracle with an arbitrary x). Line 6 can be done by invoking a standard ε-kernel algorithm [1, 9] on $O((1/\varepsilon)^{d-1})$ points.

The total query time is thus dominated by the $O((1/\varepsilon)^{d-1})$ oracle calls in line 5. The error probability δ needs to be adjusted by an $O((1/\varepsilon)^{d-1})$ factor. Note that in Theorem 4 or 5, the query vectors should be independent of the random choices made by the data structure. This can be ensured by creating $d+1$ independent versions of the data structure, and using the i-th version for the i-th iteration in line 2, and the $(d+1)$-th version for line 5.

▶ **Theorem 6.** *For any constant dimension d, we can maintain an ε-kernel with*

- $O((1/\varepsilon)^{d+1+o(1)} (\log U)^{2d+3+o(1)} \log(1/\delta\varepsilon))$ *words of space and update time, and*
- $O((1/\varepsilon)^{2d+1+o(1)} (\log U)^{2d+3+o(1)} \log(1/\delta\varepsilon))$ *query time,*

or alternatively,

- $O((1/\varepsilon)^{d+1}(\log U)^{3d+4}\log(1/\delta\varepsilon)/\log\log U)$ *words of space,*
- $O((1/\varepsilon)(\log U)^3\log(1/\delta\varepsilon)/\log\log U)$ *expected update time, and*
- $O((1/\varepsilon)^{2d}(\log U)^{3d+4}\log(1/\delta\varepsilon)/\log\log U)$ *query time,*

with error probability at most δ.

3.2 ... with Outliers

For the variant of ε-kernels with k outliers, we invoke the peeling algorithm by Agarwal, Har-Peled, and Yu [3]: we compute an ε-kernel of $O((1/\varepsilon)^{(d-1)/2})$ size, delete its points, and repeat for $2k+2$ iterations; we output all the $O((1/\varepsilon)^{(d-1)/2}k)$ points deleted and reinsert them back.

Note that in Theorem 4 or 5, the update sequence for the data structure should be independent of the random choices made by the data structure. This can be ensured by creating $2k+2$ independent versions of the data structure, and using the i-th version for the i-th iteration. The space and update time bounds are increased by an $O(k)$ factor. The query time is $O(k)$ times the query time bound of our ε-kernel data structure, plus $O((1/\varepsilon)^{(d-1)/2}k)$ times the update time bound. The error probability needs to be adjusted by another $O(k)$ factor.

▶ **Corollary 7.** *For any constant dimension d, we can maintain a (k,ε)-kernel with*

- $O((1/\varepsilon)^{d+1+o(1)}k(\log U)^{2d+3+o(1)}\log(k/\delta\varepsilon))$ *words of space and update time, and*
- $O((1/\varepsilon)^{2d+o(1)}k(\log U)^{2d+3+o(1)}\log(k/\delta\varepsilon)+(1/\varepsilon)^{(3d+1)/2}k^2(\log U)^{2d+3}\log(k/\delta\varepsilon))$ *query time,*

or alternatively,

- $O((1/\varepsilon)^{d+1}k(\log U)^{3d+4}\log(k/\delta\varepsilon)/\log\log U)$ *words of space,*
- $O((1/\varepsilon)k(\log U)^3\log(k/\delta\varepsilon)/\log\log U)$ *expected update time, and*
- $O((1/\varepsilon)^{2d}k(\log U)^{3d+4}\log(k/\delta\varepsilon)/\log\log U+(1/\varepsilon)^{(d+1)/2}k^2(\log U)^3\log(k/\delta\varepsilon)/\log\log U)$ *query time,*

with error probability at most δ.

Acknowledgement. I thank Ke Yi for discussions that led to this work.

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

2 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In Emo Welzl, editor, *Current Trends in Combinatorial and Computational Geometry*, pages 1–30. Cambridge University Press, 2005.

3 Pankaj K. Agarwal, Sariel Har-Peled, and Hai Yu. Robust shape fitting via peeling and grating coresets. *Discrete & Computational Geometry*, 39(1-3):38–58, 2008. doi:10.1007/s00454-007-9013-2.

4 Pankaj K. Agarwal, Jeff M. Phillips, and Hai Yu. Stability of ε-kernels. In *Proceedings of the 18th Annual European Symposium on Algorithms, Part I*, pages 487–499, 2010. doi:10.1007/978-3-642-15775-2_42.

5 Alexandr Andoni and Huy L. Nguyen. Width of points in the streaming model. In *Proceedings of the 23rd Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 447–452, 2012. *ACM Trans. Algorithms*, to appear.

6 Sunil Arya and Timothy M. Chan. Better ε-dependencies for offline approximate nearest neighbor search, Euclidean minimum spanning trees, and ε-kernels. In *Proceedings of the 30th Annual Symposium on Computational Geometry*, pages 416–425, 2014. doi:10.1145/2582112.2582161.

7 Gill Barequet and Sariel Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38(1):91–109, 2001.

8 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996. doi:10.1145/235809.235813.

9 Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom.*, 35(1-2):20–35, 2006. doi:10.1016/j.comgeo.2005.10.002.

10 Timothy M. Chan. Dynamic coresets. *Discrete & Computational Geometry*, 42(3):469–488, 2009. doi:10.1007/s00454-009-9165-3.

11 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. *Int. J. Comput. Geometry Appl.*, 18(1/2):3–28, 2008. doi:10.1142/S0218195908002520.

12 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 373–380, 2004. doi:10.1145/1007352.1007413.

13 Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 174–183, 2014. doi:10.1145/2591796.2591812.

14 R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

15 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

16 Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011. doi:10.1007/s00453-010-9392-2.

Two Approaches to Building Time-Windowed Geometric Data Structures*

Timothy M. Chan[1] and Simon Pratt[2]

1 **Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada**
tmchan@uwaterloo.ca

2 **Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada**
s2pratt@uwaterloo.ca

Abstract

Given a set of geometric objects each associated with a time value, we wish to determine whether a given property is true for a subset of those objects whose time values fall within a query time window. We call such problems *time-windowed decision problems*, and they have been the subject of much recent attention, for instance studied by Bokal, Cabello, and Eppstein [SoCG 2015]. In this paper, we present new approaches to this class of problems that are conceptually simpler than Bokal *et al.*'s, and also lead to faster algorithms. For instance, we present algorithms for preprocessing for the time-windowed 2D diameter decision problem in $O(n \log n)$ time and the time-windowed 2D convex hull area decision problem in $O(n\alpha(n) \log n)$ time (where α is the inverse Ackermann function), improving Bokal *et al.*'s $O(n \log^2 n)$ and $O(n \log n \log\log n)$ solutions respectively.

Our first approach is to reduce time-windowed decision problems to a generalized range successor problem, which we solve using a novel way to search range trees. Our other approach is to use dynamic data structures directly, taking advantage of a new observation that the total number of combinatorial changes to a planar convex hull is near linear for any *FIFO* update sequence, in which deletions occur in the same order as insertions. We also apply these approaches to obtain the first $O(n \operatorname{polylog} n)$ algorithms for the time-windowed 3D diameter decision and 2D orthogonal segment intersection detection problems.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases time window, geometric data structures, range searching, dynamic convex hull

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.28

1 Introduction

Time-windowed geometric problems have been the subject of many recent papers and are motivated by timestamped social network data and *Geographic Information System* (GIS) data, the latter of which may consist not only of longitude, latitude, and altitude coordinates but also time. A 2014 paper by Bannister *et al.* [4] examined time-windowed versions of convex hull, approximate spherical range searching, and approximate nearest neighbor queries.

* Research supported by The Natural Sciences and Engineering Research Council of Canada and the Ontario Graduate Scholarship.

© Timothy M. Chan and Simon Pratt;
licensed under Creative Commons License CC-BY

At SoCG 2015, Bokal *et al.* [5] presented more results on a variety of other time-windowed problems. In the same year, Chan and Pratt [12] studied the time-windowed closest pair problem.

Let S be a set of n objects, where each object $s \in S$ is associated with a time value $t(s)$. In this paper, we consider problems for which the answer is a Boolean value: given a query interval of time $[t_1, t_2]$ called a *time window*, does the subset of S whose time values are within the query window have property $\mathcal{P}$ or not? We call these *time-windowed decision problems*. For brevity, we say objects whose time values are within the query window are themselves within the query window.

Without loss of generality, we assume that time values are given as integers from 1 to n, for otherwise we can replace time values with their rank during preprocessing. The query time only increases by $O(1)$ predecessor searches on the query time values.

In this and the previous paper [5], we focus only on hereditary properties, meaning if a set S has $\mathcal{P}$ then any superset $S' \supseteq S$ also has it.[1] Examples of hereditary properties include: the set of points has greater than unit diameter, or the convex hull of a set of points has greater than unit area.

As observed by Bokal *et al.*, it suffices to find for each start time t the maximal end-time t' such that all objects within the window $[t, t']$ have $\mathcal{P}$. Afterwards, we can easily obtain a data structure with $O(n)$ words of space and $O(1)$ query time.[2] We do so by storing the resulting t' in a table indexed by start time t (recall that time values have been initially reduced to integers from 1 to n). A query for a time window $[t_1, t_2]$ is answered by looking up the t' in the table for start time t_1, and checking if $t_2 \leq t'$.

For this reason, Bokal *et al.* refer to time-windowed decision problems as the problem of finding maximal contiguous subsequences with hereditary properties.

Since answering a query after preprocessing is trivial, for the rest of the paper we focus only on bounding the preprocessing time.

1.1 Previous results

Recently, Bokal *et al.* [5] presented an approach to time-windowed decision problems. They achieve the following geometric results:

1. *2D diameter decision*: Given a set of n time-labeled points in $\mathbb{R}^2$, determine if there exist two points greater than unit distance apart, whose time values are within a query time window. Their approach obtains $O\big(n \log^2 n\big)$ preprocessing time.
2. *2D convex hull area decision*: Given a set of n time-labeled points in $\mathbb{R}^2$, determine whether the convex hull of points within a query time window has greater than unit area. Their approach obtains $O(n \log n \log \log n)$ preprocessing time.
3. *2D monotone paths*: Given a set of n points in $\mathbb{R}^2$, determine if the points within a query time window form a monotone path in some (subpath-dependent) direction. Their approach obtains $O(n)$ preprocessing time.

They also show that their approach works for graph planarity. Given a graph whose edge set contains n time-labeled edges, determine if the subgraph on the edges within a query time window is planar. Their approach obtains $O(n \log n)$ preprocessing time.

---

[1] The definition in [5] considers subsets instead of supersets, but is equivalent after complementation.

[2] In fact, Chan and Pratt [12] show that we can reduce space to $O(n)$ bits while maintaining $O(1)$ query time, by using succinct rank/select data structures [23].

Chan and Pratt [12] studied the time-windowed closest pair decision problem, in which we are given a set of n time-labeled points in $\mathbb{R}^d$ and we wish to determine whether there exist two points at most unit distance apart. They solve the problem in $O(n)$ time using grids. They also consider the exact version of the problem, which is to find the closest pair of points within the time window. They solve this problem in $O(n \log n \log\log n)$ preprocessing time and $O(\log\log n)$ query time with $O(n \log n)$ words of space. Their techniques relied on geometric properties of the closest pair such that they could not be trivially modified to solve the time-windowed diameter problem.

1.2 New results

We achieve the following results:

1. *2D and 3D diameter decision*: We improve Bokal *et al.*'s preprocessing time bound in 2D from $O\big(n \log^2 n\big)$ to $O(n \log n)$. Thus, we obtain the first optimal algorithm for the problem in the algebraic decision-tree model [26]. Furthermore, we obtain the first nontrivial result in 3D with $O\big(n \log^2 n\big)$ preprocessing time. See Section 2 for details.
2. *2D orthogonal segment intersection detection*: Given a set of n orthogonal (horizontal or vertical) time-labeled line segments in $\mathbb{R}^2$, we want to determine if there are any intersections between segments whose time values are within a query time window. We give the first nontrivial result for this problem, obtaining $O(n \log n \log\log n)$ preprocessing time. See Section 2 for details.
3. *2D convex hull area decision*: We improve Bokal *et al.*'s preprocessing time bound from $O(n \log n \log\log n)$ to $O(n\alpha(n) \log n)$ (where α is the inverse Ackermann function). See Section 3 for details.
4. *2D width decision*: Given a set of n time-labeled points in $\mathbb{R}^2$, we want to determine whether the points within a query time window have greater than unit width. We give the first nontrivial result for this problem, obtaining $O\big(n \log^8 n\big)$ preprocessing time. See Section 3 for details. Previously, a naive approach using Chan's dynamic data structure [8] would give a worse $O\big(n^{3/2} \operatorname{polylog} n\big)$ time bound.

1.3 Techniques

Bokal *et al.*'s main approach computes the upper triangle of a binary $n \times n$ matrix where entry i, j has value 1 if and only if the set of objects within the window $[i, j]$ has $\mathcal{P}$. The first step is to greedily decompose the upper triangle into disjoint rectangles along the diagonal. The second step is to compute the values within each rectangle. First they compute the maximal end-time for the row which divides height of the rectangle in half. This splits the rectangle into 4 sub-rectangles, above and below the median row and left and right of its maximal end-time. The entries in the top-right have value 0, the entries in the bottom-left have value 1, and they recurse on top-left and bottom-right. Efficiency comes from the using a bounded-size *sketch* of uncomputed regions during recursion. A sketch is similar to a coreset in that it approximates a subset of objects, and its exact nature depends on the problem. Their results for both the 2D diameter and the 2D convex area decision problem are obtained via this approach.

Our first approach, which we discuss in Section 2, is much more direct: we simply reduce the problem to a *range successor search* problem. This (static) data structure problem can be solved by standard range searching techniques, and if we are not too concerned with extra logarithmic factors, we immediately obtain efficient solutions to the diameter decision

problem in 2D *and* 3D, as well as orthogonal line segment intersection detection. To further improve the logarithmic factors, we come up with a more clever way to traverse a range tree.

Our second approach, which we discuss in Section 3, simply adds the objects in sequence to a dynamic data structure until $\mathcal{P}$ is true for the objects in the structure, then removes from the beginning until $\mathcal{P}$ is not true, then repeats. Bokal *et al.* [5] have already suggested this naive use of dynamic data structures as a natural solution to the problem, but dismissed it as inefficient. We show that it is actually efficient in the case of the 2D convex hull area and width decision problems! We do so by a new bound on the combinatorial complexity of the structural changes to the convex hull under a certain sequence of updates in which the order of insertions is the same as the order of deletions. We call these *FIFO* update sequences. With this combinatorial bound (which may be of independent interest) it is straightforward to solve the time-windowed 2D width decision problem in $O(n \operatorname{polylog} n)$ time using Eppstein's dynamic width data structure [18], and with a more careful analysis, we can solve time-windowed 2D convex hull area decision problem in $O(n\alpha(n) \log n)$ time using hull trees [24].

2 Generalized range successor approach

In this section, we focus on time-windowed decision problems for properties that deal with pairs. More precisely, given a symmetric relation $\mathcal{R} = S \times S$, we consider the property $\mathcal{P}$ that there exist $p, q \in S$ such that $(p, q) \in \mathcal{R}$. We call such properties *pairwise interaction properties*. If $(p, q) \in \mathcal{R}$, we say that p *interacts with* q; we also say that p is *in* q*'s range*.

Examples of such problems include: diameter decision, for which two points interact if they are further apart than unit distance; segment intersection detection, in which two segments interact with each other if they intersect; and closest pair decision, in which two points interact if they are nearer than unit distance. Note that such properties are *hereditary*.

Our approach is to reduce the time-windowed problem to the following data structure problem:

▶ **Definition 1.** Let each object $s \in S$ have weight $w(s)$. In the *generalized range successor problem*, we want to preprocess S to find the *successor* of a query object q among the objects in q's range, that is, the object $p \in S$ that interacts with q with the smallest $w(p) > w(q)$.

The above is a generalization of the original 1D range successor problem where the objects are points in 1D and the objects' ranges are intervals; for example, see [30] for the latest results.

To see how the above data structure problem can be used to solve the time-windowed pairwise interaction problem, we simply find the successor p_q of every $q \in S$ in the generalized range successor problem with weights equal to time values.

▶ **Observation 2.** *A query window* $[t_1, t_2]$ *contains an interacting pair if and only if* $[t_1, t_2]$ *contains* $[t(q), t(p_q)]$ *for some* $q \in S$.

Proof. The "if" direction is trivial. For the "only if" direction, let p, q be an interacting pair in $[t_1, t_2]$ and assume that $t(q) \leq t(p)$ without loss of generality. Then $t(q) \leq t(p_q) \leq t(p)$ by definition of the successor p_q, and the claim follows. ◀

Thus, the answer to a query window $[t_1, t_2]$ is yes if and only if the point $(t_1, -t_2)$ is dominated by some point $(t(p), -t(p_q))$. The time-windowed problem can then be solved by precomputing the *maxima* [26] of the 2D point set $\{(t(p), -t(p_q)) \mid q \in S\}$, which takes linear time by a standard plane sweep after pre-sorting (recall that time values have been

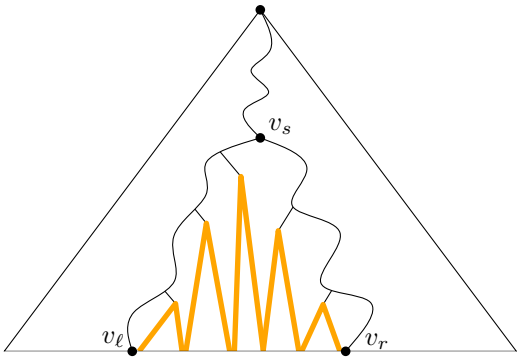


Figure 1 A range tree showing the path followed by a query for an interval $[\ell, r]$ which splits at v_s, and the subtrees in **bold** which fall within the query range between leaves v_ℓ and v_r.

initially reduced to integers in $\{1, \ldots, n\}$ and can be trivially sorted in linear time). The running time is then dominated by the cost of computing the successors p_q for all $q \in S$. If we can solve the generalized range successor problem in $P(n)$ preprocessing time and $Q(n)$ query time, we can solve the corresponding time-windowed problem in $O(P(n) + nQ(n))$ preprocessing time.

In the rest of this section, we can thus focus on solving the generalized range successor problem.

One approach is to first consider the decision version of the problem: deciding whether there exists an object p that lies in q's range and has weight $w(p)$ in the interval $[\ell, r]$ for $\ell = w(q)$ and a given value r. This problem can be solved using standard multi-level data structuring techniques: The primary structure is a 1D range tree [26] on the weights (i.e., time values). See Figure 1. This naturally decomposes any interval $[\ell, r]$ of weights into $O(\log n)$ canonical subtrees by performing a binary search for both ℓ and r until we reach their lowest common ancestor node v_s at which the search splits. The search continues leftward and rightward to leaves v_ℓ and v_r with values ℓ and r respectively. Every right subtree on the path from v_s to v_ℓ, and every left subtree on the path from v_s to v_r are within the interval $[\ell, r]$. At each node v, we store the subset S_v of all objects within its interval in a *secondary structure* for the original range searching problem—deciding whether there exists an object in S_v that lies in a query object q's range. A query for the decision problem can then be answered by making $O(\log n)$ queries to the secondary structures. This increases the query time by a logarithmic factor.

Finally, we can reduce the generalized range successor problem to its decision problem by a binary search over all time values r. This increases the query time by a second logarithmic factor.

2.1 Avoiding binary search

In this subsection, we describe a still better algorithm that solves the generalized range successor problem without going through the decision problem, thereby removing one of the extra logarithmic factors caused by the binary search.

We first find the leaf node v storing q in $O(\log n)$ time. To answer a successor query for q, we proceed in two phases. (See Figure 2.)

- In the first (i.e., "up") phase, we walk upward from v towards the root. Each time our search follows a parent pointer from a left child, we query the secondary structure at the

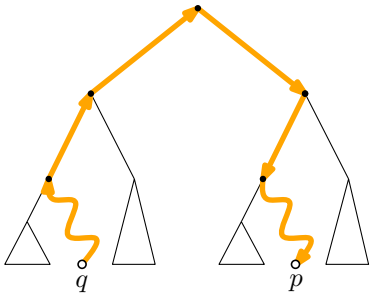


Figure 2 A search path finding the successor p of a point q is shown in **bold**. The search is divided into an "up" phase followed by a "down" phase.

right child to see if there exists an object stored at the right child that is in q's range. If no, we continue upward. If yes, the answer is in the subtree at the right child and we proceed to the second phase starting at this node.

- In the second (i.e., "down") phase, we walk downward from the current node to a leaf. Each time our search descends from a node, we query the secondary structure at the left child to see if there exists an object stored at the left child that is in q's range. If no, the answer is in the right subtree and we descend right. Otherwise, we descend left.

This algorithm makes $O(\log n)$ queries in the secondary structures. We next apply this algorithm to specific time-windowed pairwise interaction problems.

2.2 2D diameter decision

For the application to 2D diameter decision, our set of objects S is composed of points in $\mathbb{R}^2$, and p, q interact if and only if $d(p, q) > 1$. In other words, q's range is the complement of a unit disk.

The secondary structure at a node v needs to handle the following type of query: decide whether a query point q has greater than unit distance from some point in S_v, that is, decide whether q lies outside the intersection D_v of all unit disks centered at the points of S_v (see Figure 3). We store the unit-disk intersection D_v, along with the sorted list of the x-coordinates of the vertices of D_v.

Since we can merge two unit-disk intersections in linear time (similar to how we can merge two planar convex hulls in linear time), we can build the secondary structures at all nodes of the range tree bottom-up in $P(n) = O(n \log n)$ time.

Given point q, a query in the secondary structure at node v reduces to binary search for the x-coordinate of q in the list X_v and takes $O(\log n)$ time. Since the algorithm in Section 2.1 requires $O(\log n)$ queries in the secondary structures, the overall query time is $Q(n) = O\big(\log^2 n\big)$.

We can use *fractional cascading* to speed up the algorithm further [15, 16]. Since the technique is well known, we give just a quick sketch. Recall that the algorithm in Section 2.1 is divided into two phases.

- For the "up" phase, we first move the list X_v of each right child v to its parent. We pass a fraction of the elements of the list at each node to the lists at both children during preprocessing. This way, we can determine where the x-coordinate of q is in the list of the parent from where it is in the list of the child in $O(1)$ time. We can then answer all $O(\log n)$ queries in the secondary structures during the "up" phase in $O(\log n)$ overall time, after an initial binary search at the leaf in $O(\log n)$ time.

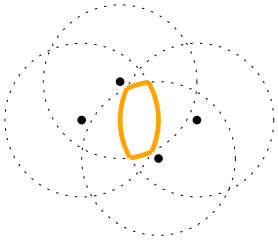


Figure 3 The boundaries of unit disks centered at four points in $\mathbb{R}^2$. The boundary of the unit-disk intersection is shown in **bold**.

- For the "down" phase, we pass a fraction of the elements of the list at each node to the list at its parent during preprocessing. This way, we can determine where the x-coordinate of q is in the list of a child from where it is in the list of its parent in $O(1)$ time. We can then answer all $O(\log n)$ queries in the secondary structures during the "down" phase in $O(\log n)$ overall time, after an initial binary search in $O(\log n)$ time.

We conclude that a generalized range successor query in this setting can be answered in $Q(n) = O(\log n)$ time. This gives us the following result.

▶ **Theorem 3.** *We can preprocess for the time-windowed 2D diameter decision problem in $O(n \log n)$ time.*

2.3 3D diameter decision

In 3D, a query in the secondary structure at node v becomes deciding whether the query point q lies outside the intersection D_v of unit balls centered at the points of S_v. In 3D, the unit-ball intersection D_v still has linear combinatorial complexity and can be constructed in $O(|S_v| \log |S_v|)$ time by Clarkson and Shor's randomized algorithm [17] or Amato *et al.*'s deterministic algorithm [3]. We store the xy-projection of the upper and lower boundary of D_v in a planar point location structure [26]. We can build the secondary structures at all nodes of the range tree in $O(n \log n)$ time per level, and thus $P(n) = O\big(n \log^2 n\big)$ total time. (Unlike in 2D, it is not clear if we could speed up the building time by linear-time merging.)

Given point q, a query in a secondary structure reduces to planar point location for the xy-projection of q and takes $O(\log n)$ time [26]. Since the algorithm in Section 2.1 requires $O(\log n)$ queries in the secondary structures, the overall query time is $Q(n) = O(\log^2 n)$. (Unlike in 2D, we cannot apply fractional cascading to speed up the algorithm.) This gives us the following result.

▶ **Theorem 4.** *We can preprocess for the time-windowed 3D diameter decision problem in $O\big(n \log^2 n\big)$ time.*

2.4 Orthogonal segment intersection detection

For the application to 2D orthogonal segment intersection detection, our set of objects S is composed of vertical and horizontal line segments in $\mathbb{R}^2$, and p, q interact if and only if they intersect.

Without loss of generality, we assume that all x- and y-coordinates are given as integers from 1 to $O(n)$, for otherwise we can replace coordinate values with their rank during preprocessing. The query time only increases by $O(1)$ predecessor searches on the coordinate values, costing no more than $O(\log n)$ time.

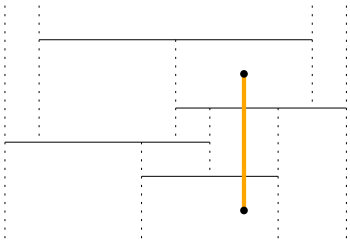


Figure 4 A set of horizontal segments is shown in solid lines, with their vertical decomposition shown in dotted lines. A query vertical segment is shown in **bold**, whose endpoints are in different cells.

The secondary structure at a node v now needs to handle the following type of query: decide whether a query segment q intersects some segment in S_v. Without loss of generality, assume that q is vertical and the segments in S_v are horizontal. We store the vertical decomposition VD_v (also called the trapezoidal decomposition) in a planar point location structure.

Since we can compute the vertical decomposition VD_v in $O(|S_v| \log\log |S_v|)$ time by a standard plane sweep with van Emde Boas trees, we can build the secondary structures at all nodes of the range tree in $P(n) = O(n \log n \log\log n)$ time.

Given vertical segment q, a query in the secondary structure at node v requires testing whether both endpoints of q lie in the same cell in VD_v, which reduces to two planar point location queries. Since the subdivision is orthogonal, we can apply Chan's orthogonal point location structure [10], which achieves $O(\log\log U)$ query time when coordinates are integers from $\{1, \ldots, U\}$—recall that coordinate values have been initially reduced to integers bounded by $U = O(n)$. Since the algorithm in Section 2.1 requires $O(\log n)$ queries in the secondary structures, the overall query time is $Q(n) = O(\log n \log\log n)$. This gives us the following result.

▶ **Theorem 5.** *We can preprocess for the time-windowed 2D orthogonal intersection detection problem in $O(n \log n \log\log n)$ time.*

▶ Remark. An open problem is to remove the extra $\log\log n$ factor. Perhaps the techniques from [11] for the 4D offline dominance searching problem may be relevant.

3 FIFO update sequence approach

In the previous section, we have presented an approach to building data structures to solve time-windowed pairwise interaction problems, but the 2D width and the 2D convex hull area decision problems, for instance, cannot be expressed in terms of a pairwise interaction property.

As mentioned in the introduction, both problems are on hereditary properties. The most obvious approach to solve a problem on a hereditary property is to use a dynamic data structure directly, inserting each object in order until $\mathcal{P}$ is satisfied, then deleting each object in the same order until $\mathcal{P}$ is no longer satisfied, and repeating. By storing for each $i \in \{1, \ldots n\}$ the largest j for which $\{s_i, \ldots, s_j\}$ satisfies $\mathcal{P}$, we can answer queries for the time-windowed problem. However, this approach does not seem to yield efficient solutions in some settings. For example, for the 2D width decision problem, we would need a fully dynamic data structure for 2D width decision, but the best result to date has near $\sqrt{n}$ update time [8]. Agarwal and Sharir [2] gave a dynamic data structure for 2D width decision

with polylogarithmic update time but only for *offline* update sequences; their data structure does not seem to work in our application when we do not know a priori in what order the deletions are intermixed with the insertions.

Both the 2D width and 2D convex hull area problem are about the convex hull. There exist sequences of n updates to the convex hull in the plane which cause $O(n^2)$ many structural changes. For example, consider the case of inserting a point which causes $O(n)$ points to be no longer on the convex hull, then deleting and re-inserting the same point n times. However, in our application points are deleted in the same order as they are inserted, so this particular example cannot occur.

Restricted update sequences on the dynamic convex hull have been studied before. The insertion-only case was studied by Preparata [25]. The deletion-only case was studied by Chazelle [14], and Hershberger and Suri [20]. Random update sequences were studied by Mulmuley [22] and Schwarzkopf [27].

We call an update sequence in which objects are inserted and deleted in the same order a *first-in-first-out (FIFO) update sequence*, which to the best of our knowledge has not been studied before. We prove a combinatorial lemma, stating that for such sequences the number of structural changes to the convex hull is always near linear.

▶ **Lemma 6.** *The number of structural changes to the upper hull of a set of points in $\mathbb{R}^2$ over n FIFO updates is at most $O(n \log n)$.*

The proof of this lemma uses an old observation by Tamir [29] for arbitrary update sequences: although the number of edge creations or destructions in the convex hull could be quadratic, it turns out that the number of distinct edges created or destroyed is near linear.

▶ **Observation 7.** *There are $O(n \log n)$ distinct edges created or destroyed in the upper hull of a set of points in $\mathbb{R}^2$ over an arbitrary sequence of n insertions or deletions.*

Proof. Consider the vertical line at the median x-coordinate. At any time, there is just one hull edge that crosses the vertical line (called the *bridge*), and thus the number of possible bridges over time is $O(n)$. Thus, the number of distinct edges that appear on the upper hull over time satisfies the recurrence

$$E(n) \;=\; 2E(n/2) + O(n)\,,$$

implying that $E(n) = O(n \log n)$. ◀

Lemma 6 now follows immediately by combining the above observation with another observation about FIFO update sequences:

▶ **Observation 8.** *For a FIFO update sequence, once an edge uv has been removed from the upper hull by the insertion of a point w, uv can never again be an edge of the upper hull.*

Proof. Since w is above the line through uv, we know that uv cannot be an edge of the upper hull while w is alive. But since w was inserted after u and v, by the FIFO property w must be deleted after u and v, and therefore uv can never again be an upper hull edge. ◀

3.1 2D width decision

We can immediately apply Lemma 6 to solve the time-windowed 2D width decision problem, by using Eppstein's dynamic 2D width data structure [18] as a black box. Eppstein's algorithm maintains the width in time $O(k \cdot f(n) \cdot \log n)$ where k is the number of structural

changes to the convex hull, and $f(n)$ is the time to solve the dynamic 3D convex hull problem (more precisely, answer gift-wrapping queries and perform updates for a 3D point set). Note that Eppstein used Agarwal and Matoušek as a black box to solve the dynamic 3D convex hull problem in $O(n^\epsilon)$ time [1], but this has since been improved by Chan to $O\big(\log^6 n\big)$ expected time [9], and derandomized by Chan and Tsakalidis [13]. This proves the following result.

▶ **Theorem 9.** *We can preprocess for the time-windowed 2D width decision problem in $O\big(n \log^8 n\big)$ time.*

3.2 2D convex hull area decision

For the time-windowed 2D convex hull area decision problem, we can now directly apply known fully dynamic convex hull data structures [24, 7, 6], most of which can be modified to maintain the area. For example, Brodal and Jacob's (extremely complicated) data structure [6] can maintain the convex hull and its area in $O(k \cdot \log n)$ amortized time, where k is the number of structural changes to the convex hull. This would imply an $O\big(n \log^2 n\big)$-time algorithm, which is worse than Bokal *et al.*'s result [5].

We show how to reduce this to $O(n\alpha(n) \log n)$ by directly adapting a simpler known dynamic convex hull data structure, namely Overmars and van Leeuwen's *hull tree* [24], and carefully analyzing it for FIFO sequences.

▶ **Lemma 10.** *We can maintain the convex hull and its area for a 2D set of n points under FIFO updates in $O(n\alpha(n) \log n)$ total time.*

Proof. It suffices to maintain the upper hull and the area above it inside a sufficiently large bounding box, since we can similarly maintain the lower hull and the area below it, and subtract the areas from the bounding box.

A *hull tree* is a binary tree whose root node stores the upper hull edge (called the *bridge*) that crosses the median vertical line, and whose left and right subtrees are the hull trees on all points left and right of the median, respectively. Here, we assume that the x-coordinates of all n points are known in advance, which is true in our application (the assumption can be removed by extra steps to balance the hull tree, for example, via tree rotations [24]). At each node, we store the area above the upper hull. Pointer structures can be set up to let us traverse the upper hull at any node of the tree [24, 21].

The following definitions will be helpful: Consider the upper hull at a tree node. When we insert a point u which causes a polygonal chain $v_1 v_2 \cdots v_k$ to disappear from the upper hull, we say that u *kills* v_i, and that (u, v_i) forms a *killing pair*, for each $i = 1, 2, \ldots, k$. (For technical reasons, we allow $i = 1$ and $i = k$ in the definition, counterintuitively.) Symmetrically, if we delete a point u which causes a polygonal chain $v_1 v_2 \cdots v_k$ to appear in the upper hull, we say that u *revives* v_i, and that (u, v_i) forms a *revival pair*, for each $i = 1, 2, \ldots, k$.

Deletion. Consider the deletion of a point p at a node of the hull tree. Without loss of generality, suppose that p is to the right of the median. We first recursively delete p in the right subtree. Suppose that p was an endpoint of the bridge of the node. We need to compute a new bridge. Overmars and van Leeuwen [24] originally proposed a binary search, but we will use a linear search instead (inspired by the variants of hull trees by Chazelle [14] and Hershberger and Suri [21] for deletion-only sequences). Specifically, let ℓ_0 be the left endpoint of the old bridge, and let r_0 and r_1 be the predecessor and successor of p in the old

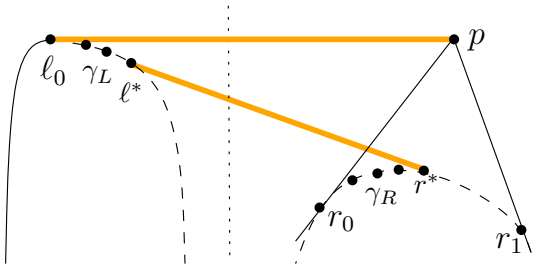


Figure 5 Deletion of p causes the old bridge (ℓ_0, p) to change to the new bridge (ℓ^*, r^*), both shown in **bold**. Computing the new bridge requires walking from ℓ_0 to ℓ^* and r_0 to r^*. If point p is instead being inserted, computing the new bridge requires walking from ℓ^* to ℓ_0. Any pair (p, ℓ) with $\ell \in \gamma_L$ or pair (p, r) with $r \in \gamma_R$ is a revival or killing pair at some node of the hull tree.

right upper hull respectively. (See Figure 5.) A simple rightward linear search from ℓ_0 and r_0 can find the new bridge (ℓ^*, r^*) in $O(|\gamma_L| + |\gamma_R|)$ time, where γ_L denotes the subchain from ℓ_0 to ℓ^* in the left upper hull, and γ_R denotes the subchain from r_0 to r^* in the right upper hull. (Note that ℓ^* must be right of ℓ_0, and r^* must be right of r_0.) The change in area at the current node can be computed in $O(|\gamma_L| + |\gamma_R|)$ time (it is the area of the polygon with vertices $\ell_0\gamma_L\ell^*r^*p$, plus the change in area at the right child, minus the area of the polygon with vertices $r_0\gamma_R r^* p$).

To account for the $O(|\gamma_L|)$ cost, observe that for each $\ell \in \gamma_L$, (p, ℓ) is a revival pair for the upper hull at the current node. We charge one unit to each such pair (p, ℓ). Note that each pair is charged at most once during the entire algorithm.

To account for the $O(|\gamma_R|)$ cost, observe that for each $r \in \gamma_R$, (p, r) is a revival pair for the upper hull at the right child. We charge one unit to each such pair (p, r). If (p, r) is charged, then r lies strictly below the upper hull at the current node and cannot be charged again at an ancestor. Thus, each pair is charged at most once this way.

Insertion. Consider the insertion of a point p at a node of the hull tree. Without loss of generality, suppose that p is to the right of the median. We first recursively insert p in the right subtree. We need to compute the new bridge (if it changes). We can just mimick the deletion algorithm in reverse. In fact, the details are a little simpler: a linear search from ℓ^* can find ℓ_0, the left endpoint of the new bridge (see Figure 5) in $O(|\gamma_L|)$ time. The change in the area can again be computed in $O(|\gamma_L| + |\gamma_R|)$ time. We can account for the cost again by charging, this time, to killing instead of revival pairs.

Total time. The total cost over all updates is proportional to the number of charges, which is bounded by $K(n)$, the worst-case number of distinct pairs (u, v) such that (u, v) is a killing/revival pair for the upper hull of at least one node of the hull tree, over all sets of n points. In the next subsection, we prove that $K(n) = O(n\alpha(n)\log n)$ (Lemma 13), which would then imply an $O(n\alpha(n)\log n)$ time bound. ◀

▶ **Theorem 11.** *We can preprocess for the time-windowed 2D convex hull area decision problem in $O(n\alpha(n)\log n)$ time.*

3.3 Bounding the number of killing/revival pairs

One ingredient is still missing: a proof that $K(n) = O(n\alpha(n)\log n)$. Naively we could bound the number of killing/revival pairs by the number of structural changes to the upper hull at

each node, and applying Lemma 6 would give us the recurrence $K(n) = 2K(n/2)+O(n \log n)$, implying a weaker bound $K(n) = O\big(n \log^2 n\big)$.

We propose a different combinatorial argument to bound $K(n)$. Our approach contains a nice application of *Davenport-Schinzel (DS) sequences* [28]. Recall that a DS sequence *of order 3* is a sequence Σ of characters $s_1, s_2, \ldots$ from an alphabet A such that no two consecutive characters are the same and for any two characters $a, b \in A$, the alternating sequence a, b, a, b, a of length 5 does not appear as a subsequence anywhere in Σ, whether contiguous or not. Hart and Sharir [19, 28] proved that an order-3 DS sequence over an alphabet of size n has length at most $O(n\alpha(n))$.

DS sequences occur often in computational geometry, such as in bounding the combinatorial complexity of the lower envelope of line segments. Surprisingly in our case, we do not relate our problem to lower envelopes or other substructures in arrangements. Rather, we relate directly to DS sequences.

It suffices to bound the number of killing pairs, since revival pairs are symmetric, by reversing time. To this end, we first concentrate on a special kind of killing pairs: for the upper hull at a fixed node of the hull tree, a *bridge killing pair* is a killing pair (u, v) where u and v lie on opposite sides of the median vertical line.

▶ **Lemma 12.** *For the upper hull at a fixed node, the number of bridge killing pairs is at most $O(n\alpha(n))$ for any FIFO update sequence.*

Proof. By symmetry, it suffices to count killing pairs (u, v) where u is to the right and v is to the left of the median vertical line. Take the sequence of all such killing pairs $(u_1, v_1), \ldots, (u_n, v_m)$ ordered by time, where in case of ties, simultaneous killings are ordered in decreasing x-order of v_i. Define the sequence Σ to be $v_1, \ldots, v_m$. We claim that Σ cannot have any alternating subsequence of length 5.

Assume that an alternating subsequence of killed points $\ldots, a, \ldots, b, \ldots, a, \ldots, b, \ldots, a, \ldots$ or $\ldots, b, \ldots, a, \ldots, b, \ldots, a, \ldots, b, \ldots$ of length 5 occurs in Σ. Without loss of generality, assume that a is to the left of b. In either case, the subsequence $\ldots, b, \ldots, a, \ldots, b, \ldots, a, \ldots$ of length 4 occurs in Σ.

Consider the time in this length-4 subsequence when a is killed ($b\underline{a}ba$) and let the vertex which kills it be u (see Figure 6). At this time, b must exist, because it will be killed later; furthermore, b is on or below the current upper hull, and so lies below the line segment $\overline{au}$. Now, fast forward to the time in the subsequence when b is next killed ($ba\underline{b}a$). Note that this must be at a different time, because if u kills a and b at the same time, b would be placed before a in Σ by our tie-breaking rule. At this new time, a and u must both exist, because a will be killed later again, and u was inserted after b and will be deleted after b by definition of FIFO sequences. But b is below $\overline{au}$ and cannot appear on the upper hull and cannot be killed at this time: a contradiction. This completes the proof of the claim.

The sequence Σ may still have identical consecutive characters, but a repeated pair can occur only "between" two different insertion events and there are at most n insertion events. After removal of $O(n)$ repeated characters, Σ thus becomes a DS sequence of order 3 and by the known upper bound has length at most $O(n\alpha(n))$ [28]. The lemma follows. ◀

▶ **Lemma 13.** *The number of distinct killing/revival pairs over all nodes in the hull tree satisfies $K(n) = O(n\alpha(n) \log n)$ for any FIFO update sequence.*

Proof. By Lemma 12, there are $O(n\alpha(n))$ bridge killing pairs for the upper hull at the root node. The remaining killing pairs are killing pairs at nodes of the left subtree and nodes of

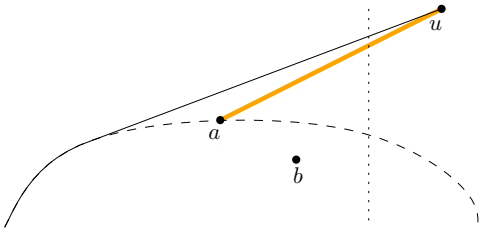


Figure 6 u kills a. The dotted line is the median vertical line and the dashed line was part of the convex hull before u was inserted. The point b must be below the **bold** line segment $\overline{au}$.

the right subtree. We thus obtain the recurrence

$$K(n) \;=\; 2K(n/2) + O(n\alpha(n))\,,$$

implying that $K(n) = O(n\alpha(n)\log n)$. ◀

▶ Remark. We leave open the possibility of removing the tiny $\alpha(n)$ factor in the combinatorial bound on $K(n)$. Alternatively, there is the possibility of reducing $\alpha(n)$ to $\log(\alpha(n))$ in the running time of Lemma 10 and Theorem 11, by replacing linear search with fingered binary search.

Another interesting question is whether Lemma 6's $O(n \log n)$ bound on the number of structural changes to the 2D convex hull for FIFO update sequences can be improved. We are not aware of any superlinear lower bound. (Concerning Lemma 7 for arbitrary update sequences, there is an $\Omega(n\alpha(n))$ lower bound [29].)

Acknowledgements. We wish to thank Haim Kaplan and Micha Sharir for suggesting the use of Tamir's observation [29], which leads to an $\alpha(n)$ factor improvement to our earlier version of Lemma 6.

References

1 Pankaj K. Agarwal and Jirı Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13(4):325–345, 1995.

2 Pankaj K. Agarwal and Micha Sharir. Off-line dynamic maintenance of the width of a planar point set. *Computational Geometry: Theory and Applications*, 1:65–78, 1991. doi:10.1016/0925-7721(91)90001-U.

3 Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 683–694, 1994. doi:10.1109/SFCS.1994.365724.

4 Michael J. Bannister, William E. Devanny, Michael T. Goodrich, Joseph A. Simons, and Lowell Trott. Windows into geometric events: Data structures for time-windowed querying of temporal point sets. In *Proceedings of the 26th Canadian Conference on Computational Geometry (CCCG)*, 2014.

5 Drago Bokal, Sergio Cabello, and David Eppstein. Finding all maximal subsequences with hereditary properties. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG)*, pages 240–254, 2015.

6 Gerth Stølting Brodal and Riko Jacob. Dynamic planar convex hull. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 617–626, 2002.

7 Timothy M. Chan. Dynamic planar convex hull operations in near-logarithmaic amortized time. *Journal of the ACM*, 48(1):1–12, 2001. doi:10.1145/363647.363652.

8 Timothy M. Chan. A fully dynamic algorithm for planar width. In *Proceedings of the 17th Annual Symposium on Computational Geometry (SoCG)*, pages 172–176, 2001.

9 Timothy M. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. ACM*, 57(3):16:1–16:15, 2010. doi:10.1145/1706591.1706596.

10 Timothy M. Chan. Persistent predecessor search and orthogonal point location on the word ram. *ACM Transactions on Algorithms*, 9(3):22, 2013.

11 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Proceedings of the 27th Annual Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

12 Timothy M. Chan and Simon Pratt. Time-windowed closest pair. In *Proceedings of the 27th Canadian Conference on Computational Geometry (CCCG)*, 2015.

13 Timothy M. Chan and Konstantinos Tsakalidis. Optimal deterministic algorithms for 2-d and 3-d shallow cuttings. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG)*, pages 719–732, 2015.

14 Bernard Chazelle. On the convex layers of a planar set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985.

15 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(1-4):133–162, 1986. doi:10.1007/BF01840440.

16 Bernard Chazelle and Leonidas J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1(1-4):163–191, 1986. doi:10.1007/BF01840441.

17 Kenneth L. Clarkson and Peter W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989. doi:10.1007/BF02187740.

18 David Eppstein. Incremental and decremental maintenance of planar width. *Journal of Algorithms*, 37(2):570–577, 2000.

19 Sergiu Hart and Micha Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6(2):151–178, 1986. doi:10.1007/BF02579170.

20 John Hershberger and Subhash Suri. Offline maintenance of planar configurations. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 32–41, 1991.

21 John Hershberger and Subhash Suri. Applications of a semi-dynamic convex hull algorithm. *BIT*, 32(2):249–267, 1992.

22 Ketan Mulmuley. Randomized multidimensional search trees: Lazy balancing and dynamic shuffling. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 180–196, 1991.

23 J. Ian Munro. Tables. In *Proceedings of the 16th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 37–42. Springer, 1996.

24 Mark H. Overmars and Jan van Leeuwen. Maintenance of configurations in the plane. *Journal of Computer and System Sciences*, 23(2):166–204, 1981.

25 Franco P. Preparata. An optimal real-time algorithm for planar convex hulls. *Communications of the ACM*, 22(7):402–405, 1979. doi:10.1145/359131.359132.

26 Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, USA, 1985.

27 Otfried Schwarzkopf. Dynamic maintenance of geometric structures made easy. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 197–206, 1991.

28 Micha Sharir and Pankaj K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, 1995.

29 Arie Tamir. Improved complexity bounds for center location problems on networks by using dynamic data structures. *SIAM Journal on Discrete Mathematics*, 1(3):377–396, 1988. doi:10.1137/0401038.

30 Gelin Zhou. Two-dimensional range successor in optimal time and almost linear space. *Information Processing Letters*, 2015.

Untangling Planar Curves*

Hsien-Chih Chang[1] **and Jeff Erickson**[1]

1 **Department of Computer Science, University of Illinois, Urbana-Champaign, USA**
hchang17@illinois.edu

2 **Department of Computer Science, University of Illinois, Urbana-Champaign, USA**
jeffe@illinois.edu

Abstract

Any generic closed curve in the plane can be transformed into a simple closed curve by a finite sequence of local transformations called homotopy moves. We prove that simplifying a planar closed curve with n self-crossings requires $\Theta(n^{3/2})$ homotopy moves in the worst case. Our algorithm improves the best previous upper bound $O(n^2)$, which is already implicit in the classical work of Steinitz; the matching lower bound follows from the construction of closed curves with large *defect*, a topological invariant of generic closed curves introduced by Aicardi and Arnold. This lower bound also implies that $\Omega(n^{3/2})$ degree-1 reductions, series-parallel reductions, and ΔY transformations are required to reduce any planar graph with treewidth $\Omega(\sqrt{n})$ to a single edge, matching known upper bounds for rectangular and cylindrical grid graphs. Finally, we prove that $\Omega(n^2)$ homotopy moves are required in the worst case to transform one non-contractible closed curve on the torus to another; this lower bound is tight if the curve is homotopic to a simple closed curve.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases computational topology, homotopy, planar graphs, ΔY transformations, defect, Reidemeister moves, tangles

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.29

1 Introduction

Any regular closed curve in the plane can be transformed into a simple closed curve by a finite sequence of the following local operations:

- 1→0: Remove an empty loop.
- 2→0: Separate two subpaths that bound an empty bigon.
- 3→3: Flip an empty triangle by moving one subpath over the opposite intersection point.

See Figure 1. Each of these operations can be performed by continuously deforming the curve within a small neighborhood of one face; consequently, we call these operations and their inverses *homotopy moves*. Our notation is nonstandard but mnemonic; the numbers before and after each arrow indicate the number of local vertices before and after the move. Homotopy moves are "shadows" of the classical Reidemeister moves used to manipulate knot and link diagrams [4, 36].

* Work on this paper was partially supported by NSF grant CCF-1408763. See http://jeffe.cs.illinois.edu/pubs/tangle.pdf for the most recent version of this paper.

© Hsien-Chih Chang and Jeff Erickson;
licensed under Creative Commons License CC-BY

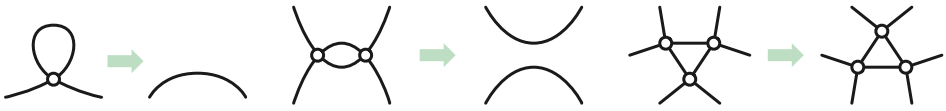


Figure 1 Homotopy moves 1→0, 2→0, and 3→3.

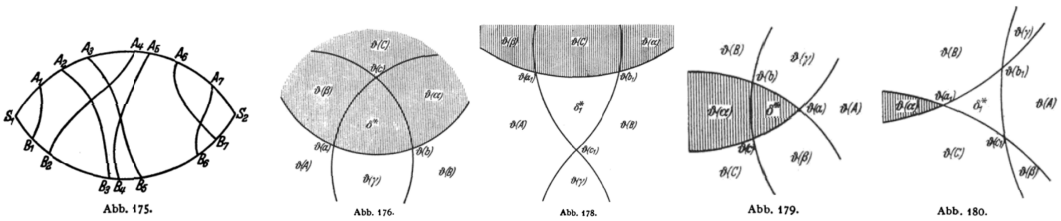


Figure 2 A minimal lens, and 3→3 moves removing triangles from the side or the end of a (shaded) minimal lens. All figures are from Steinitz and Rademacher [42].

We prove that $\Theta(n^{3/2})$ homotopy moves are required in the worst case to simplify a closed curve in the plane with n self-crossings. Before describing our results in more detail, we review several previous results.

1.1 Past Results

An algorithm to simplify any planar closed curve using at most $O(n^2)$ homotopy moves is implicit in Steinitz's proof that every 3-connected planar graph is the 1-skeleton of a convex polyhedron [41, 42]. Specifically, Steinitz proved that any non-simple closed curve (in fact, any 4-regular plane graph) with no empty loops contains a *lens* ("Spindel"): a disk bounded by a pair of simple subpaths that cross exactly twice, where the endpoints of the subpaths lie outside the disk. Steinitz then proves that any *minimal* lens ("irreduzible Spindel") can be transformed into an empty bigon with a sequence of 3→3 moves, each removing one triangular face from the lens, as shown in Figure 2. Once the lens is empty, it can be deleted with a single 2→0 move. See Grünbaum [23], Hass and Scott [25], Colin de Verdière *et al.* [12], or Nowik [33] for more modern treatments of Steinitz's technique. The $O(n^2)$ upper bound also follows from algorithms for *regular* homotopy, which forbids 0↔1 moves, by Francis [19], Vegter [45] (for polygonal curves), and Nowik [33].

The $O(n^2)$ upper bound can also be derived from an algorithm of Feo and Provan [17] for reducing a planar graph to a single edge by *electrical transformations*: degree-1 reductions, series-parallel reductions, and ΔY-transformations. (We consider electrical transformations in more detail in Section 3.) Any curve divides the plane into regions, called its *faces*. The *depth* of a face is its distance to the outer face in the dual graph of the curve. Call a homotopy move *positive* if it decreases the sum of the face depths; in particular, every 1→0 and 2→0 move is positive. Feo and Provan prove that every non-simple curve in the plane admits a positive homotopy move [17, Theorem 1]. Thus, the sum of the face depths is an upper bound on the minimum number of moves required to simplify the curve. Euler's formula implies that every curve with n crossings has $O(n)$ faces, and each of these faces has depth $O(n)$.

Gitler [21] conjectured that a variant of Feo and Provan's algorithm that always makes the *deepest* positive move requires only $O(n^{3/2})$ moves. Song [40] observed that if Feo and Provan's algorithm always chooses the *shallowest* positive move, it can be forced to make $\Omega(n^2)$ moves even when the input curve can be simplified using only $O(n)$ moves.

Tight bounds are known for two special cases that forbid certain types of homotopy moves. First, Nowik proved a tight $\Omega(n^2)$ lower bound for regular homotopy [33]. Second, Khovanov [30] defined two curves to be *doodle equivalent* if one can be transformed into the other using 1↔0 and 2↔0 moves. Khovanov [30] and Ito and Takimura [28] independently proved that any planar curve can be transformed into its unique equivalent doodle with the smallest number of vertices, using only 1→0 and 2→0 moves. Thus, two doodle equivalent curves are connected by a sequence of $O(n)$ moves, which is obviously tight.

1.2 New Results

In Section 2, we derive an $\Omega(n^{3/2})$ lower bound, using a numerical curve invariant called *defect*, introduced by Arnold [8, 7] and Aicardi [1]. Any homotopy move changes the defect of a closed curve by at most 2. Thus, the lower bound follows from constructions of Hayashi *et al.* [26, 27] and Even-Zohar *et al.* [16] of closed curves with defect $\Omega(n^{3/2})$. We simplify and generalize their results by computing the defect of the standard planar projection of any $p \times q$ torus knot where either $p \bmod q = 1$ or $q \bmod p = 1$. Our calculations imply that for any integer p, reducing the standard projection of the $p \times (p+1)$ torus knot requires at least $\binom{p+1}{3} \geq n^{3/2}/6 - O(n)$ homotopy moves.

In Section 3, we sketch a proof, based on arguments of Truemper [43] and Noble and Welsh [32], that reducing a planar graph G using electrical transformations requires at least as many steps as reducing the medial graph of G to a simple closed curve using homotopy moves. The homotopy lower bound from Section 2 then implies that reducing any n-vertex planar graph with treewidth $\Omega(\sqrt{n})$ requires $\Omega(n^{3/2})$ electrical transformations. This lower bound matches known upper bounds for rectangular and cylindrical grid graphs. Due to space limitations, we omit most technical details from this section; we refer the interested reader to our preprint [9].

We develop a new algorithm to simplify any closed curve in $O(n^{3/2})$ homotopy moves in Section 4. First we describe an algorithm that uses $O(D)$ moves, where D is the sum of the face depths of the input curve. At a high level, our algorithm can be viewed as a variant of Steinitz's algorithm that empties and removes *loops* instead of lenses. We then extend our algorithm to *tangles*: collections of boundary-to-boundary paths in a closed disk. Our algorithm simplifies a tangle as much as possible in $O(D + ns)$ moves, where D is the sum of the depths of the tangle's faces, s is the number of paths, and n is the number of intersection points. Finally, we prove that for any curve with maximum face depth $\Omega(\sqrt{n})$, we can find a simple closed curve whose interior tangle has $s = O(\sqrt{n})$ strands, maximum face depth $O(\sqrt{n})$, and at least s^2 interior vertices. Simplifying this tangle and then recursively simplifying the resulting curve requires a total of $O(n^{3/2})$ moves.

Finally, in Section 5, we consider the natural generalization of the homotopy problem to curves on higher-genus surfaces. We prove that $\Omega(n^2)$ homotopy moves are required in the worst case to transform one non-contractible closed curve on the torus to another. Results of Hass and Scott [24] imply that this lower bound is tight if the curve is homotopic to a simple closed curve.

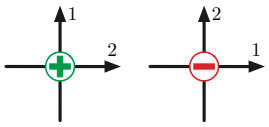
1.3 Definitions

A *closed curve* in a surface M is a continuous map $\gamma\colon S^1 \to M$; in this paper, we consider only *regular* closed curves, which are injective except at a finite number of self-intersections, each of which is a transverse double point. A closed curve is *simple* if it is injective. For

most of the paper, we consider only closed curves in the plane; we consider more general surfaces in Section 5.

The image of any non-simple closed curve has a natural structure as a 4-regular plane graph. Thus, we refer to the self-intersection points of a curve as its *vertices*, the maximal subpaths between vertices as *edges*, and the components of the complement of the curve as its *faces*. Two curves γ and γ' are isomorphic if their images are isomorphic as planar maps; we will not distinguish between isomorphic curves.

We adopt a standard sign convention for vertices first used by Gauss [20]. Choose an arbitrary *basepoint* $\gamma(0)$. We call a vertex *positive* if the first traversal through the vertex crosses the second traversal from right to left, and *negative* otherwise. We define $\operatorname{sgn}(x) = +1$ for every positive vertex x and $\operatorname{sgn}(x) = -1$ for every negative vertex x.



A *homotopy* between two curves γ and γ' is a continuous function $H\colon S^1 \times [0,1] \to M$ such that $H(\cdot,0) = \gamma$ and $H(\cdot,1) = \gamma'$. Each homotopy move can be executed by a homotopy. Conversely, Alexander's simplicial approximation theorem [3], together with combinatorial arguments of Alexander and Briggs [4] and Reidemeister [36], imply that any generic homotopy between two closed curves can be decomposed into a finite sequence of homotopy moves. Two curves are *homotopic*, or in the same *homotopy class*, if there is a homotopy from one to the other. All closed curves in the plane are homotopic.

2 Lower Bounds

2.1 Defect

To prove our main lower bound, we consider a numerical invariant of closed curves in the plane introduced by Arnold [8, 7] and Aicardi [1] called *defect*. Polyak [35] proved that defect can be computed—or for our purposes, defined—as follows:

$$\mathit{defect}(\gamma) := -2 \sum_{x \between y} \operatorname{sgn}(x) \cdot \operatorname{sgn}(y).$$

Here the sum is taken over all *interleaved* pairs of vertices of γ: two vertices $x \neq y$ are interleaved, denoted $x \between y$, if they alternate in cyclic order—x, y, x, y—along γ. Even though the signs of individual vertices depend on the basepoint and orientation of the curve, the defect of a curve is independent of those choices. Trivially, every simple closed curve has defect zero. Straightforward case analysis [35] implies that any single homotopy move changes the defect of a curve by at most 2; the various cases are illustrated in Figure 3.

- A 1→0 move leaves the defect unchanged.
- A 2→0 move decreases the defect by 2 if the two disappearing vertices are interleaved, and leaves the defect unchanged otherwise.
- A 3→3 move increases the defect by 2 if the three vertices before the move contain an even number of interleaved pairs, and decreases the defect by 2 otherwise.

▶ **Lemma 1.** *Let γ be an arbitrary closed curve in the plane. Simplifying γ requires at least $|\mathit{defect}(\gamma)|/2$ homotopy moves.*

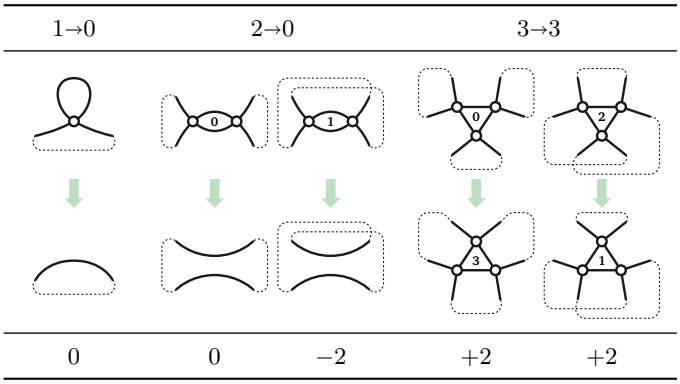


Figure 3 Changes to defect incurred by homotopy moves. Numbers in each figure indicate how many pairs of vertices are interleaved; dashed lines indicate how the rest of the curve connects.

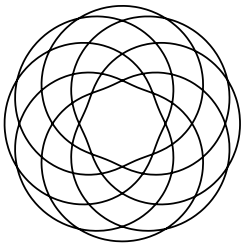


Figure 4 The flat torus knot $T(7,8)$.

2.2 Flat Torus Knots

For any relatively prime positive integers p and q, let $T(p,q)$ denote the curve with the following parametrization, where θ runs from 0 to 2π:

$$T(p,q)(\theta) \coloneqq \big((\cos(q\theta)+2)\cos(p\theta),\ (\cos(q\theta)+2)\sin(p\theta)\big).$$

The curve $T(p,q)$ winds around the origin p times, oscillates q times between two concentric circles, and crosses itself exactly $(p-1)q$ times. We call these curves *flat torus knots*.

Hayashi *et al.* [27, Proposition 3.1] proved that for any integer q, the flat torus knot $T(q+1,q)$ has defect $-2\binom{q}{3}$. Even-Zohar *et al.* [16] used a star-polygon representation of the curve $T(p,2p+1)$ as the basis for a universal model of random knots; in our notation, they proved that $\mathit{defect}(T(p,2p+1)) = 4\binom{p+1}{3}$ for any integer p. In this section we simplify and generalize both of these results to all $T(p,q)$ where either $q \bmod p = 1$ or $p \bmod q = 1$.

▶ **Lemma 2.** *$\mathit{defect}(T(p,ap+1)) = 2a\binom{p+1}{3}$ for all integers $a \geq 0$ and $p \geq 1$.*

Proof. For purposes of illustration, we cut any torus knot $T(p,q)$ open into a "flat braid" consisting of p x-monotone paths, which we call *strands*, between two fixed diagonal lines. All strands are directed from left to right.

The curve $T(p,1)$ can be reduced using only 1→0 moves, so its defect is zero. For any integer $a \geq 0$, we can reduce $T(p,ap+1)$ to $T(p,(a-1)p+1)$ by straightening the leftmost block of $p(p-1)$ crossings in the flat braid representation, one strand at a time. Within this block, each pair of strands in the flat braid intersect twice. Straightening the bottom strand of this block requires the following $\binom{p}{2}$ moves, as shown in Figure 5.

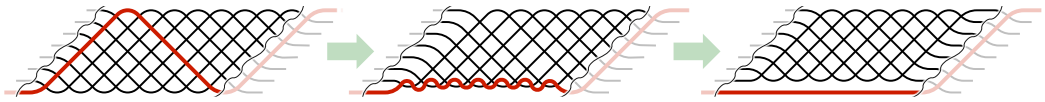


Figure 5 Straightening one strand in a block of $T(8, 8a + 1)$.

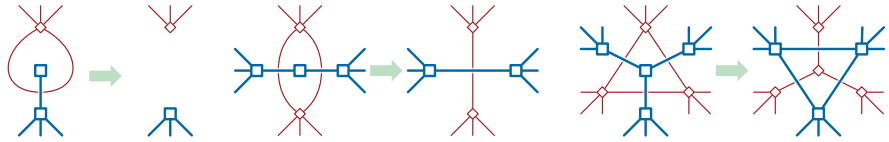


Figure 6 Electrical transformations in a plane graph G and its dual graph G^*.

- $\binom{p-1}{2}$ 3→3 moves pull the bottom strand downward over one intersection point of every other pair of strands. Just before each 3→3 move, exactly one of the three pairs of the three relevant vertices is interleaved, so each move decreases the defect by 2.
- $(p-1)$ 2→0 moves eliminate a pair of intersection points between the bottom strand and every other strand. Each of these moves also decreases the defect by 2.

Altogether, straightening one strand decreases the defect by $2\binom{p}{2}$. Proceeding similarly with the other strands, we conclude that $\mathit{defect}(T(p, ap + 1)) = \mathit{defect}(T(p, (a - 1)p + 1)) + 2\binom{p+1}{3}$. The lemma follows immediately by induction.

◀

A similar argument [9, Lemma 4.3] gives us the exact value of $\mathit{defect}(T(p, q))$ when $p \bmod q = 1$.

▶ **Lemma 3.** *$\mathit{defect}(T(aq + 1, q)) = -2a\binom{q}{3}$ for all integers $a \geq 0$ and $q \geq 1$.*

Either of the previous lemmas imply the following lower bound, which is also implicit in Hayashi *et al.* [27].

▶ **Theorem 4.** *For every positive integer n, there is a closed curve with n crossings that requires at least $n^{3/2}/6 - O(n)$ homotopy moves to reduce to a simple closed curve.*

3 Electrical Transformations

Now we consider a related set of local operations on plane graphs, called *electrical transformations*, consisting of six operations in three dual pairs, as shown in Figure 6.

- *degree-1 reduction*: Contract the edge incident to a vertex of degree 1, or delete the edge incident to a face of degree 1
- *series-parallel reduction*: Contract either edge incident to a vertex of degree 2, or delete either edge incident to a face of degree 2
- *ΔY transformation*: Delete a vertex of degree 3 and connect its neighbors with three new edges, or delete the edges bounding a face of degree 3 and join the vertices of that face to a new vertex.

Electrical transformations have been used since the end of the 19th century [29, 39] to analyze resistor networks and other electrical circuits, but have since been applied to a number of other combinatorial problems on planar graphs, including shortest paths and maximum flows [2]; multicommodity flows [18]; and counting spanning trees, perfect matchings, and

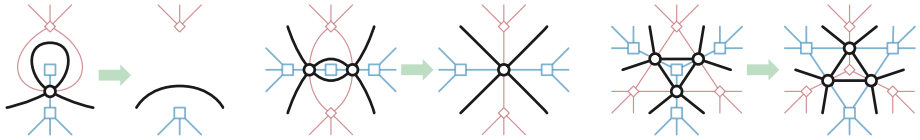


Figure 7 Electrical transformations and medial electrical moves 1→0, 2→1, and 3→3.

cuts [11]. We refer to our earlier preprint [9, Section 1.1] for a more detailed history and an expanded list of applications.

Epifanov [15] was the first to prove that any planar graph with two special vertices called *terminals* can be reduced to a single edge between the terminals by a finite number of electrical transformations. Simpler algorithmic proofs were later given by Truemper [43], Feo and Provan [17], and others. In particular, Truemper's algorithm reduces the $p \times p$ grid using $O(p^3)$ moves, which is the best bound known for this special case. Since every n-vertex planar graph is a minor of a $\Theta(n) \times \Theta(n)$ grid [44], Truemper's algorithm implies an $O(n^3)$ upper bound for arbitrary planar graphs; see Lemma 5. Feo and Provan's algorithm uses $O(n^2)$ electrical transformations, which is the best general upper bound known.

However, for the simpler problem of reducing a planar graph without terminals to a single *vertex*, an $O(n^2)$-move algorithm already follows from the lens-reduction argument of Steinitz described in the introduction [41, 42]. Steinitz reduced local transformations of planar *graphs* to local transformations of planar *curves* by defining *medial graphs* ("Θ-Prozess").

The *medial graph* of a plane graph G, which we denote $G^\times$, is another plane graph whose vertices correspond to the edges of G and whose edges correspond to incidences (with multiplicity) between vertices of G and faces of G. Two vertices of $G^\times$ are connected by an edge if and only 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, and every 4-regular plane graph is a medial graph. A 4-regular plane graph is *unicursal* if it is the image of a single closed curve.

Electrical transformations in any plane graph G correspond to local transformations in the medial graph $G^\times$ that are almost identical to homotopy moves. Each degree-1 reduction in G corresponds to a 1→0 homotopy move in $G^\times$, and each ΔY transformation in G corresponds to a 3→3 homotopy move in $G^\times$. A series-parallel reduction in G contracts an empty bigon in $G^\times$ to a single vertex. Extending our earlier notation, we call this transformation a 2→1 move. We collectively refer to these transformations and their inverses as *medial electrical moves*; see Figure 7.

Here we sketch a proof that $\Omega(n^{3/2})$ electrical transformations are required in the worst case to reduce an n-vertex planar graph to a single vertex. Our proof builds on two key lemmas. The first lemma follows from close reading of the proofs by Truemper [43, Lemma 4] and Gitler [21, Lemma 2.3.3] that if a graph G can be can be reduced to a single vertex by electrical transformations, then so can every minor of G. The second lemma is implicit in the work of Noble and Welsh [32]; informally, we can replace 2→1 medial electrical moves with 2→0 homotopy moves. For self-contained proofs of these lemmas, which argue directly in terms of medial electrical moves, we refer to our earlier preprint [9, Section 3].

▶ **Lemma 5.** *Let G be any plane graph. Reducing any proper minor of G to a single vertex requires strictly fewer electrical transformations than reducing G to a single vertex.*

▶ **Lemma 6.** *Let G be any plane graph whose medial graph $G^\times$ is unicursal. The minimum number of homotopy moves required to reduce $G^\times$ to a simple closed curve is no greater than the minimum number of electrical transformations required to reduce G to a single vertex.*

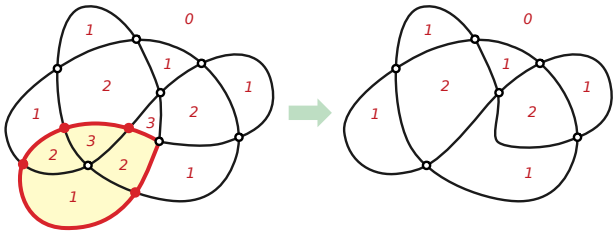


Figure 8 Transforming γ into γ' by contracting a simple loop. Numbers are face depths.

▶ **Theorem 7.** *For every positive integer t, every planar graph with treewidth t requires $\Omega(t^3)$ electrical transformations to reduce to a single vertex.*

Proof. Every planar graph with treewidth t contains an $\Omega(t) \times \Omega(t)$ grid minor [38], which in turn contains an $k \times (2k+1)$ cylindrical grid minor for some integer $k = \Omega(t)$. The medial graph of the $k \times (2k+1)$ cylindrical grid is the flat torus knot $T(2k, 2k+1)$. The theorem now follows from Lemmas 1, 2, 5, and 6. ◀

In particular, reducing any planar graph with n vertices and treewidth $\Theta(\sqrt{n})$ requires $\Omega(n^{3/2})$ electrical transformations. It follows that Truemper's $O(p^3)$ upper bound for reducing the $p \times p$ square grid [43] is tight. Similar arguments using Lemmas 2 and 3 imply that Nakahara and Takahashi's $O(\min\{pq^2, p^2q\})$ upper bound for reducing the $p \times q$ cylindrical grid [31] is also tight.

Like Gitler [21], Feo and Provan [17], and Archdeacon *et al.* [6], we conjecture that any planar graph with n vertices can be reduced using only $O(n^{3/2})$ electrical transformations, but so far we have only been able to prove a matching upper bound for homotopy moves.

4 Upper Bound

For any point p, let $\mathit{depth}(p, \gamma)$ denote the minimum number of times a path from p to infinity crosses γ. Any two points in the same face of γ have the same depth, so each face f has a well-defined depth, which is its distance to the outer face in the dual graph of γ; see Figure 8. The depth of the curve, denoted $\mathit{depth}(\gamma)$, is the maximum depth of the faces of γ; and the *potential* $D(\gamma)$ is the sum of the depths of the faces. Euler's formula implies that any 4-regular planar graph with n vertices has exactly $n+2$ faces; thus, for any curve γ with n vertices, we have $n + 1 \leq D(\gamma) \leq (n+1) \cdot \mathit{depth}(\gamma)$.

4.1 Contracting Simple Loops

▶ **Lemma 8.** *Every closed curve γ in the plane can be simplified using at most $3D(\gamma) - 3$ homotopy moves.*

Proof. The lemma is trivial if γ is already simple, so assume otherwise. Let $x := \gamma(\theta) = \gamma(\theta')$ be the first vertex to be visited twice by γ after the (arbitrarily chosen) basepoint $\gamma(0)$. Let ℓ denote the subcurve of γ from $\gamma(\theta)$ to $\gamma(\theta')$; our choice of x implies that ℓ is a simple loop. Let m and s denote the number of vertices and maximal subpaths of γ in the interior of ℓ respectively. Finally, let γ' denote the closed curve obtained from γ by removing ℓ. The first stage of our algorithm transforms γ into γ' by contracting the loop ℓ via homotopy moves.

We remove the vertices and edges from the interior of ℓ one at a time as follows. If we can perform a 2→0 move to remove one edge of γ from the interior of ℓ and decrease s, we

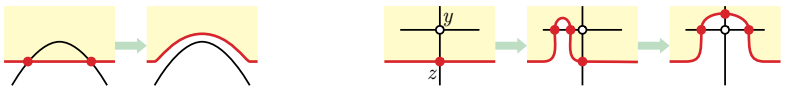


Figure 9 Moving a loop over an interior empty bigon or an interior vertex.

do so. Otherwise, either ℓ is empty, or some vertex of γ lies inside ℓ. Let y be a vertex of γ that lies inside ℓ and has at least one neighbor z that lies on ℓ; we move y outside ℓ with a 0→2 move (which increases s) followed by a 3→3 move, as shown on the right of Figure 9.

Once ℓ is an empty loop, we remove it with a single 1→0 move. Altogether, our algorithm transforms γ into γ' using at most $3m + s + 1$ homotopy moves. Let M denote the actual number of homotopy moves used.

Euler's formula implies that ℓ contains exactly $m + s/2 + 1$ faces of γ. The Jordan Curve theorem implies that $\mathit{depth}(p, \gamma') \leq \mathit{depth}(p, \gamma) - 1$ for any point p inside ℓ, and trivially $\mathit{depth}(p, \gamma') \leq \mathit{depth}(p, \gamma)$ for any point p outside ℓ. It follows that $D(\gamma') \leq D(\gamma) - (m + s/2 + 1) \leq D(\gamma) - M/3$, and therefore $M \leq 3D(\gamma) - 3D(\gamma')$. The induction hypothesis implies that we can recursively simplify γ' using at most $3D(\gamma') - 3$ moves. The lemma now follows immediately. ◀

Our upper bound is a factor of 3 larger than Feo and Provan's [17]; however our algorithm has the benefit that it extends to *tangles*, as described in the next subsection.

4.2 Tangles

A *tangle* is a collection of boundary-to-boundary paths $\gamma_1, \gamma_2, \ldots, \gamma_s$ in a closed topological disk Σ, which (self-)intersect only pairwise, transversely, and away from the boundary of Σ. (In knot theory, a tangle usually refers to the intersection of a knot or link with a closed 3-dimensional ball [13, 10]; our object is more properly called a *flat tangle*, as it is the image of a tangle under an appropriate projection. Our tangles are unrelated to the obstructions to small branchwidth studied by Robertson and Seymour [37].)

We call each individual path γ_i a *strand* of the tangle. The *boundary* of a tangle is the boundary of the disk Σ that contains it; we usually denote the boundary by σ. By the Jordan-Schönflies theorem, we can assume without loss of generality that σ is actually a circle. We can obtain a tangle from any closed curve γ by considering its restriction to any closed disk whose boundary σ intersects γ transversely away from its vertices; we call this restriction the *interior tangle* of σ.

The strands and boundary of any tangle define a plane graph T whose boundary vertices each have degree 3 and whose interior vertices each have degree 4. Depths and potential are defined exactly as for closed curves: The depth of any face f of T is its distance to the outer face in the dual graph T^*; the depth of the tangle is its maximum face depth; and the potential $D(T)$ of the tangle is the sum of its face depths.

A tangle is *tight* if every pair of strands intersects at most once and *loose* otherwise. Every loose tangle contains either an empty loop or a (not necessarily empty) lens. Thus, any tangle with n vertices can be transformed into a tight tangle—or less formally, *tightened*—in $O(n^2)$ homotopy moves using Steinitz's algorithm. On the other hand, there are infinite classes of loose tangles for which no homotopy move decreases the potential, so we cannot directly apply Feo and Provan's algorithm to this setting.

▶ **Lemma 9.** *Every tangle T with n vertices and s strands can be tightened using at most $3D(T) + 3ns$ homotopy moves.*

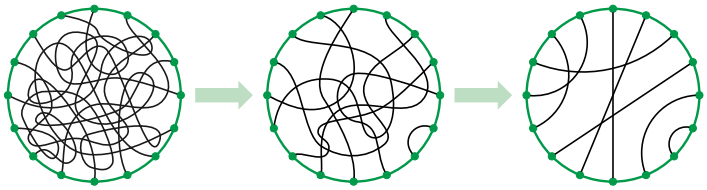


Figure 10 Tightening a tangle in two phases: First simplifying the individual strands, then removing excess crossings between pairs of strands.

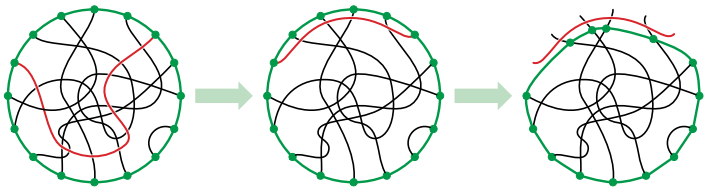


Figure 11 Moving one strand out of the way and shrinking the tangle boundary.

Proof. Our algorithm consists of two stages: first we simplify the individual strands using at most $3D(T)$ homotopy moves, and then we remove excess intersections between every pair of strands using at most $3ns$ homotopy moves.

First, as long as any strand in T is non-simple, we identify a simple loop ℓ in that strand and remove it as described in the proof of Lemma 8. Let T' be the remaining strand after all such loops are removed. The analysis in the proof of Lemma 8 implies that transforming T into T' requires at most $3D(T) - 3D(T') \leq 3D(T)$ homotopy moves.

Now fix an arbitrary reference point on the boundary circle σ that is not an endpoint of a strand. For each index i, let σ_i be the arc of σ between the endpoints of γ_i that does not contain the reference point. A strand γ_i is *extremal* if the corresponding arc σ_i does not contain any other arc σ_j.

Choose an arbitrary extremal strand γ_i. Let m_i denote the number of tangle vertices in the interior of the disk bounded by γ_i and the boundary arc σ_i; let s_i denote the number of intersections between γ_i and other strands. Finally, let γ_i' be a path inside the disk Σ defining tangle T, with the same endpoints as γ_i, that intersects each other strand in T at most once, such that the disk bounded by σ_i and γ_i' has no tangle vertices inside its interior.

We can deform γ_i into γ_i' using essentially the algorithm from Lemma 8. If the disk bounded by γ_i and σ_i contains an empty bigon, remove it with a 2→0 move. If the disk has an interior vertex with a neighbor on γ_i, remove it using at most two homotopy moves (and possibly increasing s_i). Altogether, this deformation requires at most $3m_i + s_i \leq 3n$ homotopy moves.

After deforming γ_i to γ_i', we shrink the boundary of the tangle slightly to exclude γ_i', without creating or removing any additional endpoints on the boundary or vertices in the tangle. We emphasize that shrinking the boundary does not modify the strands and therefore does not require any homotopy moves. The resulting smaller tangle has exactly $s-1$ strands, each of which is simple. Thus, the induction hypothesis implies that we can recursively tighten this smaller tangle using at most $3n(s-1)$ homotopy moves. The base case of this recursion is a tangle with no strands. ◀

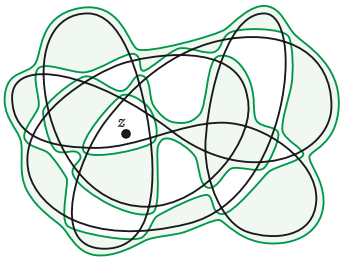


Figure 12 Nested depth cycles around a point of maximum depth.

4.3 Main Algorithm

We call a simple closed curve σ *useful* for γ if it intersects γ transversely away from its vertices, and the interior tangle of σ has at least s^2 vertices, where $s = |\sigma \cap \gamma|/2$ is the number of strands. Our main algorithm repeatedly finds a useful closed curve and tightens its interior tangle; if there are no useful closed curves, then we fall back to the loop-contraction algorithm of Lemma 8.

▶ **Lemma 10.** *Let γ be an arbitrary closed curve in the plane with n vertices. Either γ has a useful simple closed curve whose interior tangle has depth $O(\sqrt{n})$, or γ itself has depth $O(\sqrt{n})$.*

Proof. To simplify notation, let $d \coloneqq \mathit{depth}(\gamma)$. For each integer j between 1 and d, let R_j be the set of points p with $\mathit{depth}(p, \gamma) \geq d + 1 - j$, and let $\tilde{R}_j$ denote a small open neighborhood of the closure of $R_j \cup \tilde{R}_{j-1}$, where $\tilde{R}_0$ is the empty set. Each region $\tilde{R}_j$ is the disjoint union of closed disks, whose boundary cycles intersect γ transversely away from its vertices, if at all. In particular, $\tilde{R}_d$ is a disk containing the entire curve γ.

Fix a point z such that $\mathit{depth}(z, \gamma) = d$. For each integer j, let Σ_j be the unique component of $\tilde{R}_j$ that contains z, and let σ_j be the boundary of Σ_j. Then $\sigma_1, \sigma_2, \ldots, \sigma_d$ are disjoint, nested, simple closed curves; see Figure 12. Let n_j be the number of vertices and let $s_j \coloneqq |\gamma \cap \sigma_j|/2$ be the number of strands of the interior tangle of σ_j. For notational convenience, we define $\Sigma_0 \coloneqq \varnothing$ and thus $n_0 = s_0 = 0$.

By construction, for each j, the interior tangle of σ_j has depth $j + 1$. Thus, to prove the lemma, it suffices to show that either $\mathit{depth}(\gamma) = O(\sqrt{n})$ or at least one curve σ_j with $j = O(\sqrt{n})$ is useful.

Fix an index j. Each edge of γ crosses σ_j at most twice. Any edge of γ that crosses σ_j has at least one endpoint in the annulus $\Sigma_j \setminus \Sigma_{j-1}$, and any edge that crosses σ_j twice has both endpoints in $\Sigma_j \setminus \Sigma_{j-1}$. Conversely, each vertex in Σ_j is incident to at most two edges that cross σ_j and no edges that cross σ_{j+1}. It follows that $|\sigma_j \cap \gamma| \leq 2(n_j - n_{j-1})$, and therefore $n_j \geq n_{j-1} + s_j$. Thus, by induction, we have

$$n_j \geq \sum_{i \leq j} s_i$$

for every index j.

Now suppose no curve σ_j with $1 \leq j \leq d$ is useful. Then we must have $s_j^2 > n_j$ and therefore

$$s_j^2 > \sum_{i \leq j} s_i$$

for all $j \geq 1$. Trivially, $s_1 \geq 1$ unless γ is simple and thus $d = 1$. A straightforward induction argument implies that $s_j \geq (j+1)/2$ and therefore

$$n \;\geq\; n_d \;\geq\; \sum_{i \leq d} \frac{i+1}{2} \;\geq\; \frac{1}{2}\binom{d+2}{2} \;>\; \frac{d^2}{4}.$$

We conclude that $d \leq 2\sqrt{n}$, which completes the proof. ◀

▶ **Theorem 11.** *Every closed curve γ in the plane with n vertices can be simplified in $O(n^{3/2})$ homotopy moves.*

Proof. If γ has depth $O(\sqrt{n})$, Lemma 8 and the trivial upper bound $D(\gamma) \leq (n+1) \cdot \mathit{depth}(\gamma)$ imply that we can simplify γ in $O(n^{3/2})$ homotopy moves. For purposes of analysis, we charge $O(\sqrt{n})$ of these moves to each vertex of γ.

Otherwise, let σ be an arbitrary useful closed curve chosen according to Lemma 10. Suppose the interior tangle of σ has m vertices, s strands, and depth d. Lemma 10 implies that $d = O(\sqrt{n})$, and the definition of useful implies that $s \leq \sqrt{m}$, which is $O(\sqrt{n})$. Thus, by Lemma 9, we can tighten the interior tangle of σ in $O(md + ms) = O(m\sqrt{n})$ moves. This simplification removes at least $m - s^2/2 \geq m/2$ vertices from γ, as the resulting tight tangle has at most $s^2/2$ vertices. Again, for purposes of analysis, we charge $O(\sqrt{n})$ moves to each deleted vertex. We then recursively simplify the resulting closed curve.

In either case, each vertex of γ is charged $O(\sqrt{n})$ moves as it is deleted. Thus, simplification requires at most $O(n^{3/2})$ homotopy moves in total. ◀

5 Higher-Genus Surfaces

Finally, we consider the natural generalization of our problem to closed curves on orientable surfaces of higher genus. Because these surfaces have non-trivial topology, not every closed curve is homotopic to a single point or even to a simple curve. A closed curve is *contractible* if it is homotopic to a single point. We call a closed curve *tight* if it has the minimum number of self-intersections in its homotopy class.

5.1 Lower Bounds

Although defect was originally defined as an invariant of *planar* curves, Polyak's formula $\mathit{defect}(\gamma) = -2\sum_{x \between y} \operatorname{sgn}(x)\operatorname{sgn}(y)$ extends naturally to closed curves on any orientable surface; homotopy moves change the resulting invariant exactly as described in Figure 3. Thus, Lemma 1 immediately generalizes to any orientable surface as follows.

▶ **Lemma 12.** *Let γ and γ' be arbitrary closed curves that are homotopic on an arbitrary orientable surface. Transforming γ into γ' requires at least $|\mathit{defect}(\gamma) - \mathit{defect}(\gamma')|/2$ homotopy moves.*

The following construction implies a quadratic lower bound for simplifying noncontractible curves on orientable surfaces with any positive genus.

▶ **Lemma 13.** *For any positive integer n, there is a closed curve on the torus with n vertices and defect $\Omega(n^2)$ that is homotopic to a simple closed curve but not contractible.*

Proof. Without loss of generality, suppose n is a multiple of 8. The curve γ is illustrated on the left in Figure 13. The torus is represented by a rectangle with opposite edges identified. We label three points a, b, c on the vertical edge of the rectangle and decompose the curve

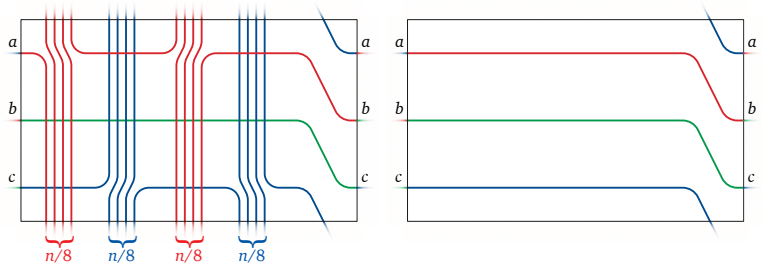


Figure 13 A curve γ on the torus with defect $\Omega(n^2)$ and a simple curve homotopic to γ.

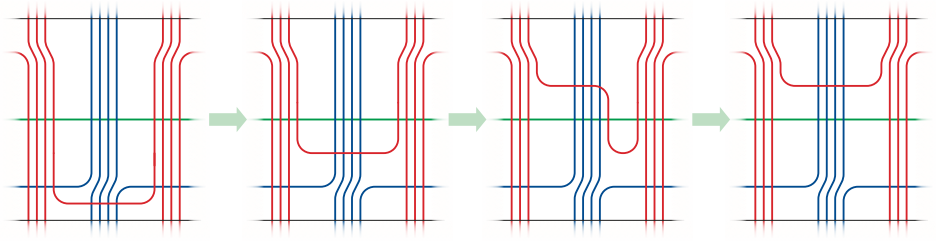


Figure 14 Unwinding one turn of the red path.

into a red path from a to b, a green path from b to c, and a blue path from c to a. The red and blue paths each wind vertically around the torus, first $n/8$ times in one direction, and then $n/8$ times in the opposite direction.

As in previous proofs, we compute the defect of γ by describing a sequence of homotopy moves that simplify the curve, while carefully tracking the changes in the defect that these moves incur. We can unwind one turn of the red path by performing one 2→0 move, followed by $n/8$ 3→3 moves, followed by one 2→0 move, as illustrated in Figure 14. Repeating this sequence of homotopy moves $n/8$ times removes all intersections between the red and green paths, after which a sequence of $n/4$ 2→0 moves straightens the blue path, yielding the simple curve shown on the right in Figure 13. Altogether, we perform $n^2/64 + n/2$ homotopy moves, where each 3→3 move increases the defect of the curve by 2 and each 2→0 move decreases the defect of the curve by 2. We conclude that $\mathit{defect}(\gamma) = -n^2/32 + n$. ◀

▶ **Theorem 14.** *Simplifying a closed curve with n crossings on a torus requires $\Omega(n^2)$ homotopy moves in the worst case, even if the curve is homotopic to a simple curve.*

5.2 Upper Bounds

Hass and Scott proved that any non-simple closed curve on any orientable surface that is homotopic to a simple closed curve contains either a simple (in fact empty) contractible loop or a simple contractible lens [24, Theorem 1]. It follows immediately that any such curve can be simplified in $O(n^2)$ moves using Steinitz's algorithm; Theorem 14 implies that the upper bound is tight for non-contractible curves.

For the most general setting, where the given curve is not necessarily homotopic to a simple closed curve, we are not even aware of a *polynomial* upper bound! Steinitz's algorithm does not work here; there are curves with excess self-intersections but no simple contractible loops or lenses [24]. Hass and Scott [25] and De Graff and Schrijver [14] independently proved that any closed curve on any surface can be simplified using a *finite* number of

homotopy moves that never increase the number of self-intersections. Both proofs use discrete variants of curve-shortening flow; for sufficiently well-behaved curves and surfaces, results of Grayson [22] and Angenent [5] imply a similar result for differential curvature flow. Unfortunately, without further assumptions about the precise geometries of both the curve and the underlying surface, the number of homotopy moves cannot be bounded by any function of the number of crossings; even in the plane, there are closed curves with three crossings for which curve-shortening flow alternates between a 3→3 move and its inverse arbitrarily many times. Paterson [34] describes a combinatorial algorithm to compute a simplifying sequence of homotopy moves without such reversals, but she offers no analysis of her algorithm.

We conjecture that any contractible curve on any surface can be simplified with at most $O(n^{3/2})$ homotopy moves, and that arbitrary curves on any surface can be simplified with at most $O(n^2)$ homotopy moves.

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 Sheldon B. Akers, Jr. The use of wye-delta transformations in network simplification. *Oper. Res.*, 8(3):311–323, 1960.

3 James W. Alexander. Combinatorial analysis situs. *Trans. Amer. Math. Soc.*, 28(2):301–326, 1926.

4 James W. Alexander and G. B. Briggs. On types of knotted curves. *Ann. Math.*, 28(1/4):562–586, 1926–1927.

5 Sigurd Angenent. Prabolic equations for curves on surfaces: Part II. Intersections, blow-up and generalized solutions. *Ann. Math.*, 133(1):171–215, 1991.

6 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.

7 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.

8 Vladimir I. Arnold. *Topological Invariants of Plane Curves and Caustics*, volume 5 of *University Lecture Series*. Amer. Math. Soc., 1994.

9 Hsien-Chih Chang and Jeff Erickson. Electrical reduction, homtoopy moves, and defect. Preprint, October 2015. URL: `http://arxiv.org/abs/1510.00571`.

10 Sergei Chmutov, Sergei Duzhin, and Jacob Mostovoy. *Introduction to Vassiliev knot invariants*. Cambridge Univ. Press, 2012.

11 Charles J. Colbourn, J. Scott Provan, and Dirk Vertigan. A new approach to solving three combinatorial enumeration problems on planar graphs. *Discrete Appl. Math.*, 60:119–129, 1995.

12 Yves Colin de Verdière, Isidoro Gitler, and Dirk Vertigan. Réseaux électriques planaires II. *Comment. Math. Helvetici*, 71(144–167), 1996.

13 John H. Conway. An enumeration of knots and links, and some of their algebraic properties. In by:John Leech, editor, *Computational Problems in Abstract Algebra*, pages 329–358. Pergamon Press, 1970.

14 Maurits de Graaf and Alexander Schrijver. Making curves minimally crossing by Reidemeister moves. *J. Comb. Theory Ser. B*, 70(1):134–156, 1997.

15 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.

16 Chaim Even-Zohar, Joel Hass, Nati Linial, and Tahl Nowik. Invariants of random knots and links. Preprint, November 2014. URL: `http://arxiv.org/abs/1411.3308`.

17 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.

18 Thomas Aurelio 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.

19 George K. Francis. The folded ribbon theorem: A contribution to the study of immersed circles. *Trans. Amer. Math. Soc.*, 141:271–303, 1969.

20 Carl Friedrich Gauß. Nachlass. I. Zur Geometria situs. In *Werke*, volume 8, pages 271–281. Teubner, 1900. Originally written between 1823 and 1840.

21 Isidoro Gitler. *Delta-wye-delta Transformations: Algorithms and Applications.* PhD thesis, Department of Combinatorics and Optimization, University of Waterloo, 1991.

22 Matthew A. Grayson. Shortening embedded curves. *Ann. Math.*, 129(1):71–111, 1989.

23 Branko Grünbaum. *Convex Polytopes.* Number XVI in Monographs in Pure and Applied Mathematics. John Wiley & Sons, 1967.

24 Joel Hass and Peter Scott. Intersections of curves on surfaces. *Israel J. Math.*, 51:90–120, 1985.

25 Joel Hass and Peter Scott. Shortening curves on surfaces. *Topology*, 33(1):25–43, 1994.

26 Chuichiro Hayashi and Miwa Hayashi. Minimal sequences of Reidemeister moves on diagrams of torus knots. *Proc. Amer. Math. Soc.*, 139:2605–2614, 2011.

27 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.

28 Noburo Ito and Yusuke Takimura. (1,2) and weak (1,3) homotopies on knot projections. *J. Knot Theory Ramif.*, 22(14):1350085 (14 pages), 2013. Addendum in *J. Knot Theory Ramif.* 23(8):1491001 (2 pages), 2014.

29 Arthur Edwin Kennelly. Equivalence of triangles and three-pointed stars in conducting networks. *Electrical World and Engineer*, 34(12):413–414, 1899.

30 Mikhail Khovanov. Doodle groups. *Trans. Amer. Math. Soc.*, 349(6):2297–2315, 1997.

31 Hiroyuki Nakahara and Hiromitsu Takahashi. An algorithm for the solution of a linear system by Δ-Y transformations. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E79-A(7):1079–1088, 1996.

32 Steven D. Noble and Dominic J. A. Welsh. Knot graphs. *J. Graph Theory*, 34(1):100–111, 2000.

33 Tahl Nowik. Complexity of planar and spherical curves. *Duke J. Math.*, 148(1):107–118, 2009.

34 Jane M. Paterson. A combinatorial algorithm for immersed loops in surfaces. *Topology Appl.*, 123:205–234, 2002.

35 Michael Polyak. Invariants of curves and fronts via Gauss diagrams. *Topology*, 37(5):989–1009, 1998.

36 Kurt Reidemeister. Elementare Begründung der Knotentheorie. *Abh. Math. Sem. Hamburg*, 5:24–32, 1927.

37 Neil Robertson and Paul D. Seymour. Graph minors. X. Obstructions to tree-decomposition. *J. Comb. Theory Ser. B*, 52(2):153–190, 1991.

38 Neil Robertson, Paul D. Seymour, and Robin Thomas. Quickly excluding a planar graph. *J. Comb. Theory Ser. B*, 62(2):232–348, 1994.

39 Alexander Russell. The method of duality. In *A Treatise on the Theory of Alternating Currents*, chapter XVII, pages 380–399. Cambridge Univ. Press, 1904.

40 Xiaohuan Song. Implementation issues for Feo and Provan's delta-wye-delta reduction algorithm. M.Sc. Thesis, University of Victoria, 2001.

41 Ernst Steinitz. Polyeder und Raumeinteilungen. *Enzyklopädie der mathematischen Wissenschaften mit Einschluss ihrer Anwendungen*, III.AB(12):1–139, 1916.

42 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.

43 Klaus Truemper. On the delta-wye reduction for planar graphs. *J. Graph Theory*, 13(2):141–148, 1989.

44 Leslie S. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Comput.*, C-30(2):135–140, 1981.

45 Gert Vegter. Kink-free deformation of polygons. In *Proceedings of the 5th Annual Symposium on Computational Geometry*, pages 61–68, 1989.

Inserting Multiple Edges into a Planar Graph

Markus Chimani[*] [1] **and Petr Hliněný**[†] [2]

1 **Theoretical Computer Science, University Osnabrück, Osnabrück, Germany**
markus.chimani@uni-osnabrueck.de
2 **Faculty of Informatics, Masaryk University Brno, Brno, Czech Republic**
hlineny@fi.muni.cz

Abstract

Let G be a connected planar (but not yet embedded) graph and F a set of additional edges not in G. The *multiple edge insertion* problem (MEI) asks for a drawing of $G+F$ with the minimum number of pairwise edge crossings, such that the subdrawing of G is plane. An optimal solution to this problem is known to approximate the crossing number of the graph $G+F$.

Finding an exact solution to MEI is NP-hard for general F, but linear time solvable for the special case of $|F|=1$ (SODA 01, Algorithmica) and polynomial time solvable when all of F are incident to a new vertex (SODA 09). The complexity for general F but with constant $k=|F|$ was open, but algorithms both with relative and absolute approximation guarantees have been presented (SODA 11, ICALP 11). We show that the problem is fixed parameter tractable (FPT) in k for biconnected G, or if the cut vertices of G have bounded degrees. We give the first exact algorithm for this problem; it requires only $\mathcal{O}(|V(G)|)$ time for any constant k.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases crossing number; edge insertion; parameterized complexity; path homotopy; funnel algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.30

1 Introduction

The crossing number $\mathrm{cr}(G)$ of a graph G is the minimum number of pairwise edge crossings in a drawing of G in the plane. Finding the crossing number of a graph is one of the most prominent difficult optimization problems in graph theory [18] and is NP-hard already in very restricted cases, e.g., even when considering a planar graph with one added edge [6] (such graphs are called *almost-planar* or near-planar). The problem has been vividly investigated for over 60 years, but there is still surprisingly little known about it; see e.g. [32] for an extensive reference. There exists a $c>1$ such that $\mathrm{cr}(G)$ cannot be approximated within a factor c in polynomial time [4], but we do not know whether $\mathrm{cr}(G)$ is approximable within some constant ratio for general G.

Several approximation algorithms arose for special graph classes. For general graphs with bounded degree, there is an algorithm that approximates the quantity $n+\mathrm{cr}(G)$ instead, giving an approximation ratio of $\mathcal{O}(\log^2 n)$ [2, 17]. A sublinear approximation factor of $\tilde{\mathcal{O}}(n^{0.9})$ for $\mathrm{cr}(G)$ in the bounded-degree setting was given in an involved algorithm [13]. We know constant factor approximations for bounded-degree graphs that are embeddable in

[*] M. Chimani has been supported by the German Research Foundation (DFG) project CH 897/2-1.
[†] P. Hliněný has been supported by the research center Institute for Theoretical Computer Science (CE-ITI); Czech Science foundation project No. P202/12/G061.


© Markus Chimani and Petr Hliněný;
licensed under Creative Commons License CC-BY

some surface of higher genus [19, 26, 24], or that have a small set of graph elements whose deletion leaves a planar graph – removing and re-inserting these elements can give strong approximation bounds such as [25, 5, 12, 14].

In this paper, we follow the latter idea and concentrate on the *Multiple Edge Insertion* problem MEI(G, F). Intuitively, we are given a planar graph G, and ask for the best way (in terms of total crossing number) to draw G and insert a set of new edges F into G such that the final drawing of $G + F$ (i.e., of the graph including the new edges of F) restricted to G remains planar. The formal definitions follow in Section 2.

This MEI problem is polynomial-time solvable for $|F| = 1$ [22] (unlike the crossing number problem of almost-planar graphs) or if all edges of F are incident to a common vertex [10], but NP-hard for general F [35]. An exact or at least approximate MEI solution constitutes an approximation for the crossing number of the graph $G+F$ [12]; see Section 2. Considering general constant $k := |F|$, there have been two different approximation approaches [14] and [11]; the former one directly targets the crossing number and achieves only a relative approximation guarantee for MEI; the latter one first specifically attains an approximation of MEI with only an additive error term, and then uses [12] to deduce a crossing number approximation. While the former is not directly practical, the latter algorithm [11] in fact turns out to be one of the best choices to obtain strong upper bounds in practice [9].

In this paper, we show that for every constant k and under mild connectivity assumptions, there is an exact linear time algorithm; till now, this was an open problem even for $k = 2$.

▶ **Theorem 1.** *Let G be a planar graph and F a set of $k \geq 1$ new edges (vertex pairs, in fact). Assume G is biconnected, or G is connected and all cut vertices of G have degree bounded from above by $2^{p(k)}$ where p is a polynomial. Then there is a polynomial function q such that the problem* MEI(G, F) *is solvable to optimality in time $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$. In terms of parameterized complexity, our algorithm is linear-time FPT with the parameter $k = |F|$.*

Note that the bounded-degree requirement for cut vertices compares favorably to existing approximation algorithms for the crossing number, all of which require a degree bound for *all* vertices. We also remark that while the crossing number problem is in FPT w.r.t. the objective value (cr(G)) [20, 28], already a planar graph with one added edge may have unbounded crossing number and so these two modes of parameterization are incomparable.

Our high-level approach is a standard idea in this area, using dynamic programming on a decomposition (known as SP(Q)R-tree) of the planar graph; see Section 4. Similar ideas have been used, e.g., in aforementioned [22, 10, 14, 11]. However, this time it turns out that the most interesting and difficult case is the basic one of rigid components. The corresponding problem *Rigid MEI*, i.e., MEI under the restriction that a planar embedding of G is fixed beforehand, is still NP-hard [35]. An FPT algorithm for Rigid MEI is given in Section 3.

On an informal level, the algorithm for Rigid MEI simultaneously searches for shortest dual paths corresponding to the edges of F in rigid G, while keeping track of their mutual crossings. Although this task may seem similar, in the dual, to the notoriously hard problem of shortest disjoint paths in planar graphs [15, 30], there is the crucial difference that our paths may share sections as long as they do not cross. Our algorithm utilizes the concept of *path homotopy* in the plane with obstacles, and uses a special structure which we call a *trinet*, to represent and search for shortest dual paths of a given homotopy.

Organization. Due to restricted space, the proofs of some statements are left for the full version of this paper (see also `http://arxiv.org/abs/1509.07952`); these statements are marked with an asterisk *.

2 Definitions

We use the standard terminology of graph theory. By default, we use the term *graph* to refer to a multigraph, i.e., we allow parallel edges (also self-loops are allowed but those can be safely ignored in the context of crossing numbers in the plane). If there is no danger of confusion, we denote an edge with the ends u and v chiefly by uv.

Drawing a graph. A *drawing* of a graph $G = (V, E)$ is a mapping of the vertices V to distinct points on a surface Σ, and of the edges E to simple (polygonal) curves on Σ, connecting their respective end points but not containing any other vertex point. Unless explicitly specified, we will always assume Σ to be the plane (or, equivalently, the sphere). A *crossing* is a common point of two distinct edge curves, other than their common end point. A drawing is *plane* if there are no crossings. *Plane embeddings* form equivalence classes over plane drawings, in that they only define the cyclic order of the edges around their incident vertices (and, if desired, the choice of the outer, infinite face). A *planar* graph is one that admits a plane embedding. A *plane* graph is an embedded graph, i.e., a planar graph together with a plane embedding.

Given a plane embedding G_0 of G, we define its geometric *dual* G_0^* as the embedded multigraph that has a (dual) vertex for each face in G_0; (dual) vertices are joined by a (dual) edge for each (primal) edge shared by their respective (primal) faces. The cyclic order of the (dual) edges around any common incident (dual) vertex v^*, is induced by the cyclic order of the (primal) edges around the (primal) face corresponding to v^*.

We refer to a path/walk in G_0^* as a *dual path/walk* in G_0, and we speak about a *dual path/walk π in G_0 between vertices u, v* if the π starts in a face incident with u and ends in a face incident with v. We shortly say a *route from u to v* (a *u–v route*) to mean a dual walk between vertices u, v (recall that a walk, unlike a path, may repeat vertices and edges).

Crossing numbers and edge insertion. Given a drawing D of G, let $\mathrm{cr}(D)$ denote the number of pairwise edge crossings in D. The *crossing number* problem asks for a drawing D° of a given graph G with the least possible number $\mathrm{cr}(D^\circ) =: \mathrm{cr}(G)$. By saying "pairwise edge crossings" we emphasize that we count a crossing point x separately for every pair of edges meeting in x (e.g., ℓ edges meeting in x give $\binom{\ell}{2}$ crossings).

It is well established that the search for optimal solutions to the crossing number problems can be restricted to so-called *good* drawings: any pair of edges crosses at most once, adjacent edges do not cross, and there is no point that is a crossing of three or more edges.

In this paper we especially consider the following variant of the crossing number problem:

▶ **Definition 2** (Multiple edge insertion, Rigid MEI and MEI).
Consider a planar, connected graph G and a set of edges (vertex pairs, in fact) F not in $E(G)$. We denote by $G + F$ the graph obtained by adding F to the edge set of G.

Let G_0 be a plane embedding of G. The *Rigid Multiple Edge Insertion* problem, denoted by r-MEI(G_0, F), is to find a drawing D of the graph $G + F$ with minimal $\mathrm{cr}(D)$ such that the restriction of D to G is the plane embedding G_0. The attained number of crossings is denoted by r-ins(G_0, F).

The *Multiple Edge Insertion* problem MEI(G, F) is to find an embedding G_1 of G (together with the subsequent drawing D as above), for which r-MEI(G_1, F) attains the minimum number of crossings. The latter is denoted by $\mathrm{ins}(G, F)$.

As mentioned above, a solution of a MEI(G, F) instance – which is a trivial upper bound on $\mathrm{cr}(G+F)$, readily gives an approximate solution of the crossing number problem of $G+F$ by the following inequality:

▶ **Theorem 3** (see [12]). *Consider a planar graph G and a set of edges F not in $E(G)$. Then* $\mathrm{ins}(G, F) \leq |F| \cdot \Delta(G) \cdot \mathrm{cr}(G+F) + \binom{|F|}{2}$, *where $\Delta(G)$ is the maximum degree in G.*

3 Rigid MEI

We first give an FPT algorithm for solving the rigid version r-MEI(G, F), parameterized by $k = |F|$. G is hence a plane graph (i.e., with a fixed embedding) throughout this section.

We first illustrate the simple cases. Solving r-MEI$(G, \{uv\})$, the fixed embedding edge insertion problem with $k = 1$, is trivial. Augment dual G^* with edges of length 0 between the terminals u, v and their respective incident faces (vertices in G^*), to suit the definition of a u–v route in G. Then, simply compute the shortest u–v route in this graph. *Realizing* a route for uv means to draw uv along it within G. If the shortest route has length ℓ, realizing it attains r-ins$(G, \{vw\}) = \ell$, the smallest number of crossings in the Rigid MEI setting.

For $k \geq 2$, the situation starts to be more interesting: not every collection of shortest routes gives rise to an optimal solution of r-MEI(G, F) since there might arise crossings between edges of F. While for $k = 2$, the only question is whether some pair of shortest routes of the two edges in F can avoid crossing each other, for larger values of k we can encounter situations in which all the optimal solutions of r-MEI(G, F) draw some edges of F quite far from their individual shortest routes (in order to avoid crossings with other edges of F), and a more clever approach is needed.

On a very high level, our approach to finding a drawing D of $G + F$ that is an optimal solution to r-MEI(G, F), can be described as follows:

1. We guess, for each pair $f, f' \in F$, whether f and f' will cross each other in D. Since $k = |F|$ is a parameter, all the possibilities can be enumerated in FPT time.
2. Let $X \subseteq \binom{F}{2}$ be a (guessed) set of pairs of edges of F. We find a collection of shortest routes for the edges of F in G under the restriction that exactly the pairs in X cross; Drawing D_X is obtained by inserting the edges of F along their computed routes. As we will see, we may restrict our attention to routes pairwise crossing at most once.
3. We select $D := D_X$ which minimizes the sum of $|X|$ and of the lengths of the routes.

3.1 Handling path homotopy of routes

Obviously, the core task of the scheme (1)–(3) is to solve the point (2) of finding a collection of shortest routes under the restriction that every route avoids crossing certain other routes (note; none of these routes are fixed in advance). The key to this is the concept of *path homotopy* in the plane with point obstacles. Due to the nature of our arguments, in this paper we choose to deal with path homotopy in a combinatorial setting of T-sequences as in Definition 5. This new setting is closely inspired by the discrete-geometry view of boundary-triangulated 2-manifolds by Hershberger and Snoeyink [23]:

First, we "triangulate" the point set $V(F)$ (our obstacles) using transversing paths in the embedding G. A *transversing path* between vertices x, y of G is a path whose ends are x, y and whose internal vertices subdivide some edges of G. Let T be the union of these transversing paths and G' denote the corresponding subdivision of G. In order to avoid a terminology clash with graph triangulations, we will call T in the pair (G', T) a *trinet of* G. This is illustrated in Figure 1 and formally given here (where $V(F) = N$):

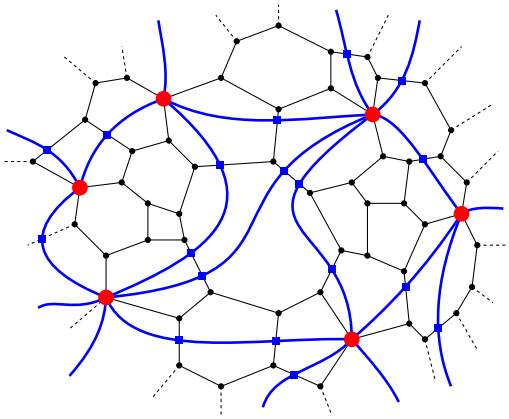


Figure 1 An example of a trinet of a plane graph G (see Definition 4): underlying G is in thin black, the trinodes in red, and the triedges in blue with square blue nodes subdividing edges of G.

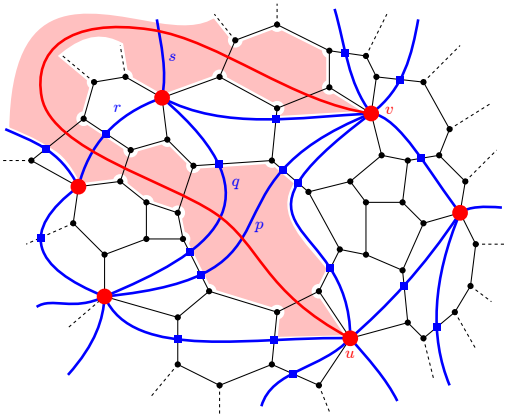


Figure 2 An example (see Definition 5): the T-sequence of the u–v route depicted in red is (p, q, r, s) from u to v. It is a proper T-sequence from u to v (Definition 7); its corresponding alley is shaded in light red.

▶ **Definition 4** (Trinet)**.** Let G be a connected plane graph and $N \subseteq V(G)$, $|N| \geq 4$. A plane graph T such that $V(T) \cap V(G) = N$ is called a *trinet* of G if the following holds:

(a) T is a subdivision of a 3-connected plane triangulation on the vertex set N (in particular, every face of T is incident with precisely three vertices of N), and

(b) there exists a subdivision G' of G such that $V(G') \setminus V(G) = V(T) \setminus N$, $E(G') \cap E(T) = \emptyset$ and the union $G' \cup T$ is a plane embedding.

Pair (G', T) is a *full trinet* of G. The vertices in $N(T) := N$ are called *trinodes* of T, the maximal paths in T internally disjoint from N are *triedges* and their set is denoted by $I(T)$, and the faces of T are *tricells*. Note that the triedges of T are transversing paths of G.

We need to introduce terms related to (and describing) a path homotopy in a full trinet (G', T) of a plane graph G. See also Figure 2 for an illustration of this definition.

▶ **Definition 5** (Alley and T-sequence)**.** Let (G', T) be a full trinet of a plane graph G. Consider a route π between $u, v \in V(G)$ in the graph $G' \cup T$. Then $V(\pi) = \{\phi_0, \phi_1, \ldots, \phi_m\}$ where each dual vertex ϕ_i of π is an open face of $G' \cup T$. Let these faces $(\phi_0, \phi_1, \ldots, \phi_m)$ be ordered along π such that ϕ_0 is incident to u and ϕ_m incident to v. Let $(e_1, e_2, \ldots, e_m) \subseteq E(G' \cup T)$ be the sequence of the primal edges of the dual edges of π, ordered from ϕ_0 to ϕ_m. As a point set, each edge e_i is considered without the endpoints.

(a) The union $\{u, v\} \cup \bigcup_{i=0}^{m} \phi_i \cup \bigcup_{i=1}^{m} e_i$ is called the *alley of* π (or, an *alley between* u, v).

(b) Let $(e'_1, \ldots, e'_\ell) \subseteq (e_1, e_2, \ldots, e_m)$ be the restriction to $E(T)$, and let $(p_1, p_2, \ldots, p_\ell) \subseteq I(T)$ be the sequence of triedges such that p_i contains the edge e'_i for $i = 1, \ldots, \ell$. Then $(p_1, p_2, \ldots, p_\ell)$ is called the *T-sequence of* π from u to v (or, of the corresponding alley from u to v).

The purpose of introducing an alley is to describe a topological corridor for all u–v arcs of a similar kind (and same number of crossings) in the embedding $G' \cup T$. The correspondence is clear: a route π *crosses* a triedge p if the alley of π contains one of the G'-edges forming p. The T-sequence of π hence describes the unique order (with repetition) in which π crosses the triedges of T. Usually, we shall consider only the case of $u, v \in N(T)$.

A route may, in general, cross the same triedge many times (even in one place), but we aim to prove that for "reasonable" routes there is an explicit upper bound; see Lemma 9.

3.2 Refined approach to Rigid MEI

We slightly generalize the shortest route setting to allow for a connected plane graph G with *integer edge weights* $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$. For a full trinet (G', T) of G, we define the edge weights of $G' \cup T$ as follows: $w'(p) := 0$ for all $p \in E(T)$ and $w'(e') := w(e)$ where $e' \in E(G')$ is obtained by subdividing $e \in E(G)$. This w' is the *weight induced by* w in the trinet (G', T). We give the same weights w' also to the edges of the geometric dual of $G' \cup T$. If α is the alley of a route π between vertices x, y in $G' \cup T$, then the *length of* α equals the length of π, i.e., the sum of the w'-weights of the dual edges of π.

For any weighted graph G' with $w\colon E(G') \to \mathbb{N}_+ \cup \{\infty\}$, as above, we correspondingly generalize the notion of a crossing number to *weighted crossing number* as follows: a crossing between two edges $e_1, e_2 \in E(G')$ accounts for the amount of $w(e_1) \cdot w(e_2)$ in $\mathrm{cr}(G')$. For MEI(G, F) problem variants with weighted G, we shall always assume that the weight of each edge $f \in F$ is $w(f) = 1$, and so the (sum of) weighted crossings of f in a drawing of $G + f$ are naturally determined by the weights of the edges crossed by f.

With the help of the framework developed in the previous section, we can now give an (again informal) high-level refinement of our solution steps of r-MEI(G, F) as follows:

4. Consider a trinet T of G on the trinodes $V(F)$. If we fix a (realizable) T-sequence S, then we can use established tools, namely an adaptation of the idea of the funnel algorithm [7, 31], to efficiently compute a shortest alley among those having the same T-sequence S. For $uv \in F$, if we compute an alley α between u, v of length ℓ, then we can easily draw the new edge uv as an arc in α with ℓ weighted crossings in G.
5. Suppose that, for $i = 1, 2$, α_i is a shortest alley between x_i and y_i having the T-sequence S_i. Then, as detailed later in Lemma 14, we can decide from *only* S_1, S_2 whether there exist arcs from x_1 to y_1 in α_1 and from x_2 to y_2 in α_2, which do not cross (note that $\alpha_1 \cap \alpha_2$ may be nonempty and yet there may exist such a pair of non-crossing arcs). Moreover, if the two arcs cross then it should be only once.
6. Consequently, it will be enough to loop through all "suitable" T-sequences for every edge of F and independently perform the steps (4), (5) for each combination of them, in order to get an optimal solution of r-MEI(G, F) as in (3). The point is to bound the number of considered T-sequences in terms of only the parameter $k = |F|$.

3.3 T-sequences of potential shortest routes

Considering the outline (4)–(6), we first resolve the last point which is a purely mathematical question. In order to achieve the goal, we shall build a special trinet of G along shortest dual paths between the trinodes in G (Definition 6), and then we will be able to restrict our attention to special T-sequences (Definition 7) of bounded length. The latter is formulated in Lemma 9 whose proof presents the main piece of technical work in this paper.

▶ **Definition 6** (Shortest-spanning trinet)**.** Let (G', T) be a full trinet of a plane graph G, and let the weights w' in (G', T) be induced by weights w in G. For a triedge $q \in I(T)$, every internal vertex t of q is incident with two edges e, e' of G' of weight $w'(e) = w'(e')$ which we call the weight of t. The *transversing weight of* q equals the sum of the weights of the internal vertices of q.

A triedge $q \in I(T)$ between trinodes x, y is *locally-shortest* if the transversing weight of q is equal to the length of a shortest dual path π in $G' \cup T$ between x, y, such that π is contained in(!) the union of the two tricells incident with q (including the points of q itself). Similarly, q is *globally-shortest* if the transversing weight of q is equal to the dual distance between x, y in $G' \cup T$.

We say that T has the *shortest-spanning property* if every triedge in $I(T)$ is locally-shortest, and there exists a subset of triedges $J \subseteq I(T)$ forming a connected subgraph of T spanning all the trinodes such that every triedge in J is globally-shortest.

▶ **Definition 7** (Proper T-sequence)**.** Consider a trinet T and trinodes $u \neq v \in N(T)$. A *nonempty* sequence $S = (p_1, p_2, \ldots, p_m) \subseteq I(T)$ of triedges of T (repetition allowed) is a *proper T-sequence from u to v* if the following holds: u is disjoint from p_1 but there exists a tricell θ_0 incident with both u and p_1, v is disjoint from p_m but there exists a tricell θ_m incident with both v and p_m, and each two consecutive triedges p_i, p_{i+1} are distinct and incident to a common tricell θ_i for $1 \leq i < m$. Finally, the *empty* sequence $S = \emptyset$ is considered a proper T-sequence from u to v if u, v are incident to a common tricell θ_0.

Since T is a subdivision of a graph triangulation and u, v are nodes of this triangulation, we immediately obtain the following fact complementing Definition 7:

▶ **Claim 8.** *For every proper nonempty T-sequence S, the sequence of tricells $(\theta_0, \theta_1, \ldots, \theta_m)$ as in Definition 7 is uniquely determined by T and S. If $S = \emptyset$, then uv is a triedge of T and there are two choices of θ_0 incident with uv; we simply make an arbitrary deterministic choice of θ_0 among those two in each case.*

Now the main technical finding, necessary for our algorithm, comes in:

▶ **Lemma 9 (*).** *Consider an instance* r-MEI(G, F) *where G is a connected plane graph. Let (G', T) be a full trinet of G having the shortest-spanning property. There exists a set $\{\pi_f : f \in F\}$ where π_f for $f = uv$ is a route in $G' \cup T$ between the trinodes u, v, such that the following hold:*

- **(a)** *There exists an optimal drawing D of $G + F$ with* r-ins(G, F) *crossings such that each edge $f \in F$ is drawn in the alley of π_f, and no two edges of F cross each other more than once.*
- **(b)** *The T-sequence S_f of each π_f is a proper T-sequence, and no triedge occurs in S_f more than $8k^4$ times where $k = |F|$.*

The full proof of Lemma 9 is rather long and cannot fit into the restricted short paper, but due to great importance of the statement we at least provide here a brief informal sketch:

- We show by local modifications of the routes for F that there exists an optimal solution which fulfills (a) and determines only proper T-sequences in the considered trinet.
- If some triedge repeats too many times in the T-sequence of some edge $f \in F$, then also one of the globally-shortest triedges of T, say p, repeats many times in this sequence. Hence, by the globally-shortest property, re-routing f partly "along" p does not increase crossings of f with $E(G)$. Though, there may be no strict improvement either. Even worse, re-routing of f may incur new crossings with F which is difficult to handle locally.
- A careful analysis of the situations in which the T-sequence of f crosses many times the same globally-shortest triedge of T then gives the desired conclusion; that either the local situation contradicts optimality of the whole solution, or a strict improvement (in terms of the lengths of T-sequences) can be achieved by a local move in the drawing.

3.4 Shortest routes in a sleeve

Next, we consider point (4) of the above outline. To recapitulate, for trinodes u, v of a trinet T of G and a given proper T-sequence S from u to v, the task is to find a shortest route from u to v among those having the same T-sequence S (and independently of other

searched routes). Since we cannot, in general, avoid repeating triedges in S and tricells in the sequence $(\theta_0, \theta_1, \ldots, \theta_m)$ in Definition 7, we use a similar workaround as in [23]; "lifting" the respective sequence of tricells into a universal cover as follows.

▶ **Definition 10** (Sleeve of a T-sequence)**.** Let (G', T) be a full trinet of a plane graph G, and consider a proper T-sequence $S = (p_1, p_2, \ldots, p_m)$ from u to v determining the sequence of tricells $(\theta_0, \theta_1, \ldots, \theta_m)$ by Claim 8. For $i = 0, 1, \ldots, m$, let L_i be a disjoint copy of the embedded subgraph of $G' \cup T$ induced by θ_i. Construct a plane graph L from the union $L_0 \cup \cdots \cup L_m$ by identifying, for $j = 1, \ldots, m$, the copy of the triedge p_j in L_{j-1} with the copy of p_j in L_j. We call L the *sleeve of* S in the trinet (G', T), and we identify u and v with their copies in L_0 and L_m, respectively. We make the unique face of L that is not covered by a copy of any tricell of T the *outer face* of L.

Observe that every route from u to v in $G' \cup T$ having its T-sequence equal to S can be easily lifted into a corresponding u–v route in the sleeve L of S. Conversely, any u–v route in L avoiding the outer face and crossing the copies of triedges in L at most once each, can be obviously projected down to $G' \cup T$ to make a route with the T-sequence equal to S. In fact, we can routinely prove that some shortest u–v route in L must be of the latter kind, under the shortest-spanning property (cf. Definition 6).

▶ **Lemma 11 (*).** *Let (G', T) be a shortest-spanning full trinet of an edge-weighted plane graph G, S a proper T-sequence between trinodes u, v of T, and let L be the sleeve of S. Let ℓ be the length of a shortest route from u to v among those having the T-sequence S. Then, ℓ is equal to the dual distance from u to v in L without the outer face, and at least one of the u–v routes of length ℓ in L crosses the copy of each triedge from S in L exactly once.*

Hence we can straightforwardly compute desired shortest routes using established linear time shortest path algorithms, such as the algorithm of Klein et al. [29] since L is planar, or Thorup's algorithms [33] since we have integral weights. In fact, since we are fine with edge weights of G given in unary, a simple adaptation of BFS can do the job for us, too.

▶ **Corollary 12 (*).** *Let (G', T) be a shortest-spanning full trinet of a plane graph G with integer edge weights, and S a proper T-sequence between trinodes u, v of T. A shortest u–v route among those having the T-sequence S can be found in $\mathcal{O}(|S| \cdot |N(T)| \cdot |V(G)|)$ time.*

Observe that in our case, by Lemma 9, we have $|S| \cdot |N(T)| \leq (6k \cdot 8k^4) \cdot 2k = 96k^6$.

3.5 Crossing of routes

Finally, it remains to address point (5). Consider a 4-tuple of distinct trinodes u, v, u', v'. Let π be a u–v route and π' be a u'–v' route. We say that an *arc b follows the route π* if b is contained in the alley of π and b intersects the faces forming the alley exactly in the order given by π (recall that a route is technically a dual walk and hence, possibly, some face might repeat in π). We say that the pair of routes π, π' is *non-crossing*, if there exist a u–v arc b following π and a u'–v' arc b' following π' such that $b \cap b' = \emptyset$. In order to characterize possible non-crossing pairs of routes in terms of their T-sequences, we bring the following:

▶ **Definition 13** (Crossing certificate)**.** Let (G', T) be a full trinet of a plane graph G, and let π be a route from u to v and π' be a route from u' to v' in $G' \cup T$, where u, v, u', v' are distinct trinodes of T. Assume the T-sequences $S = (p_1, \ldots, p_n)$ of π and $S' = (p'_1, \ldots, p'_\ell)$ of π' are proper and let $(\theta_0, \ldots, \theta_n)$ and $(\theta'_0, \ldots, \theta'_\ell)$ be their tricell sequences by Claim 8. For technical reasons, let $p_0 := u$, $p_{n+1} := v$ and $p'_0 := u'$, $p'_{\ell+1} := v'$.

A *crossing certificate for* S, S' is a triple of indices (c, d, m) where $c, d, m \geq 0$, $c + m \leq n$, $d + m \leq \ell$, such that the following holds:

(a) $\theta_{c+j} = \theta'_{d+j}$ for $0 \leq j \leq m$, but $p_c \neq p'_d$ and $p_{c+m+1} \neq p'_{d+m+1}$,

(b) the triple p_c, p'_d, p_{c+1} occurs around the tricell $\theta_c = \theta'_d$ in the same cyclic orientation as the triple $p_{c+m+1}, p'_{d+m+1}, p_{c+m}$ occurs around $\theta_{c+m} = \theta'_{d+m}$.

Furthermore, a crossing certificate for the same sequence S and the reversal of S' from v' to u' is also called a crossing certificate for S, S'.

Definition 13 deserves a closer explanation. Assume that a crossing certificate satisfies $0 < c < n$ and $0 < d < \ell$. Then all four elements $p_c, p'_d, p_{c+1}, p'_{d+1}$ are triedges of the same tricell $\theta_c = \theta'_d$, and since $p_{c+1} \neq p_c \neq p'_d \neq p'_{d+1}$, we get $p_{c+1} = p'_{d+1}$. Hence $m > 0$ and the situation is such that S and S' "merge" at θ_c where (up to symmetry) S comes on the left of S', and they again "split" at θ_{c+m} where S leaves on the right of S', thereby "crossing it". The full definition, though, covers also the boundary cases of crossing certificates for which $c \in \{0, n\}$ or $d \in \{0, \ell\}$ (or both), and when S and S' may have no triedge in common; those can be easily examined case by case.

▶ **Lemma 14 (*).** *Let (G', T) be a full trinet of an edge-weighted plane graph G, and u_i, v_i, $i = 1, 2$, be four distinct trinodes. Assume that S_i from u_i to v_i are proper T-sequences. In $G' \cup T$, for $i = 1, 2$, there exist routes π_i from u_i to v_i having the the T-sequence S_i, such that π_1, π_2* are non-crossing*, if and only if there exists no crossing certificate for S_1, S_2.*

3.6 Summary of the r-MEI algorithm

We can now summarize the overall algorithm to solve Rigid MEI, see Algorithm 1. Based thereon, together with Lemmas 9, 11, Corollary 12, and Lemma 14 we obtain:

▶ **Theorem 15 (*).** *Let G be a connected plane graph with edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, and F a set of $k \geq 1$ new edges (vertex pairs, in fact) such that $w(f) = 1$ for all $f \in F$. Algorithm 1 finds an optimal solution to the w-weighted* r-MEI(G, F) *problem, if a finite solution exists, in time $\mathcal{O}\big(2^{p(k)} \cdot |V(G)|\big)$, where $p(k)$ is a polynomial function in k.*

In fact, one can see that $p(k) = \mathcal{O}(k^6 \log k)$ in Theorem 15.

4 General MEI

Now, we turn our attention to the general MEI(G, F) problem, in which the embedding of a planar graph G is not pre-specified. Recall that triconnected planar graphs have a unique embedding (up to mirroring), but already biconnected graphs can have an exponential number of embeddings in general. As it is commonly done in insertion problems since [22], we will use the *SPR-tree* datastructure (sometimes also known as SPQR-tree) to encode and work with all these possible embeddings. The structure was first defined in a slightly different form in [16], based on prior work of [3, 34]. It can be constructed in linear time [27, 21] and only requires linear space.

▶ **Definition 16** (SPR-tree, cf. [8])**.** Let G be a biconnected graph with at least three vertices. The *SPR-tree* $\mathcal{T}$ of G is the unique smallest tree satisfying the following properties:

(a) Each node ν in $\mathcal{T}$ holds a specific (small) graph $S_\nu = (V_\nu, E_\nu)$, with $V_\nu \subseteq V(G)$, called a *skeleton*. Each edge e of E_ν is either a *real* edge $e \in E(G)$, or a *virtual* edge $e = xy \notin E(G)$ (while still, $x, y \in V(G)$).

In: a plane graph G, edge weights $w\colon E(G) \to \mathbb{N}_+ \cup \{\infty\}$, new edge set F s.t. $w(f) = 1$ for $f \in F$.
Out: an optimal solution to (w-weighted) r-MEI(G, F).

1. Compute a full trinet (G', T), $N(T) := V(F)$, with the shortest-spanning property of T.
   - **a.** Pick any trinode $x \in N(T)$ and greedily compute globally-shortest triedges (Def. 6) from x to all other trinodes, using a simple shortest path computation.
   - **b.** The remaining triedges can be greedily computed as locally-shortest, one after another.
2. For each $f = uv \in F$:
   - **a.** Compute $\mathcal{S}_f$ as the set of all its possible and relevant proper T-sequences from u to v. The size of $\mathcal{S}_f$ is bounded due to Lemma 9(b) by $2^{s(|F|)}$ where $s(k) = \mathcal{O}(k^5 \log k)$.
   - **b.** For each $S \in \mathcal{S}_f$, compute a shortest u–v route π_S in $G' \cup T$ among those having the T-sequence S (where the length function is induced by w), using Corollary 12.
3. For each possible system of representatives $\mathcal{P} = \{S_f\}_{f \in F}$ with $S_f \in \mathcal{S}_f$:
   - **a.** Check, for each pair $f, f' \in F$, whether there exists a crossing certificate for $S_f, S_{f'}$ (e.g., using brute force by Def. 13).
     Let $X_\mathcal{P}$ be the set of pairs $\{f, f'\}$ for which such a certificate has been found.
   - **b.** If any pair $\{f, f'\} \in X_\mathcal{P}$ requires more than a single crossing (which can be found by checking again for two "independent" (✱) crossing certificates of $S_f, S_{f'}$), let $\mathrm{cr}_\mathcal{P} := \infty$.
   - **c.** Otherwise, let $\mathrm{cr}_\mathcal{P} := |X_\mathcal{P}| + \sum_{f \in F} len_w(\pi_{S_f})$, where π_{S_f} is the shortest route for f and S_f computed in step (2) and $len_w(\pi_{S_f})$ is the length.
4. Among all $\mathcal{P}$ considered in (3), pick one with smallest $\mathrm{cr}_\mathcal{P} < \infty$. Let this be $\mathcal{P}^\circ = \{S_f^\circ\}_{f \in F}$.
5. In the plane graph G, realize each edge $f \in F$ following its respective route $\pi_{S_f^\circ}$, such that the overall resulting weighted number of crossings is $\mathrm{cr}_{\mathcal{P}^\circ}$:
   - **a.** By the minimality setup in (4), no $\pi_{S_f^\circ}$ is self-intersecting.
   - **b.** A standard postprocessing argument – removing consecutive crossings between any pair f, f' (within a section of $S_f^\circ \cap S_{f'}^\circ$) by re-routing f' partially along f – prevents multiple crossings in the pairs from $X_\mathcal{P}$ and makes the pairs of F not in $X_\mathcal{P}$ crossing-free.

Algorithm 1 Algorithm to solve the (weighted) Rigid MEI problem.

(b) $\mathcal{T}$ has three different node types with the following skeleton structures: **(S)** S_ν is a simple cycle; **(P)** S_ν consists of two vertices and at least three multiple edges between them; (**R**) S_ν is a simple triconnected graph on at least four vertices.

(c) For every edge $\nu\mu$ in $\mathcal{T}$ we have $|V_\nu \cap V_\mu| = 2$. These two common vertices, say x, y, form a vertex 2-cut in G. The skeleton S_ν contains a specific virtual edge $e_\mu \in E(S_\nu)$ that represents the node μ and, symmetrically, some specific $e_\nu \in E(S_\mu)$ represents ν. Both e_ν, e_μ have the ends x, y; the two virtual edges may refer to one another as *twins*.

(d) For an edge $\nu\mu \in E(\mathcal{T})$, let $e_\mu \in E_\nu$, $e_\nu \in E_\mu$ be the pair of virtual edges as in (c) connecting the same x, y. The operation of *merging* at $\nu\mu$ creates a graph $(S_\nu \cup S_\mu) - \{e_\mu, e_\nu\}$ obtained by gluing the two skeletons S_ν, S_μ at x, y and removing e_μ, e_ν.
Consider the tree $\mathcal{T}$ rooted at any node. For $\nu \in V(\mathcal{T})$ we define the *pertinent graph* P_ν of ν by recursively merging the skeletons at every tree-edge of the subtree rooted at ν, and removing the parent virtual edge of ν (if not the root itself). Then the pertinent graph of the root of $\mathcal{T}$ is G.

We again start with an illustration of the "simple" case of $|F| = 1$. The central theorem of [22] states that an optimal solution of MEI($G, \{uv\}$) for biconnected G can be obtained by looking only at the shortest path in the SPR-tree $\mathcal{T}$ of G between a node whose skeleton contains u and a node whose skeleton contains v. For each skeleton S_ν along this path, one simply finds an optimal embedding and a shortest partial route in this embedding between the virtual edge representing u (or u itself) and the virtual edge representing v (or v itself).

In the case of S- and P-nodes this route requires no crossings. For an R-node, one is looking for a shortest route in the rigid setting. Thereby, each virtual edge e_μ in S_ν with ends x, y is assigned its *pertinent weight* $w(e_\mu)$: the size of a minimum x–y edge-cut in the pertinent graph P_μ. See [22] for more details.

We return to the general MEI(G, F) problem for biconnected planar graphs G. Considering an arbitrarily rooted SPR-tree $\mathcal{T}$ of G, we devise a dynamic programming procedure to solve MEI bottom-up over $\mathcal{T}$. The core is to describe which subproblems we are going to solve at each node ν of $\mathcal{T}$, assuming we know the solutions of the corresponding subproblems at the child nodes of ν. For better understanding, this task is illustrated and described in Figure 3. We say a virtual edge e_μ in S_ν representing a child node μ of ν, is *dirty* if its pertinent graph P_μ contains an end vertex of $f \in F$ other than one of the ends of e_μ.

On a high level, we describe our procedure as follows:

▶ **Algorithm 17.** A dynamic programming scheme to solve MEI bottom-up over an SPR-tree.

1. We compute the pertinent weights $w(e_\mu)$ for all non-dirty virtual edges that appear in skeletons used in the following step (2).
2. For each dirty child virtual edge e_μ in the skeleton S_ν, we assume to know the optimal number of crossings for every "reasonable" scheme of routing new edges of F to, from, or across e_μ (we stress that we speak about *all* edges of F, including those not having an end in P_μ). We exhaustively process, over all possible embeddings of S_ν in case of a P-node, all the subproblems stated by every admissible combination of routing schemes for the dirty child virtual edges in S_ν and for the parent virtual edge of S_ν. For each "reasonable" routing scheme of the parent virtual edge, we then select and store the best obtained solution in terms of minimal overall crossing number.
3. Each particular subproblem of the exhaustive processing in (2) can be formulated as a weighted r-MEI(H, F'') instance. The plane graph H is constructed from S_ν by inserting special gadgets at the dirty virtual edges (see the colored digons in Figure 3), and the new edges F'' are suitable segments of the edges of F (their ends are either an end of the original F-edge or a vertex on such a gadget). All non-virtual edges of S_ν and all of F'' have weight 1. Each non-dirty virtual edge e_μ gets weight $w(e_\mu)$, computed in (1), and all dirty virtual edges get weight ∞. See Figure 3 for closer details.
   This r-MEI(H, F'') instance can then be solved using Theorem 15.
4. If ν is the root node of $\mathcal{T}$, then there is no parent virtual edge and the exhaustive processing in (2) selects a single optimal solution. Realizing this solution, together with the corresponding subsolutions at the descendants, gives an optimal solution to the original MEI(G, F) instance. ◀

We first give a more precise definition regarding step (2). For a dirty virtual edge $e_\mu = xy$ in the skeleton S_ν with its pertinent graph P_μ, let M be the set of those ends of the edges of F that belong to $V(P_\mu) \setminus \{x, y\}$. A *routing scheme* of e_μ is an arbitrary pair of disjoint sequences Q_1, Q_2, where $(Q_1 \cup Q_2) \cap V(G) = \emptyset$, together with a perfect matching on the set $M \cup Q_1 \cup Q_2$ (the cardinality of which must be even). The matching edges are meant to represent segments of edges of F drawn across the pertinent graph P_μ, but on this level we do not need to distinguish which segment belongs to which of the F-edges.

If φ is the parent node of ν then a routing scheme of e_φ in S_ν, as dealt with in step (2), is formally not the same as the corresponding routing scheme of the twin virtual edge e_ν in S_φ; these two are of course actually "complementary". We neglect this technical difference in a high-level description of our dynamic programming scheme.

Second, we informally outline three claims which make Algorithm 17 run in FPT time:

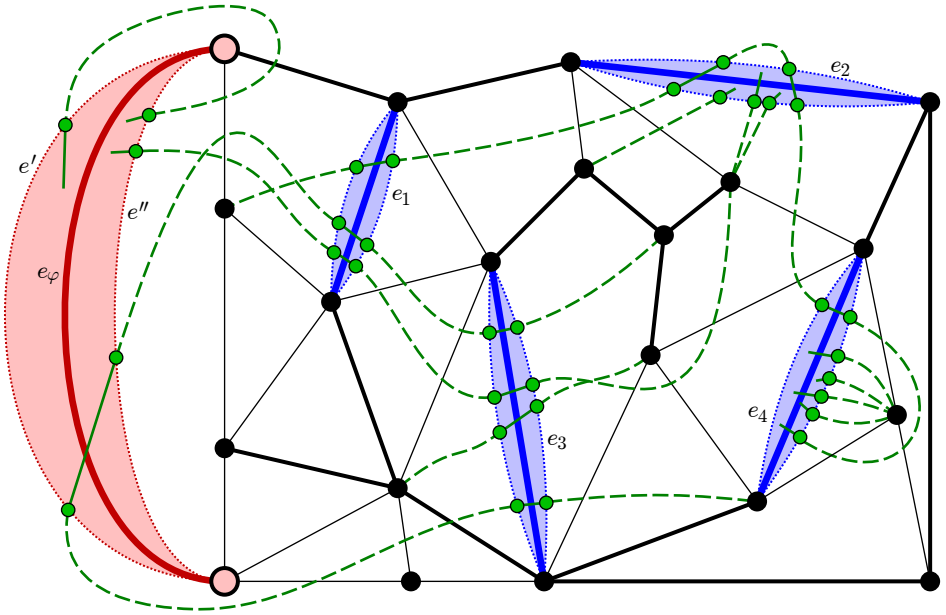


Figure 3 One of the possible r-MEI instances considered at a rigid R-node ν in Algorithm 17. The mostly black graph shows the skeleton S_ν, whereby the red virtual edge e_φ corresponds to ν's parent, the blue virtual edges $e_1, \ldots, e_4$ to ν's dirty children, and the thick black edges to the non-dirty children with their implicit pertinent weights. The new edges of F are depicted in green. For each dirty virtual edge e_i, including e_φ, we consider a fixed (by the processing in step (2)) *routing scheme* that externally encodes at which points segments of edges of F enter and leave the pertinent subgraph of e_i. This is depicted with a blue or red digon of e_i and solid green segments of the F-edges within it. Consequently, we do not have to care for internal details of the digonal parts (they are, in fact, not part of our r-MEI instance and only visualized for the reader's context), as they are either already resolved in the subproblems of the children or are going to be resolved in the ancestor nodes. Likewise, for an r-MEI instance at ν, the only relevant information for any non-dirty virtual edge is its pertinent weight and no subembedding details are necessary.
The *gadget* (see step (3)) used to enforce the aforementioned routing schemes at every dirty virtual edge e_i, is composed of the edge e_i itself (solid blue or red), and two new edges e_i', e_i'' parallel to e_i (dotted) which are subdivided by the prescribed entry/exit *terminals* (green dots) of the segments of F-edges. The gadget edges are weighted ∞ and their embedding is rigid. Possible end vertices of F-edges in $V(S_\nu)$ are also counted as the terminals. By a combination of the routing schemes we mean an arbitrary perfect matching (the green dashed segments) on these terminals. It is an *admissible combination* (cf. step (2)) if the union of the prescribed solid green and the matching dashed green segments forms an actual "realization" of the new edges of F.
An admissible combination then gives rise to the depicted r-MEI instance, as described in step (3), which can be solved using Theorem 15.

- A skeleton S_ν may contain arbitrarily more than k virtual edges, but only at most $2k$ of them may be dirty (containing an end vertex of F). The computation of the pertinent weights of non-dirty virtual edges can be done in linear overall time.
- A P-node skeleton S_ν may have an unbounded number of inequivalent embeddings (precisely $(q-1)!$ where q is the number of parallel edges in S_ν). However, if two *non-dirty* edges e_1, e_2 appear non-consecutively within the embedding of S_ν, there exists an alternative solution with the same number of crossings where e_2 is rerouted alongside e_1 (by optimality, both edges encounter the same number of crossings either way). Thus we

can restrict our attention to only those at most $(2k)!$ embeddings in which all non-dirty virtual edges are consecutive (in any internal order) within the embedding of S_ν.

- In contrast to the single edge insertion, an edge of F may easily be forced to cross the same virtual edge e_μ multiple times (a number depending only on k). However, the complexity of any "reasonable" routing of the edges of F across e_μ is bounded by a function of $k = |F|$, as shown in Lemma 9.

As a corollary we see that the number of subproblems generated in step (2), as well as the amount of information stored at any SPR-tree node, are bounded by a function of k.

The solution of each one subproblem in step (3) can be obtained using the algorithm for r-MEI in Theorem 15 in linear time for constant k, and there are at most $\mathcal{O}(|V(G)|)$ nodes in the tree $\mathcal{T}$. Instead of the naïve quadratic runtime bound, we even achieve a linear overall runtime bound by observing that the union of all skeletons is still only of $\mathcal{O}(|V(G)|)$ size. We obtain, as given in the introduction:

▶ **Theorem 18** (The biconnected case of Theorem 1 – *). *Let G be a planar biconnected graph and F a set of $k \geq 1$ new edges (vertex pairs, in fact). An optimal solution of the problem* MEI(G, F) *can be found in $\mathcal{O}(2^{q(k)} \cdot |V(G)|)$ time, where $q(k)$ is a polynomial function in k.*

For essentially all known insertion algorithms (in particular the single edge insertion [22], the vertex insertion [10], and the MEI approximation [11]), one typically first resolves the case of biconnected graphs (using SPR-trees as above). Then, it is relatively straightforward to lift the algorithms to connected graphs, by considering BC-trees (see below). Interestingly, this step seems much more complicated in the case of exact MEI.

Consider the well-known *block-cut tree* (BC-tree) decomposing any connected graph into its blocks (biconnected components). Using analogous techniques as in [11], we extend our dynamic programming approach by amalgamating the BC-tree of G with respective SPR-trees of the blocks, to obtain a linear-sized *con-tree* – with an additional node type **C**, for the cut vertices. The outline in Algorithm 17 is completed with the following:

(2⁺). If ν is a C-node representing a cut vertex c of G, then we exhaustively process all possibilities to combine the dirty blocks of G incident to c (while non-dirty blocks can be safely ignored).

Unfortunately, although we can again bound the number of dirty blocks by $2k$, there is now no easy "external description of routing" with respect to a cut vertex available (analogous to a routing scheme of a virtual edge). Consequently, the number of possibilities to consider in (2⁺) depends on k and the degree of the cut vertex c. We conclude:

▶ **Theorem 19** (The connected case of Theorem 1 – *). *Let G be a planar connected graph and F a set of $k \geq 1$ new edges (vertex pairs, in fact). Let Δ_c be the maximum degree over cut vertices of G. Then an optimal solution of the problem* MEI(G, F) *can be found in $\mathcal{O}\big(2^{q'(k)} \cdot \Delta_c^k \cdot |V(G)|\big)$ time, where $q'(k)$ is a polynomial function in k.*

5 Conclusion

In this paper, we have affirmatively answered the long standing open question, floating around ever since [1, 22], whether there is a polynomial-time algorithm to insert a constant-sized set of edges into a planar graph in a crossing minimal way. Previously, this has only been known for single edges; the problem with multiple edges could only be approximated. Our result also induces a new approximation algorithm for the general crossing number of graphs with

bounded number of edges beyond planarity. While the original problem was defined over unweighted graphs, we considered weighted graphs in our subproblems but only to limited extent.

▶ **Open Problem 20.** Is MEI fixed-parameter tractable if both G and F are weighted?

In fact, we can straight-forwardly answer this question affirmatively for weighted G but unweighted F, as should be clear from the above proofs. We may also assume weighted F, if the weights are bounded by k, by simply adding multiple copies of an edge to F. For generally weighted F, we observe that the dynamic programming part of solving MEI would be capable of achieving this feat. However, the required r-MEI subproblems are not, due to a technical rerouting argument in the proof of Lemma 9. It is not easy to circumvent this argument and the problem seems surprisingly related to that of decidability of string graphs – a connection that deserves future investigation.

As our final open question we include:

▶ **Open Problem 21.** Is there a polynomial-time algorithm to solve MEI for simply-connected graphs independent of Δ_c?

Such a dependency on Δ_c does not show up when inserting a single edge or a star into planar G [22, 10]. On the other hand, even more restrictive degree dependencies are very common and seemingly unavoidable in the known general crossing number approximations.

Acknowledgments. We thank Sergio Cabello and Carsten Gutwenger for helpful discussions.

References

1 C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *Journal of Systems and Software*, 4:163–173, 1984.

2 S. N. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984.

3 D. Bienstock and C. L. Monma. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990.

4 S. Cabello. Hardness of approximation for crossing number. *Discrete & Computational Geometry*, 49(2):348–358, 2013.

5 S. Cabello and B. Mohar. Crossing number and weighted crossing number of near-planar graphs. *Algorithmica*, 60(3):484–504, 2011.

6 S. Cabello and B. Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013.

7 B. Chazelle. A theorem on polygon cutting with applications. In *Proc. FOCS'82*, pages 339–349. IEEE Computer Society, 1982.

8 M. Chimani. *Computing Crossing Numbers*. PhD thesis, TU Dortmund, Germany, 2008. URL: `http://hdl.handle.net/2003/25955`.

9 M. Chimani and C. Gutwenger. Advances in the planarization method: effective multiple edge insertions. *J. Graph Algorithms Appl.*, 16(3):729–757, 2012.

10 M. Chimani, C. Gutwenger, P. Mutzel, and C. Wolf. Inserting a vertex into a planar graph. In *Proc. SODA'09*, pages 375–383, 2009.

11 M. Chimani and P. Hliněný. A tighter insertion-based approximation of the crossing number. In *Proc. ICALP'11*, volume 6755 of *LNCS*, pages 122–134. Springer, 2011.

12 M. Chimani, P. Hliněný, and P. Mutzel. Vertex insertion approximates the crossing number for apex graphs. *European Journal of Combinatorics*, 33:326–335, 2012.

13 J. Chuzhoy. An algorithm for the graph crossing number problem. In *Proc. STOC'11*, pages 303–312. ACM, 2011.

14 J. Chuzhoy, Y. Makarychev, and A. Sidiropoulos. On graph crossing number and edge planarization. In *Proc. SODA'11*, pages 1050–1069. ACM Press, 2011.

15 É. Colin de Verdière and A. Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. *ACM Transactions on Algorithms*, 7(2):19, 2011.

16 G. Di Battista and R. Tamassia. On-line planarity testing. *SIAM Journal on Computing*, 25:956–997, 1996.

17 G. Even, S. Guha, and B. Schieber. Improved approximations of crossings in graph drawings and VLSI layout areas. *SIAM J. Comput.*, 32(1):231–252, 2002.

18 M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Alg. Discr. Meth.*, 4:312–316, 1983.

19 I. Gitler, P. Hliněný, J. Leanos, and G. Salazar. The crossing number of a projective graph is quadratic in the face-width. *Electronic Journal of Combinatorics*, 15(1):#R46, 2008.

20 M. Grohe. Computing crossing numbers in quadratic time. *J. Comput. Syst. Sci.*, 68(2):285–302, 2004.

21 C. Gutwenger and P. Mutzel. A linear time implementation of SPQR trees. In *Proc. GD'00*, volume 1984 of *LNCS*, pages 77–90. Springer, 2001.

22 C. Gutwenger, P. Mutzel, and R. Weiskircher. Inserting an edge into a planar graph. *Algorithmica*, 41(4):289–308, 2005.

23 J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom.*, 4:63–97, 1994.

24 P. Hliněný and M. Chimani. Approximating the crossing number of graphs embeddable in any orientable surface. In *Proc. SODA'10*, pages 918–927, 2010.

25 P. Hliněný and G. Salazar. On the crossing number of almost planar graphs. In *Proc. GD'05*, volume 4372 of *LNCS*, pages 162–173. Springer, 2006.

26 P. Hliněný and G. Salazar. Approximating the crossing number of toroidal graphs. In *Proc. ISAAC'07*, volume 4835 of *LNCS*, pages 148–159. Springer, 2007.

27 J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

28 K-I. Kawarabayashi and B. Reed. Computing crossing number in linear time. In *Proc. STOC'07*, pages 382–390, 2007.

29 P. Klein, S. Rao, M. Rauch, and S. Subramanian. Faster shortest-path algorithms for planar graphs. In *Proc. STOC'94*, pages 27–37, 1994.

30 Y. Kobayashi and C. Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7(4):234–245, 2010.

31 D.-T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–410, 1984.

32 M. Schaefer. The graph crossing number and its variants: A survey. *Electronic Journal of Combinatorics*, #DS21, May 15, 2014.

33 M. Thorup. Undirected single source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46:362–394, 1999.

34 W. T. Tutte. *Connectivity in graphs*, volume 15 of *Mathematical Expositions*. University of Toronto Press, 1966.

35 T. Ziegler. *Crossing Minimization in Automatic Graph Drawing*. PhD thesis, Saarland University, Germany, 2001.

Polynomial-Sized Topological Approximations Using the Permutahedron

Aruni Choudhary[1], Michael Kerber[2], and Sharath Raghvendra[3]

1 **Max Planck Institute for Informatics, Saarbrücken, Germany**
`aruni.choudhary@mpi-inf.mpg.de`
2 **Graz University of Technology, Graz, Austria**
`kerber@tugraz.at`
3 **Virginia Tech, Blacksburg, USA**
`sharathr@vt.edu`

Abstract

Classical methods to model topological properties of point clouds, such as the Vietoris-Rips complex, suffer from the combinatorial explosion of complex sizes. We propose a novel technique to approximate a multi-scale filtration of the Rips complex with improved bounds for size: precisely, for n points in $\mathbb{R}^d$, we obtain a $O(d)$-approximation with at most $n2^{O(d \log k)}$ simplices of dimension k or lower. In conjunction with dimension reduction techniques, our approach yields a $O(\text{polylog}(n))$-approximation of size $n^{O(1)}$ for Rips filtrations on arbitrary metric spaces. This result stems from high-dimensional lattice geometry and exploits properties of the permutahedral lattice, a well-studied structure in discrete geometry.

Building on the same geometric concept, we also present a lower bound result on the size of an approximate filtration: we construct a point set for which every $(1+\varepsilon)$-approximation of the Čech filtration has to contain $n^{\Omega(\log \log n)}$ features, provided that $\varepsilon < \frac{1}{\log^{1+c} n}$ for $c \in (0,1)$.

1998 ACM Subject Classification F.2.2 Geometric problems and computation

Keywords and phrases Persistent Homology, Topological Data Analysis, Simplicial Approximation, Permutahedron, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.31

1 Introduction

Motivation and previous work. Topological data analysis aims at finding and reasoning about the underlying topological features of metric spaces. The idea is to represent a data set by a set of discrete structures on a range of scales and to track the evolution of homological features as the scale varies. The theory of *persistent* homology allows for a topological summary, called the *persistence diagram* which summarizes the lifetimes of topological features in the data as the scale under consideration varies monotonously. A major step in the computation of this topological signature is the question of how to compute a *filtration*, that is, a multi-scale representation of a given data set.

For data in the form of finite point clouds, two frequently used constructions are the *(Vietoris-)Rips* complex $\mathcal{R}_\alpha$ and the *Čech* complex $\mathcal{C}_\alpha$ which are defined with respect to a scale parameter $\alpha \geq 0$. Both are simplicial complexes capturing the proximity of points at scale α, with different levels of accuracy. Increasing α from 0 to ∞ yields a nested sequence of simplicial complexes called a *filtration*.

Unfortunately, Rips and Čech filtrations can be uncomfortably large to handle. For homological features in low dimensions, it suffices to consider the k-skeleton of the complex,


© Aruni Choudhary, Michael Kerber, and Sharath Raghvendra;
licensed under Creative Commons License CC-BY

that is, all simplices of dimension at most k. Still, the k-skeleton of Rips and Čech complexes can be as large as n^{k+1} for n points, which is already impractical for small k when n is large. One remedy is to construct an *approximate filtration*, that is, a filtration that yields a similar topological signature as the original filtration but is significantly smaller in size. The notion of "similarity" in this context can be made formal through a distance measure on persistence diagrams. The most frequently used similarity measure is the *bottleneck distance*, which finds correspondences between topological features of two filtrations, such that the lifetimes of each pair of matched features are as close as possible. A related notion is the *log-scale bottleneck distance* which allows a larger discrepancy for larger scales and thus can be seen as a relative approximation, with usual bottleneck distance as its absolute counterpart. We call an approximate filtration a *c-approximation* of the original, if their persistence diagrams have log-scale bottleneck distance at most c.

Sheehy [23] gave the first such approximate filtration for Rips complexes with a formal guarantee. For $0 < \varepsilon \leq 1/3$, he constructs a $(1+\varepsilon)$-approximate filtration of the Rips filtration. The size of its k-skeleton is only $n(\frac{1}{\varepsilon})^{O(\Delta k)}$, where Δ is the doubling dimension of the metric. Since then, several alternative technique have been explored for Rips [11] and Čech complexes [4, 9, 19], all arriving at the same complexity bound.

While the above approaches work well for instances where Δ and k are small, we focus on high-dimensional point sets. This has two reasons: first, one might simply want to analyze data sets for which the intrinsic dimension is high, but the existing methods do not succeed in reducing the complex size sufficiently. Second, even for medium-size dimensions, one might not want to restrict its scope to the low-homology features, so that $k = \Delta$ is not an unreasonable parameter choice. To adapt the aforementioned schemes to play nice with high dimensional point clouds, it makes sense to use dimension reduction results to eliminate the dependence on Δ. Indeed, it has been shown, in analogy to the famous Johnson-Lindenstrauss Lemma [16], that an orthogonal projection to a $O(\log n/\varepsilon^2)$-dimensional subspace yields another $(1+\varepsilon)$ approximate filtration [18, 24]. Combining these two approximation schemes, however, yields an approximation of size $O(n^{k+1})$ (ignoring ε-factors) and does not improve upon the exact case.

Our contributions. We present two results about the approximation of Rips and Čech filtrations: we give a scheme for approximating the Rips filtration with smaller complex size than existing approaches, at the price of guaranteeing only an approximation quality of polylog(n). Since Rips and Čech filtrations approximate each other by a constant factor, our result also extends to the Čech filtration, with an additional constant factor in the approximation quality. Second, we prove that any approximation scheme for the Čech filtration has superpolynomial size in n if high accuracy is required. For this result, our proof technique does not extend to Rips complexes. In more detail, our results are as follows:

Upper bound: We present a $6(d+1)$-approximation of the Rips filtration for n points in $\mathbb{R}^d$ whose k-skeleton has a size of $n2^{O(d \log k)}$ on each scale. This shows that by using a more rough approximation, we can achieve asymptotic improvements on the complex size. The real power of our approach reveals itself in high dimensions, in combination with dimension reduction techniques. In conjunction with the lemma of Johnson and Lindenstrauss [16], we obtain an $O(\log n)$-approximation with size $n^{O(\log k)}$ at any scale, which is much smaller than the original filtration; however, for the complete case $k = \log n$, the bound is still superpolynomial in n. Combined with a different dimension reduction result of Matoušek [20], we obtain a $O(\log^{3/2} n)$-approximation of size $n^{O(1)}$. This is the first polynomial bound in n of

an approximate filtration, independent of the dimensionality of the point set. For inputs from arbitrary metric spaces (instead of points in $\mathbb{R}^d$), the same results hold with an additional $O(\log n)$ factor in the approximation quality.

Our approximations are discrete, and the number of scales that have to be considered is determined by the logarithm of the spread of the point set (the ratio of diameter and closest point distance). In this work, we tacitly assume the spread to be constant, and concentrate on the complex size on a fixed scale as our quality measurement.

Lower bound: We construct a point set of n points in $d = \Theta(\log n)$ dimensions whose Čech filtration has $n^{\Omega(\log \log n)}$ persistent features with "relatively long" lifetime. Precisely, that means that any $(1+\delta)$-approximation has to contain a bar of non-zero length for each of those features if $\delta < O(\frac{1}{\log^{1+c} n})$ with $c \in (0,1)$. This shows that it is impossible to define an approximation scheme that yields an accurate approximation of the Čech complexes as well as polynomial size in n.

Methods: Our results follow from a link to lattice geometry: the A^*-lattice is a configuration of points in $\mathbb{R}^d$ which realizes the thinnest known coverings for low dimensions [10]. The dual Voronoi polytope of a lattice point is the *permutahedron*, whose vertices are obtained by all coordinate permutations of a fixed point in $\mathbb{R}^d$.

Our technique resembles the perhaps simplest approximation scheme for point sets: if we digitize $\mathbb{R}^d$ with d-dimensional pixels, we can take the union of pixels that contain input points as our approximation. Our approach does the same, except that we use a tessellation of permutahedra for digitization. In $\mathbb{R}^2$, our approach corresponds to the common approach of replacing the square tiling by a hexagonal tiling. We exploit that the permutahedral tessellation is in generic position, that is, no more than $d+1$ polytopes have a common intersection. At the same time, permutahedra are still relatively round, that is, they have small diameter and non-adjacent polytopes are well-separated. These properties ensure good approximation quality and a small complex. In comparison, a cubical tessellation yields a $O(\sqrt{d})$-approximate Rips filtration which looks like an improvement over our $O(d)$-approximation, but the highly degenerate configuration of the cubes yields a complex size of $n2^{O(dk)}$, and therefore does not constitute an improvement over Sheehy's approach [23].

For the lower bound, we arrange n points in a way that one center point has the permutahedron as Voronoi polytope, and we consider simplices incident to that center point in a fixed dimension. We show a superpolynomial number of these simplices create or destroy topological features of non-negligible persistence.

Outline of the paper. We begin by reviewing basics of persistent homology in Section 2. Next, we study several relevant properties of the A^* lattice in Section 3. An approximation algorithm based on concepts from Section 3 is presented in Section 4. In Section 5, we present the lower bound result on the size of Čech filtrations. We conclude in Section 6. We frequently refer to the arXiv version of our paper [8] for details of several proofs.

2 Topological background

We review some topological concepts needed in our argument. More extensive treatments covering most of the material can be found in the textbooks [12, 15, 21].

Simplicial complexes. For an arbitrary set V, called *vertices*, a *simplicial complex* over V is a collection of non-empty subsets which is closed under taking non-empty subsets. The elements of a simplicial complex K are called *simplices* of K. A simplex σ is a *face* of τ if $\sigma \subseteq \tau$. A *facet* is a face of co-dimension 1. The dimension of σ is $k := |\sigma| - 1$; we also call σ a k-simplex in this case. The *k-skeleton* of K is the collection of all simplices of dimension at most k. For instance, the 1-skeleton of K is a graph defined by its 0- and 1-simplices.

We discuss two ways of generating simplicial complexes. In the first one, take a collection $\mathcal{S}$ of sets over a common universe (for instance, polytopes in $\mathbb{R}^d$), and define the *nerve* of $\mathcal{S}$ as the simplicial complex whose vertex set is $\mathcal{S}$, and a k-simplex σ is in the nerve if the corresponding $(k+1)$ sets have a non-empty common intersection. The *nerve theorem* [3] states that if all sets in $\mathcal{S}$ are convex subsets of $\mathbb{R}^d$, their nerve is homotopically equivalent to the union of the sets (the statement can be generalized significantly; see [15, Sec. 4.G]). The second construction that we consider are *flag complexes*: Given a graph $G = (V, E)$, we define a simplicial complex K_G over the vertex set V such that a k-simplex σ is in K if for every distinct pair of vertices $v_1, v_2 \in \sigma$, the edge (v_1, v_2) is in E. In other words, K_G is the maximal simplicial complex with G as its 1-skeleton. In general, a complex K is called a flag complex, if $K = K_G$ with G being the 1-skeleton of K.

Given a set of points P in $\mathbb{R}^d$ and a parameter r, the *Čech complex at scale r*, $\mathcal{C}_r$ is defined as the nerve of the balls centered at the elements of P, each of radius r. This is a collection of convex sets. Therefore, the nerve theorem is applicable and it asserts that the nerve agrees homotopically with the union of balls. In the same setup, we can as well consider the intersection graph G of the balls (that is, we have an edge between two points if their distance is at most $2r$). The flag complex of G is called the *(Vietoris-)Rips complex at scale r*, denoted by $\mathcal{R}_r$. The relation $\mathcal{C}_r \subseteq \mathcal{R}_r \subseteq \mathcal{C}_{\sqrt{2}r}$ follows from Jung's Theorem [17].

Persistence Modules and simplicial filtrations. A *persistence module* $(V_\alpha)_{\alpha \in G}$ for a totally ordered index set $G \subseteq \mathbb{R}$ is a sequence of vector spaces with linear maps $F_{\alpha,\alpha'} : V_\alpha \to V_{\alpha'}$ for any $\alpha \leq \alpha'$, satisfying $F_{\alpha,\alpha} = id$ and $F_{\alpha',\alpha''} \circ F_{\alpha,\alpha'} = F_{\alpha,\alpha''}$. Persistence modules can be decomposed into *indecomposable intervals* giving rise to a *persistent barcode* which is a complete discrete invariant of the corresponding module.

A distance measure between persistence modules is defined through interleavings: we call two modules (V_α) and (W_α) with linear maps $F_{\cdot,\cdot}$ and $G_{\cdot,\cdot}$ *additively ε-interleaved*, if there exist linear maps $\phi : V_\alpha \to W_{\alpha+\varepsilon}$ and $\psi : W_\alpha \to V_{\alpha+\varepsilon}$ such that the maps ϕ and ψ commute with F and G (see [7]). We call the modules *multiplicatively c-interleaved* with $c \geq 1$, if there exist linear maps $\phi : V_\alpha \to W_{c\alpha}$ and $\psi : W_\alpha \to V_{c\alpha}$ with the same commuting properties. Equivalently, this means that the modules are additively $(\log c)$-interleaved when switching to a logarithmic scale. In this case, we also call the module (G_α) a *c-approximation* of (F_α) (and vice versa). Note that the case $c = 1$ implies that the two modules give rise to the same persistent barcode, which is usually referred to as the *persistence equivalence theorem* [12].

The most common way to generate persistence modules is through the homology of sequences of simplicial complexes: a *(simplicial) filtration* $(K_\alpha)_{\alpha \in G}$ over a totally order index set $G \subseteq \mathbb{R}$ is a sequence of simplicial complexes connected by simplicial maps $f_{\alpha,\alpha'} : K_\alpha \to K_{\alpha'}$ for any $\alpha \leq \alpha'$, such that $f_{\alpha,\alpha} = id$ and $f_{\alpha',\alpha''} \circ f_{\alpha,\alpha'} = f_{\alpha,\alpha''}$. By the functorial properties of homology (using some fixed field $\mathbb{F}$ and some fixed dimension $p \geq 0$), such a filtration gives rise to a persistence module $(H_p(K_\alpha, \mathbb{F}))_{\alpha \in G}$. We call a filtration a c-approximation of another filtration if the corresponding persistence modules induced by homology are c-approximations of each other.

The standard way of obtaining a filtration is through a nested sequence of simplicial complexes, where the simplicial maps are induced by inclusion. Examples are the *Čech*

filtration $(\mathcal{C}_\alpha)_{\alpha\in\mathbb{R}}$ and the *Rips filtration* $(\mathcal{R}_\alpha)_{\alpha\in\mathbb{R}}$. By the relations of Rips and Čech complexes from above, the Rips filtration is a $\sqrt{2}$-approximation of the Čech filtration.

Simplex-wise Čech filtrations and (co-)face distances. In the Čech filtration $(\mathcal{C}_\alpha)$, every simplex has an *alpha value* $\alpha_\sigma := \min\{\alpha \geq 0 \mid \sigma \in \mathcal{C}_\alpha\}$, which equals the radius of the minimal enclosing ball of its boundary vertices. If the point set P is finite, the Čech filtration consists of a finite number of simplices, and we can define a *simplex-wise filtration*

$$\emptyset = \mathcal{C}^0 \subsetneq \mathcal{C}^2 \subsetneq \ldots \subsetneq \mathcal{C}^m,$$

where exactly one simplex is added from $\mathcal{C}^i$ to $\mathcal{C}^{i+1}$, and where σ is added before τ whenever $\alpha_\sigma < \alpha_\tau$. The filtration is not unique and ties can be broken arbitrarily.

In a simplex-wise filtration, passing from $\mathcal{C}^i$ to $\mathcal{C}^{i+1}$ means adding the k-simplex $\sigma := \sigma_{i+1}$. The effect of this addition is that either a k-homology class comes into existence, or a $(k-1)$-homology class is destroyed. Depending on the case, we call σ *positive* or *negative*, accordingly. In terms of the corresponding persistent barcode, there is exactly one interval *associated to* σ either starting at i (if σ is positive) or ending at i (if σ is negative). We define the *(co-)face distance* L_σ (L^*_σ) of σ as the minimal distance between α_σ and its (co-)facets,

$$L_\sigma := \min_{\tau \text{ facet of } \sigma} \alpha_\sigma - \alpha_\tau \qquad L^*_\sigma := \min_{\tau \text{ co-facet of } \sigma} \alpha_\tau - \alpha_\sigma.$$

Note that L_σ and L^*_σ can be zero. Nevertheless, they constitute lower bounds for the persistence of the associated barcode intervals. An alternative to our proof is to argue using structural properties of the matrix reduction algorithm for persistent homology [12].

▶ **Lemma 1.** *If σ is negative, the barcode interval associated to σ has persistence at least L_σ.*

Proof. σ kills a $(k-1)$-homology class by assumption, and this class is represented by the cycle $\partial\sigma$. However, this cycle came into existence when the last facet τ of σ was added. Therefore, the lifetime of the cycle destroyed by σ is at least $\alpha_\sigma - \alpha_\tau$. ◀

▶ **Lemma 2.** *If σ is positive, the homology class created by σ has persistence at least L^*_τ*

Proof. σ creates a k-homology class; every representative cycle of this class is non-zero for σ. To turn such a cycle into a boundary, we have to add a $(k+1)$-simplex τ with σ in its boundary (otherwise, any $(k+1)$-chain formed will be zero for σ). Therefore, the cycle created at σ persists for at least $\alpha_\tau - \alpha_\sigma$. ◀

3 The A^*-lattice and the permutahedron

A *lattice* L in $\mathbb{R}^d$ is the set of all integer-valued linear combination of d independent vectors, called the *basis* of the lattice. Note that the origin belongs to every lattice. The *Voronoi polytope* of a lattice L is the closed set of all points in $\mathbb{R}^d$ for which the origin is among the closest lattice points. Since lattices are invariant under translations, the Voronoi polytopes for other lattice points are just translations of the one at the origin, and these polytopes tile $\mathbb{R}^d$. An elementary example is the integer lattice, spanned by the unit vectors $(e_1, \ldots, e_d)$, whose Voronoi polytope is the unit d-cube, shifted by $(-1/2)$ in each coordinate direction.

We are interested in a different lattice, called the A^*_d-lattice, whose properties are also well-studied [10]. First, we define the A_d lattice as the set of points $(x_1, \cdots, x_{d+1}) \in \mathbb{Z}^{d+1}$ satisfying $\sum_{i=1}^{d+1} x_i = 0$. A_d is spanned by vectors of the form $(e_i, -1)$, $i = 1, \ldots, d$. While it is defined in $\mathbb{R}^{d+1}$, all points lie on the hyperplane H defined by $\sum_{i=1}^{d+1} y_i = 0$. After a

suitable change of basis, we can express A_d by d vectors in $\mathbb{R}^d$; thus, it is indeed a lattice. In low dimensions, A_2 is the hexagonal lattice, and A_3 is the FCC lattice that realizes the best sphere packing configuration in $\mathbb{R}^3$ [14].

The *dual lattice* L^* of a lattice L is defined as the set of points $(y_1, \ldots, y_d)$ in $\mathbb{R}^d$ such that $y \cdot x \in \mathbb{Z}$ for all $x \in L$ [10]. Both the integer lattice and the hexagonal lattice are self-dual, while the dual of A_3 is the BCC lattice that realizes the thinnest sphere covering configuration among lattices in $\mathbb{R}^3$ [2].

We are mostly interested in the Voronoi polytope Π_d generated by A_d^*. Again, the definition becomes easier when embedding $\mathbb{R}^d$ one dimension higher as the hyperplane H. In that representation, it is known [10] that Π_d has $(d+1)!$ vertices obtained by all permutations of the coordinates of

$$\frac{1}{2(d+1)}(d, d-2, d-4, \cdots, -d+2, -d).$$

Π_d is known as the *permutahedron* [25, Lect. 0]. [1] Our approximation results in Section 4 and 5 are based on various combinatorial and geometric properties of Π_d, which we describe next. We will fix d and write $A^* := A_d^*$ and $\Pi := \Pi_d$ for brevity.

Combinatorics. The k-faces of Π correspond to ordered partitions of the coordinate indices $[d+1] := \{1, \cdots, d+1\}$ into $(d+1-k)$ non-empty subsets $S_1, \cdots, S_{d+1-k}$ such that all coordinates in S_i are smaller than all coordinates in S_j for $i < j$ [25]. For example, with $d = 3$, the partition $(\{1,3\}, \{2,4\})$ is the 2-face spanned by all points for which the two smallest coordinates appear at the first and the third position. This is an example of a facet of Π, for which we need to partition the indices in exactly 2 subsets; equivalently, the facets of Π are in one-to-one correspondence to non-empty proper subsets of $[d+1]$ so Π has $2^{d+1} - 2$ facets. The vertices of Π are the $(d+1)$-fold ordered partitions which correspond to permutations of $[d+1]$, reassuring the fact that Π has $(d+1)!$ vertices. Moreover, two faces σ, τ of Π with $\dim \sigma < \dim \tau$ are incident if the partition of σ is a refinement of the partition of τ. Continuing our example from before, the four 1-faces bounding the 2-face $(\{1,3\}, \{2,4\})$ are $(\{1\}, \{3\}, \{2,4\})$,$(\{3\}, \{1\}, \{2,4\})$, $(\{1,3\}, \{2\}, \{4\})$, and $(\{1,3\}, \{4\}, \{2\})$. Vice versa, we obtain co-faces of a face by combining consecutive partitions into one larger partition. For instance, the two co-facets of $(\{1,3\}, \{4\}, \{2\})$ are $(\{1,3\}, \{2,4\})$ and $(\{1,3,4\}, \{2\})$.

▶ **Lemma 3.** *Let σ and τ be two facets of Π, defined by the partitions $(S_\sigma, [d+1] \setminus S_\sigma)$ and $(S_\tau, [d+1] \setminus S_\tau)$, respectively. Then σ and τ are adjacent in Π iff $S_\sigma \subseteq S_\tau$ or $S_\tau \subseteq S_\sigma$.*

Proof. Two facets are adjacent if they share a common face. By the properties of the permutahedron, this means that the two facets are adjacent if and only if their partitions permit a common refinement, which is only possible if one set is contained in the other. ◀

We have already established that Π has "few" $(2^{d+1} - 2 = O(2^d))$ $(d-1)$-faces and "many" $((d+1)! = O(2^{d \log d}))$ 0-faces. We give an interpolating bound for all intermediate dimensions.

▶ **Lemma 4.** *The number of $(d-k)$-faces of Π is bounded by $2^{2(d+1)\log_2(k+1)}$.*

Proof. By our characterization of faces of Π, it suffices to count the number of ordered partitions of $[d+1]$ into $(k+1)$ subsets. That number equals $(k+1)!$ times the number

[1] Often, a scaled, translated and rotated version is considered, in which all permutations of the point $(1, \ldots, d+1)$ are taken.

of unordered partitions. The number of unordered partitions, in turn, is known as *Stirling number of the second kind* [22] and bounded by $\frac{1}{2}\binom{d+1}{k+1}(k+1)^{d-k}$. Multiplying with $(k+1)!$ yields an upper bound for $(d-k)$-faces, which can be bounded by $(k+1)^{2(d+1)}$ for $k \leq d$. ◀

Geometry. All vertices of Π are equidistant from the origin, and it can be checked with a simple calculation that this distance is $\sqrt{\frac{d(d+2)}{12(d+1)}}$. Using the triangle inequality, we obtain:

▶ **Lemma 5.** *The diameter of* Π *is at most* $\sqrt{d}$.

The permutahedra centered at all lattice points of A^* define the Voronoi tessellation of A^*. Its nerve is the Delaunay triangulation $\mathcal{D}$ of A^*. An important property of A^* is that, unlike for the integer lattice, $\mathcal{D}$ is non-degenerate – this will ultimately ensure small upper bounds for the size of our approximation scheme.

▶ **Lemma 6.** *Each vertex of a permutahedral cell has precisely* $d+1$ *cells adjacent to it. In other words, the* A_d^* *lattice points are in general position.*

The proof idea is to look at any vertex of the Voronoi cell and argue that it has precisely $(d+1)$ equidistant lattice points. See [1, Thm.2.5] for a concise, or [8] for a detailed argument. As a consequence, we can identify Delaunay simplices incident to the origin with faces of Π.

▶ **Proposition 7.** The $(k-1)$-simplices in $\mathcal{D}$ that are incident to the origin are in one-to-one-correspondence to the $(d-k+1)$-faces of Π and, hence, in one-to-one correspondence to the ordered k-partitions of $[d+1]$.

Let V denote the set of lattice points that share a Delaunay edge with the origin. The following statement shows that the point set V is in convex position, and the convex hull encloses Π with some "safety margin". The proof is a mere calculation, deriving an explicit equation for each hyperplane supporting the convex hull and applying it to all vertices of V and of Π. The argument is detailed in [8].

▶ **Lemma 8.** *For each d-simplex attached to the origin, the facet* τ *opposite to the origin lies on a hyperplane which is at least a distance* $\frac{1}{\sqrt{2}(d+1)}$ *to* Π *and all points of* V *are either on the hyperplane or on the same side as the origin.*

▶ **Lemma 9.** *If two lattice points are not adjacent in* $\mathcal{D}$*, the corresponding Voronoi polytopes have a distance of at least* $\frac{\sqrt{2}}{d+1}$.

Proof. Lemma 8 shows that Π is contained in a convex polytope C and the distance of Π to the boundary of C is at least $\frac{1}{\sqrt{2}(d+1)}$. Moreover, if Π' is the Voronoi polytope of a non-adjacent lattice point o', the corresponding polytope C' is interior-disjoint from C. To see that, note that the simplices in $\mathcal{D}$ incident to the origin triangulate the interior of C, and likewise for o' any interior intersection would be covered by a simplex incident to o and one incident to o', and since they are not connected, the simplices are distinct, contradicting the fact that $\mathcal{D}$ is a triangulation. Having established that C and C' are interior-disjoint, the distance between Π and Π' is at least $\frac{2}{\sqrt{2}(d+1)}$, as required. ◀

Recall the definition of a flag complex as the maximal simplicial complex one can form from a given graph. We next show that $\mathcal{D}$ is of this form. While our proof exploits certain properties of A^*, we could not exclude the possibility that the Delaunay triangulation of any lattice is a flag complex.

▶ **Lemma 10.** $\mathcal{D}$ *is a flag complex.*

Proof. The proof is based on two claims: consider two facets f_1 and f_2 of Π that are disjoint, that is, do not share a vertex. In the tessellation, there are permutahedra Π_1 attached to f_1 and Π_2 attached to f_2. The first claim is that Π_1 and Π_2 are disjoint. We prove this explicitly by constructing a hyperplane separating Π_1 and Π_2. See [8] for further details.

The second claim is that if k facets of Π are pairwise intersecting, they also have a common intersection. Another way to phrase this statement is that the link of any vertex in $\mathcal{D}$ is a flag complex. This is a direct consequence of Lemma 3. See [8] for more details.

The lemma follows directly with these two claims: consider $k+1$ vertices of $\mathcal{D}$ which pairwise intersect. We can assume that one point is the origin, and the other k points are the centers of permutahedra that intersect Π in a facet. By the contrapositive of the first claim, all these facets have to intersect pairwisely, because all vertices have pairwise Delaunay edges. By the second claim, there is some common vertex of Π to all these facets, and the dual Delaunay simplex contains the k-simplex spanned by the vertices. ◀

4 Approximation scheme

Given a point set P of n points in $\mathbb{R}^d$, we describe our approximation complex X_β for a fixed scale $\beta > 0$. For that, let L_β denote the A_d^* lattice in $\mathbb{R}^d$, with each lattice vector scaled by β. Recall that the Voronoi cells of the lattice points are scaled permutahedra which tile $\mathbb{R}^d$. The bounds for the diameter (Lemma 5) as well as for the distance between non-intersecting Voronoi polytopes (Lemma 9) remain valid when multiplying them with the scale factor. Hence, any cell of L_β has diameter at most $\beta\sqrt{d}$. Moreover any two non-adjacent cells have a distance at least $\beta\frac{\sqrt{2}}{d+1}$.

We call a permutahedron *full*, if it contains a point of P, and *empty* otherwise (we assume for simplicity that each point in P lies in the interior of some permutahedron; this can be ensured with well-known methods [13]). Clearly, there are at most n full permutahedra for a given P. We define X_β as the nerve of the full permutahedra defined by L_β. An equivalent formulation is that X_β is the subcomplex of $\mathcal{D}$ defined in Section 3 induced by the lattice points of full permutahedra. This implies that X_β is also a flag complex. We usually identify the permutahedron and its center in L_β and interpret the vertices of X_β as a subset of L_β.

Interleaving. To prove that X_β approximates the Rips filtration, we define simplicial maps connecting the complexes on related scales.

Let V_β denote the subset of L_β corresponding to full permutohedra. To construct X_β, we use a map $v_\beta : P \to V_\beta$, which maps each point $p \in P$ to its closest lattice point. Vice versa, we define $w_\beta : V_\beta \to P$ to map a vertex in V_β to the closest point of P. Note that $v_\beta \circ w_\beta$ is the identity map, while $w_\beta \circ v_\beta$ is not.

▶ **Lemma 11.** *The map v_β induces a simplicial map $\phi_\beta : \mathcal{R}_{\frac{\beta}{\sqrt{2}(d+1)}} \to X_\beta$.*

Proof. Because X_β is a flag complex, it is enough to show that for any edge (p, q) in $\mathcal{R}_{\frac{\beta}{\sqrt{2}(d+1)}}$, $(v_\beta(p), v_\beta(q))$ is an edge of X_β. This follows at once from the contrapositive of Lemma 9. ◀

▶ **Lemma 12.** *The map w_β induces a simplicial map $\psi_\beta : X_\beta \to \mathcal{R}_{\beta 2\sqrt{d}}$.*

Proof. It is enough to show that for any edge (p, q) in X_β, $(w_\beta(p), w_\beta(q))$ is an edge of $\mathcal{R}_{\beta 2\sqrt{d}}$. Note that $w_\beta(p)$ lies in the permutahedron of p and similarly, $w_\beta(q)$ lies in the permutahedron of q, so their distance is bounded by twice the diameter of the permutahedron. The statement follows from Lemma 5. ◀

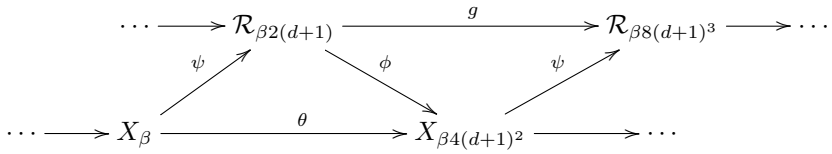
Since $\beta 2\sqrt{d} < \beta 2(d+1)$, we can compose the map ψ_β from the previous lemma with an inclusion map to a simplicial map $X_\beta \to \mathcal{R}_{\beta 2(d+1)}$ which we denote by ψ_β as well. Composing the simplicial maps ψ and ϕ, we obtain simplicial maps

$$\theta_\beta : X_\beta \to X_{\beta(2(d+1))^2}$$

for any β, giving rise to a discrete filtration

$$\left(X_{\beta(2(d+1))^{2k}}\right)_{k\in\mathbb{Z}}.$$

The maps define the following diagram of complexes and simplicial maps between them (we omit the indices in the maps for readability):



(1)

Here, g is the inclusion map of the corresponding Rips complexes. Applying the homology functor yields a sequence of vector spaces and linear maps between them.

▶ **Lemma 13.** *Diagram 1 commutes on the homology level, that is, $\theta_* = \phi_* \circ \psi_*$ and $g_* = \psi_* \circ \phi_*$, where the asterisk denotes the homology map induced by the simplicial map.*

Proof. For the first statement, note that θ is defined as $\phi \circ \psi$, so the maps commute already at the simplicial level. The second identity is not true on a simplicial level; we show that the maps g and $h := \psi \circ \phi$ are *contiguous*, that means, for every simplex $(x_0, \ldots, x_k) \in \mathcal{R}_{\beta 2(d+1)}$, the simplex $(g(x_0), \ldots, g(x_k), h(x_0), \ldots, h(x_k))$ forms a simplex in $\mathcal{R}_{\beta 8(d+1)^3}$. Contiguity implies that the induced homology maps g_* and $h_* = \psi_* \circ \phi_*$ are equal [21, §12].

It suffices to prove that any pair of vertices among $\{g(x_0), \ldots, g(x_k), h(x_0), \ldots, h(x_k)\}$ is at most $\beta 16(d+1)^3$ apart. This is immediately clear for any pair $(g(x_i), g(x_j))$ and $(h(x_i), h(x_j))$, so we can restrict to pairs of the form $(g(x_i), h(x_j))$. Note that $g(x_i) = x_i$ since g is the inclusion map. Moreover, $h(x_j) = \psi(\phi(x_j))$, and $\ell := \phi(x_j)$ is the closest lattice point to x_j in $X_{\beta 4(d+1)^2}$. Since $\psi(\ell)$ is the closest point in P to ℓ, it follows that $\|x_j - h(x_j)\| \le 2\|x_j - \ell\|$. With Lemma 5, we know that $\|x_j - \ell\| \le \beta 4(d+1)^2\sqrt{d}$, which is the diameter of the permutahedron cell. Using triangle inequality, we obtain

$$\|g(x_i) - h(x_j)\| \le \|x_i - x_j\| + \|x_j - h(x_j)\| \le \beta 4(d+1) + \beta 8(d+1)^2\sqrt{d} < \beta 16(d+1)^3$$

◀

▶ **Theorem 14.** *The persistence module $\left(H_*(X_{\beta(2(d+1))^{2k}})\right)_{k\in\mathbb{Z}}$ is a $6(d+1)$-approximation of $(H_*(\mathcal{R}_\beta))_{\beta\ge 0}$.*

Proof. Lemma 13 proves that on the logarithmic scale, the two filtrations are *weakly ε-interleaved* with $\varepsilon = 2(d+1)$, in the sense of [7]. Theorem 4.3 of [7] asserts that the bottleneck distance of the filtrations is at most 3ε. ◀

Complexity bounds. We exploit the non-degenerate configuration of the permutahedral tessellation to prove that X_β is not too large. We let $X_\beta^{(k)}$ denote the k-skeleton of X_β.

▶ **Theorem 15.** *For any β, $X_\beta^{(k)}$ has at most $n2^{O(d\log k)}$ simplices.*

Proof. We fix k and a vertex v of V_β. Recall that v represents a permutahedron, which we also denote by $\Pi(v)$. By definition, any k-simplex containing v corresponds to an intersection of $(k+1)$ permutahedra, involving $\Pi(v)$. By Proposition 7, such an intersection corresponds to a $(d-k)$-face of $\Pi(v)$. Therefore, the number of k-simplices involving v is bounded by the number of $(d-k)$-faces of the permutahedron, which is $2^{O(d \log k)}$ using Lemma 4. The bound follows because X_β has at most n vertices. ◀

▶ **Theorem 16.** *For any β, $X_\beta^{(k)}$ can be computed in $O(n2^d + k^2 2^d |X_\beta^{(k)}|)$ time. In particular, the construction takes $n2^{O(d \log k)}$ in the worst case.*

Proof. To find the vertices of X_β, we find, for each $p \in P$, the closest point to p in the scaled lattice L_β. For that, we use the algorithm from [10, Chap.20] which first finds the closest point in the coarser lattice A_d and then inspects a neighborhood of that lattice point to find the closest point in L_β. This algorithm inspects at most $O(d^2)$ lattice points, thus finding the vertex set runs in $O(nd^2)$ time.

To find the edges of X_β, we fix a vertex $v \in V_\beta$ and inspect all the 2^d neighbors, checking for each neighbor whether it is in V_β or not. This can be done in time $O(n2^d)$ time.

Finally, to find the higher-dimensional simplices, we simply compute the flag complex over the obtained graph (Lemma 10). For every $v \in V_\beta$ and any k-simplex $\sigma \in X_\beta$ involving v, we search for co-facets of σ: for every neighbor w not involved in X_β, we test whether $w * \sigma$ is a $(k+1)$-simplex of X_β. This test is combinatorial and costs $O(k^2)$ time. Consequently, for every simplex encountered, we spend an overhead of $O(k^2 2^d)$. ◀

Dimension reduction. For large d, our approximation complex plays nicely together with dimension reduction techniques. We start with noting that interleavings satisfy the triangle inequality. This result is folklore; see [6, Thm 3.3] for a proof in a generalized context.

▶ **Lemma 17.** *Let (A_β), (B_β), and (C_β) be persistence modules. If (A_β) is a t_1-approximation of (B_β) and (B_β) is a t_2-approximation of (C_β), then (A_β) is a $(t_1 t_2)$-approximation of (C_β).*

The following statement is a straight-forward application of interleaving distances from [7]. We provide a proof in [8].

▶ **Lemma 18.** *Let $f : P \to \mathbb{R}^m$ be an injective map such that*

$$\xi_1 \|p - q\| \leq \|f(p) - f(q)\| \leq \xi_2 \|p - q\|$$

for some constants $\xi_1 \leq 1 \leq \xi_2$. Let $\overline{\mathcal{R}}_\alpha$ denote the Rips complex of the point set $f(P)$. Then, the persistence module $(H_*(\overline{\mathcal{R}}_\alpha))_{\alpha \geq 0}$ is an $\frac{\xi_2}{\xi_1}$-approximation of $(H_*(\mathcal{R}_\alpha))_{\alpha \geq 0}$.

As a first application, we show that we can shrink the approximation size from Theorem 15 for the case $d \gg \log n$, only worsening the approximation quality by a constant factor.

▶ **Theorem 19.** *Let P be a set of n points in $\mathbb{R}^d$. There exists a constant c and a discrete filtration of the form $\left(\bar{X}_{(c \log n)^{2k}}\right)_{k \in \mathbb{Z}}$ that is $(3c \log n)$-interleaved with the Rips filtration of P and at each scale β, $\bar{X}_\beta^{(k)}$ has only $n^{O(\log k)}$ simplices. Moreover, we can compute, with high success probability, a complex $\bar{X}_\beta^{(k)}$ with this property in deterministic running time $O(dn \log n) + k^2 n^{O(1)} |X_\beta^{(k)}| = n^{O(\log k)}$.*

Proof. The famous lemma of Johnson and Lindenstrauss [16] asserts the existence of a map f as in Lemma 18 for $m = \lambda \log n / \varepsilon^2$ with some absolute constant λ and $\xi_1 = (1 - \varepsilon)$,

$\xi_2 = (1+\varepsilon)$. Choosing $\varepsilon = 1/2$, we obtain that $m = O(\log n)$ and $\xi_2/\xi_1 = 3$. With $\overline{\mathcal{R}}_\alpha$ the Rips complex of the Johnson-Lindenstrauss transform, we have therefore that $(H_*(\overline{\mathcal{R}}_\alpha))_{\alpha\geq 0}$ is a 3-approximation of $(H_*(\mathcal{R}_\alpha))_{\alpha\geq 0}$. Moreover, using the approximation scheme from this section, we can define a filtration $(\bar{X}_\beta)_{\beta\geq 0}$ whose induced persistence module $(H_*(X_\beta))_{\beta\geq 0}$ is a $6(m+1)$-approximation of $(H_*(\overline{\mathcal{R}}_\alpha))_{\alpha\geq 0}$, and its size at each scale is $n2^{O(\log n \log k)} = n^{O(\log k)}$. The first half of the result follows using Lemma 17.

The Johnson-Lindenstrauss lemma further implies that an orthogonal projection to a randomly chosen subspace of dimension m will yield an f as above, with high probability. Our algorithm picks such a subspace, projects all points into this subspace (this requires $O(dn \log n)$ time) and applies the approximation scheme for the projected point set. The runtime bound follows from Theorem 16. ◀

Note that for $k = \log n$, the approximation complex from the previous theorem is of size $n^{O(\log\log n)}$ and thus super-polynomial in n. Using a slightly more elaborated dimension reduction result by Matoušek [20], we can get a size bound polynomial in n, at the price of an additional $\log n$-factor in the approximation quality. Let us first state Matoušek result (whose proof follows a similar strategy as for the Johnson-Lindenstrauss lemma):

▶ **Theorem 20.** *Let P be an n-point set in $\mathbb{R}^d$. Then, a random orthogonal projection into $\mathbb{R}^k$ for $3 \leq k \leq C \log n$ distorts pairwise distances in P by at most $O(n^{2/k}\sqrt{\log n/k})$. The constants in the bound depend only on C.*

By setting $k := \frac{4\log n}{\log\log n}$ in Matoušek's result, we see that this results in a distortion of at most $O(\sqrt{\log n \log\log n})$.

▶ **Theorem 21.** *Let P be a set of n points in $\mathbb{R}^d$. There exists a constant c and a discrete filtration of the form $\left(\bar{X}_{\left(c\log n\left(\frac{\log n}{\log\log n}\right)^{1/2}\right)^{2k}}\right)_{k\in\mathbb{Z}}$ that is $3c\log n\left(\frac{\log n}{\log\log n}\right)^{1/2}$-interleaved with the Rips filtration on P and at each scale β, $\bar{X}_\beta^{(k)}$ has at most $n^{O(1)}$ simplices. Moreover, we can compute, with high success probability, a complex $\bar{X}_\beta^{(k)}$ with this property in deterministic running time $n^{O(1)}$.*

Proof. The proof follows the same pattern of Theorem 19 with a few changes. We use Matoušek's dimension reduction result described in Theorem 20 with the projection dimension being $m := \frac{4\log n}{\log\log n}$. Hence, $\xi_2/\xi_1 = O(\sqrt{\log n \log\log n})$ for the Rips construction. The final approximation factor is $6(m+1)\xi_2/\xi_1$ which simplifies to $O(\log n\left(\frac{\log n}{\log\log n}\right)^{1/2})$. The size and runtime bounds follow by substituting the value of m in the respective bounds. ◀

Finally, we consider the important generalization that P is not given as an embedding in $\mathbb{R}^d$, but as a point sample from a general metric space. We use the classical result by Bourgain [5] to embed P in Euclidean space with small distortion. In the language of Lemma 18, Bourgain's result permits an embedding into $m = O(\log^2 n)$ dimensions with a distortion $\xi_2/\xi_1 = O(\log n)$, where the constants are independent of n. Our strategy for approximating a general metric space consists of first embedding it into $\mathbb{R}^{O(\log^2 n)}$, then reducing the dimension, and finally applying our approximation scheme on the projected embedding. The results are similar to Theorems 19 and 21, except that the approximation quality further worsens by a factor of $\log n$ due to Bourgain's embedding. We only state the generalized version of Theorem 21, omitting the corresponding generalization of Theorem 19. The proof is straight-forward with the same techniques as before.

▶ **Theorem 22.** *Let P be a general metric space with n points. There exists a constant c and a discrete filtration of the form $\left(\bar{X}_{\left(c\log^2 n(\frac{\log n}{\log\log n})^{1/2}\right)^{2k}}\right)_{k\in\mathbb{Z}}$ that is $3c\log^2 n(\frac{\log n}{\log\log n})^{1/2}$-interleaved with the Rips filtration on P and at each scale β, $\bar{X}_\beta^{(k)}$ has at most $n^{O(1)}$ simplices. Moreover, we can compute, with high success probability, a complex $\bar{X}_\beta^{(k)}$ with this property in deterministic running time $n^{O(1)}$.*

5 A lower bound for approximation schemes

We describe a point configuration for which the Čech filtration gives rise to a large number, say N, of features with "large" persistence, relative to the scale on which the persistence appears. Any ε-approximation of the Čech filtration, for ε small enough, has to contain at least one interval per such feature in its persistent barcode, yielding a barcode of size at least N. This constitutes a lower bound on the size of the approximation itself, at least if the approximation stems from a simplicial filtration: in this case, the introduction of a new interval in the barcode requires at least one simplex to be added to the filtration; also more generally, it makes sense to assume that any representation of a persistence module is at least as large as the size of the resulting persistence barcode.

To formalize what we mean by a "large" persistent feature, we call an interval (α,α') of $(H_*(\mathcal{C}_\alpha))_{\alpha\geq 0}$ *δ-significant* for $0<\delta<\frac{\alpha'-\alpha}{2\alpha'}$. Our approach from above translates into the following statement:

▶ **Lemma 23.** *For $\delta>0$, and a point set P, let N denote the number of δ-significant intervals of $(H_*(\mathcal{C}_\alpha))_{\alpha\geq 0}$. Then, any persistence module $(X_\alpha)_{\alpha\geq 0}$ that is an $(1+\delta)$-approximation of $(H_*(\mathcal{C}_\alpha))_{\alpha\geq 0}$ has at least N intervals in its barcode.*

Proof. If (α,α') is δ-significant, that means that there exist some $\varepsilon>0$ and $c\in(\alpha,\alpha')$ such that $\alpha/(1-\varepsilon)\leq c/(1+\delta)<c(1+\delta)\leq\alpha'$. Any persistence module that is an $(1+\delta)$-approximation of $(H_*(\mathcal{C}_\alpha))_{\alpha\geq 0}$ needs to represent an approximation of the interval in the range $(c(1-\varepsilon)/2,c)$; in other words, there is an interval corresponding to (α,α') in the approximation. See [8] for more details. ◀

Setup. We next define our point set for a fixed dimension d. Consider the A^* lattice with origin o. Recall that o has $2^{d+1}-2$ neighbors in the Delaunay triangulation $\mathcal{D}$ of A_d^*, because its dual Voronoi polytope, the permutahedron Π, has that many facets. We define P as the union of o with all its Delaunay neighbors, yielding a point set of cardinality $2^{d+1}-1$. As usual, we set $n:=|P|$, so that $d=\Theta(\log n)$.

We write $\mathcal{D}_P$ for the Delaunay triangulation of P. Since P contains o and all its neighbors, the Delaunay simplices of $\mathcal{D}_P$ incident to o are the same as the Delaunay simplices of $\mathcal{D}$ incident to o. Thus, according to Proposition 7, a $(k-1)$-simplex of $\mathcal{D}_P$ incident to o corresponds to a $(d-k+1)$-face of Π and thus to an ordered k-partition of $[d+1]$.

Fix a integer parameter $\ell\geq 3$, to be defined later. We call an ordered k-partition $(S_1,\ldots,S_k)$ *good*, if $|S_i|\geq\ell$ for every $i=1,\ldots,k$. We define good Delaunay simplices and good permutahedron faces accordingly using Proposition 7.

Our proof has two main ingredients: First, we show that a good Delaunay simplex either gives birth to or kills an interval in the Čech module that has a lifetime of at least $\frac{\ell}{8(d+1)^2}$. This justifies our notion of "good", since good k-simplices create features that have to be preserved by a sufficiently precise approximation. Second, we show that there are $2^{\Omega(d\log\ell)}$ good k-partitions, so good faces are abundant in the permutahedron.

Persistence of good simplices. Let us consider our first statement. Recall that α_σ is the filtration value of σ in the Čech filtration. It will be convenient to have an upper bound for α_σ. Clearly, such a value is given by the diameter of P. It is not hard to see the following bound (compare Lemma 5), which we state for reference:

▶ **Lemma 24.** *The diameter of P is at most $2\sqrt{d}$. Consequently, $\alpha_\sigma \leq 2\sqrt{d}$ for each simplex σ of the Čech filtration.*

Recall that by fixing a simplex-wise filtration of the Čech filtration, it makes sense to talk about the persistence of an interval associated to a simplex. Fix a $(k-1)$-simplex σ of $\mathcal{D}_P$ incident to o (which also belongs to the Čech filtration).

▶ **Lemma 25.** *Let f_σ be the $(d-k)$ face of Π dual to σ, and let o_σ denote its barycenter. Then, α_σ is the distance of o_σ from o.*

Proof. o_σ is the closest point to o on f_σ because $\vec{o}o_\sigma$ is orthogonal to $\vec{p}o_\sigma$ for any boundary vertex p of f_α. Since f_σ is dual to σ, all vertices of σ are in same distance to o_σ. ◀

Recall L_σ and L^*_σ from Section 2 as the difference of the alpha value of σ and its (co-)facets.

▶ **Theorem 26.** *For a good simplex σ of $\mathcal{D}_P$, both L_σ and L^*_σ are at least $\frac{\ell}{24(d+1)^{3/2}}$.*

Proof. We start with L^*_σ. Let σ be a $(k-1)$-simplex and let $S_1, \ldots, S_k$ be the corresponding partition. We obtain a co-facet τ of σ through splitting one S_i into two non-empty parts.

The main step is to bound the quantity $\alpha_\tau^2 - \alpha_\sigma^2$. By Lemma 25, the alpha values are the squared norms of the barycenters o_τ of τ and o_σ of σ, respectively. It is possible to derive an explicit expression of the coordinates of o_σ and o_τ. It turns out that almost all coordinates are equal, and thus cancel out in the sum, except at those indices that lie in the split set S_i. Carrying out the calculations (as we do in [8]), we obtain the bound

$$\alpha_\tau^2 - \alpha_\sigma^2 \geq \frac{(\ell-1)}{4(d+1)}.$$

Moreover, $\alpha_\tau \leq 2\sqrt{d}$ by Lemma 24. This yields

$$\alpha_\tau - \alpha_\sigma = \frac{\alpha_\tau^2 - \alpha_\sigma^2}{\alpha_\tau + \alpha_\sigma} \geq \frac{\alpha_\tau^2 - \alpha_\sigma^2}{2\alpha_\tau} \geq \frac{\ell - 1}{16(d+1)\sqrt{d}} \geq \frac{\ell}{24(d+1)^{3/2}}$$

for $\ell \geq 3$. The bound on L^*_σ follows. For L_σ, note that $\min_{\tau \text{ facet of } \sigma} L^*_\tau \leq L_\sigma$, so it is enough to bound L^*_τ for all facets of σ. With σ being a $(k-1)$-simplex, all but one of its facets are obtained by merging two consecutive S_i and S_{i+1}. However, the obtained partition is again good (because σ is good), so the first part of the proof yields the lower bound for all these facets. It remains to argue about the facet of σ that is not attached to the origin. For this, we change the origin to any vertex of σ. It can be observed (through the combinatorial properties of Π) that with respect to the new origin, σ has the representation $(S_j, \ldots, S_k, S_1, \ldots, S_{j-1})$, thus the partition is cyclically shifted. In particular, σ is still good with respect to the new origin. We obtain the missing facet by merging the (now consecutive) sets S_k and S_1, which is also a good face, and the first part of the statement implies the result. ◀

As a consequence of Theorem 26, the interval associated with a good simplex has length at least $\frac{\ell}{24(d+1)^{3/2}}$ using Lemma 1 and 2. Moreover, the interval cannot persist beyond the scale $2\sqrt{d}$ by Lemma 24. It follows

▶ **Corollary 27.** *The interval associated to a good simplex is δ-significant for $\delta < \frac{\ell}{96(d+1)^2}$.*

The number of good simplices. We assume for simplicity that $d+1$ is divisible by ℓ. We call a good partition $(S_1, \ldots, S_k)$ *uniform*, if each set consists of *exactly* ℓ elements. This implies that $k = (d+1)/\ell$.

▶ **Lemma 28.** *The number of uniform good partitions is exactly* $\frac{(d+1)!}{\ell!^{(d+1)/\ell}}$.

Proof. Choose an arbitrary permutation and place the first ℓ entries in the S_1, the second ℓ entries in S_2, and so forth. In each S_i, we can interchange the elements and obtain the same k-simplex. Thus, we have to divide out $\ell!$ choices for each of the $(d+1)/\ell$ bins. ◀

We use this result to bound the number of good k-simplices in the following theorem. To obtain the bound, we use estimates for the factorials using Stirling's approximation. Moreover, we fix some constant $\rho \in (0,1)$ and set $\ell = (d+1)^\rho$. After some calculations (see [8]), we obtain:

▶ **Theorem 29.** *For any constant* $\rho \in (0,1)$, $\ell = (d+1)^\rho$, $k = (d+1)/\ell$ *and* d *large enough, there exists a constant* $\lambda \in (0,1)$ *that only depends only on* ρ*, such that the number of good* k*-simplices is at least* $(d+1)^{\lambda(d+1)} = 2^{\Omega(d \log d)}$.

Putting everything together, we prove our lower bound theorem:

▶ **Theorem 30.** *There exists a point set of* n *points in* $d = \Theta(\log n)$ *dimensions, such that any* $(1+\delta)$*-approximation of its Čech filtration contains* $2^{\Omega(d \log d)}$ *intervals in its persistent barcode, provided that* $\delta < \frac{1}{96(d+1)^{1+\varepsilon}}$ *with an arbitrary constant* $\varepsilon \in (0,1)$.

Proof. Setting $\rho := 1 - \varepsilon$, Theorem 29 guarantees the existence of $2^{\Omega(d \log d)}$ good simplices, all in a fixed dimension k. In particular, the intervals of the Čech persistence module associated to these intervals are all distinct. Since $\ell = (d+1)^{1-\varepsilon}$, Corollary 27 states that all these intervals are significant because $\delta < \frac{1}{96d^{1+\varepsilon}} = \frac{\ell}{96(d+1)^2}$. Therefore, by Lemma 23, any $(1+\delta)$-approximation of the Čech filtration has $2^{\Omega(d \log d)}$ intervals in its barcode. ◀

Replacing d by $\log n$ in the bounds of theorem, we see the number of intervals appearing in any approximation super-polynomial is n if δ is small enough.

6 Conclusion

We presented upper and lower bound results on approximating Rips and Čech filtrations of point sets in arbitrarily high dimensions. For Čech complexes, the major result can be summarized as: for a dimension-independent bound on the complex size, there is no way to avoid a super-polynomial complexity for fine approximations of about $O(\log^{-1} n)$, while polynomial size can be achieved for rough approximation of about $O(\log^2 n)$.

Filling in the large gap between the two approximation factors is an attractive avenue for future work. A possible approach is to look at other lattices. It seems that lattices with good covering properties are correlated with a good approximation quality, and it may be worthwhile to study lattices in higher dimension which improve largely on the covering density of A^* (e.g., the Leech lattice [10]).

Our approach, like all other known approaches, approximate also the geometry of the point set as a by-product, and we have to allow for large error rates to overcome the curse of dimensionality. An alternative approach to bridge the gap between upper and lower bounds with an approximation scheme that only approximates topological features.

An unpleasant property of our approach is the dependence on the spread of the point set. We pose the question whether it is possible to eliminate this dependence by a more elaborate construction that avoids the mere gluing of approximation complexes of consecutive scales.

Acknowledgments. Michael Kerber acknowledges support of the Max Planck Center for Visual Computing and Communication. Sharath Raghvendra acknowledges support of NSF CRII grant CCF-1464276.

References

1 J. Baek and A.Adams. Some useful properties of the Permutohedral Lattice for Gaussian filtering. Technical report, Stanford University, 2009. URL: http://graphics.stanford.edu/papers/permutohedral/permutohedral_techreport.pdf.

2 R. Bambah. On Lattice coverings by Spheres. *Proceedings of the National Institute of Sciences of India*, 20:25–52, 1954.

3 K. Borsuk. On the embedding of systems of Compacta in Simplicial Complexes. *Fundamenta Mathematicae*, pages 217–234, 1948.

4 M. Botnan and G. Spreemann. Approximating Persistent Homology in Euclidean space through collapses. *Applied Algebra in Engineering, Communication and Computing*, 26(1-2):73–101, 2015. doi:10.1007/s00200-014-0247-y.

5 J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. doi:10.1007/BF02776078.

6 P. Bubenik and J.A. Scott. Categorification of Persistent Homology. *Discrete & Computational Geometry*, 51(3):600–627, 2014. doi:10.1007/s00454-014-9573-x.

7 F. Chazal, D. Cohen-Steiner, M. Glisse, L.J. Guibas, and S.Y. Oudot. Proximity of Persistence Modules and their Diagrams. In *ACM Symposium on Computational Geometry (SoCG)*, pages 237–246, 2009. doi:10.1145/1542362.1542407.

8 A. Choudhary, M. Kerber, and S. Raghvendra. Polynomial-sized Topological Approximations using the Permutahedron. *CoRR*, abs/1601.02732, 2016. URL: http://arxiv.org/abs/1601.02732.

9 A. Choudhary, M. Kerber, and R. Sharathkumar. Approximate Čech complexes in low and high dimensions. URL: http://people.mpi-inf.mpg.de/~achoudha/Files/AppCech.pdf.

10 J. H. Conway, N. J. A. Sloane, and E. Bannai. *Sphere-packings, Lattices, and Groups*. Springer-Verlag, 1987.

11 T.K. Dey, F. Fan, and Y. Wang. Computing topological persistence for simplicial maps. In *ACM Symposium on Computational Geometry (SoCG)*, pages 345–354, 2014. doi:10.1145/2582112.2582165.

12 H. Edelsbrunner and J. Harer. *Computational Topology – an Introduction*. American Mathematical Society, 2010. URL: http://www.ams.org/bookstore-getitem/item=MBK-69.

13 H. Edelsbrunner and E.P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 1990. doi:10.1145/77635.77639.

14 T. Hales. A proof of the Kepler conjecture. *Annals of Mathematics*, 162:1065–1185, 2005.

15 A. Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

16 W.B. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of Lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986. doi:10.1007/BF02764938.

17 H. Jung. Über die kleinste Kugel, die eine räumliche Figur einschliesst. *Journal reine angewandte Mathematik*, 123:241–257, 1901.

18 M. Kerber and S. Raghvendra. Approximation and streaming algorithms for projective clustering via random projections. In *Canadian Conference on Computational Geometry (CCCG)*, pages 179–185, 2015.

19 M. Kerber and R. Sharathkumar. Approximate Čech complex in low and high dimensions. In *Intern. Symp. on Algortihms and Computation (ISAAC)*, pages 666–676, 2013.

20 J. Matoušek. Bi-Lipschitz embeddings into low-dimensional Euclidean spaces. *Commentationes Mathematicae Universitatis Carolinae*, 1990.

21 J.R. Munkres. *Elements of algebraic topology*. Westview Press, 1984.

22 B.C. Rennie and A.J. Dobson. On Stirling numbers of the second kind. *Journal of Combinatorial Theory*, 7(2):116–121, 1969.

23 D. Sheehy. Linear-size approximations to the Vietoris-Rips Filtration. *Discrete & Computational Geometry*, 49(4):778–796, 2013. doi:10.1007/s00454-013-9513-1.

24 D. Sheehy. The persistent homology of distance functions under random projection. In *ACM Symposium on Computational Geometry (SoCG)*, 2014.

25 G.M. Ziegler. *Lectures on polytopes*. Springer-Verlag, 1995.

Faster Algorithms for Computing Plurality Points

Mark de Berg[*1], Joachim Gudmundsson[†2], and Mehran Mehr[‡3]

1 **Department of Computer Science, TU Eindhoven, The Netherlands**
`mdberg@win.tue.nl`
2 **School of IT, University of Sydney, Australia**
`joachim.gudmundsson@sydney.edu.au`
3 **Department of Computer Science, TU Eindhoven, The Netherlands**
`mmehr@tue.nl`

Abstract

Let V be a set of n points in $\mathbb{R}^d$, which we call voters, where d is a fixed constant. A point $p \in \mathbb{R}^d$ is preferred over another point $p' \in \mathbb{R}^d$ by a voter $v \in V$ if $\text{dist}(v,p) < \text{dist}(v,p')$. A point p is called a plurality point if it is preferred by at least as many voters as any other point p'.

We present an algorithm that decides in $O(n \log n)$ time whether V admits a plurality point in the L_2 norm and, if so, finds the (unique) plurality point. We also give efficient algorithms to compute a minimum-cost subset $W \subset V$ such that $V \setminus W$ admits a plurality point, and to compute a so-called minimum-radius plurality ball.

Finally, we consider the problem in the personalized L_1 norm, where each point $v \in V$ has a preference vector $\langle w_1(v), \ldots, w_d(v)\rangle$ and the distance from v to any point $p \in \mathbb{R}^d$ is given by $\sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$. For this case we can compute in $O(n^{d-1})$ time the set of all plurality points of V. When all preference vectors are equal, the running time improves to $O(n)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Computational geometry, computational social choice, voting theory, plurality points, Condorcet points

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.32

1 Introduction

We study computational problems concerning plurality points, a concept arising in social choice and voting theory, defined as follows. Let V be a set of n *voters* and let $\mathcal{C}$ be a space of possible choices. Each voter $v \in V$ has a utility function indicating how much v likes a certain choice, i.e. the utility function of v determines for any two choices from $\mathcal{C}$ which one is preferred by v or whether both choices are equally preferable. A (weak) plurality point is now defined as a choice $p \in \mathcal{C}$ such that no alternative $p' \in \mathcal{C}$ is preferred by more voters.

When there are different issues on which the voters can decide, then the space $\mathcal{C}$ becomes a multi-dimensional space. This has led to the study of plurality points in the setting where $\mathcal{C} = \mathbb{R}^d$ and each voter has an ideal choice which is a point in $\mathbb{R}^d$. To simplify the presentation, from now on we will not distinguish the voters from their ideal choice and

* MdB is supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003 and 022.005025.
† JG supported under Australian Research Council's Discovery Projects funding scheme (project number DP150101134).
‡ MM is supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 024.002.003 and 022.005025.


© Mark de Berg, Joachim Gudmundsson, and Mehran Mehr;
licensed under Creative Commons License CC-BY

so we view each voter $v \in V$ as being a point in $\mathbb{R}^d$, the so-called *spatial model* in voting theory [15]. Thus the utility of a point $p \in \mathbb{R}^d$ for a voter v is inversely proportional to $\text{dist}(v, p)$, the distance from v to p under a given distance function, and v prefers a point p over a point p' if $\text{dist}(v, p) < \text{dist}(v, p')$. Now a point $p \in \mathbb{R}^d$ is a plurality point if for any point $p' \in \mathbb{R}^d$ we have $|\{v \in V : \text{dist}(v, p) < \text{dist}(v, p')\}| \geqslant |\{v \in V : \text{dist}(v, p') < \text{dist}(v, p)\}|$.

Plurality points and related concepts were already studied in the 1970s in voting theory [6, 11, 10, 15, 17]. McKelvey and Wendell [15] define three different notions of plurality points – majority Condorcet, plurality Condorcet, and majority core – and for each notion they define a weak and a strong variant. Under certain assumptions on the utility functions, which are satisfied for the L_2 norm, the three notions are equivalent. Thus for the L_2 norm we only have two variants: weak plurality points (which should be at least as popular as any alternative) and strong plurality points (which should be strictly more popular than any alternative). We focus on weak plurality points, since they are more challenging from an algorithmic point of view. From now on, whenever we speak of plurality points we refer to weak plurality points.

Plurality points represent a stable choice with respect to the opinions of the voters. One can also look at the concept from the viewpoint of competitive facility location. Here one player wants to place a facility in the space $\mathcal{C}$ such that she always wins at least as many clients (voters) as her competitor, no matter where the competitor places his facility. Competitive facility location problems have been studied widely in a discrete setting, where the clients and the possible locations for the facilities are nodes in a network; see the survey by Kress and Pesch [13]. Competitive facility location has also been studied in a geometric, continuous setting under the name Voronoi games [1, 4]. Here one is given a region R in $\mathbb{R}^2$, say the unit square, and the goal is to win the maximum area within R. In other words, the set V of voters is no longer finite, but we have $V = \mathcal{C} = R$. The plurality-point problem in a geometric space lies in between the network setting and the fully continuous setting: the space $\mathcal{C}$ of choices is $\mathbb{R}^d$, but the set V of voters is finite.

When the L_2 norm defines the distance between voters and potential plurality points, then plurality points can be defined in terms of Tukey depth [16]. The *Tukey depth* of a point $p \in \mathbb{R}^d$ with respect to a given set V of n points is defined as the minimum number of points from V lying in any closed halfspace containing p. A point of maximum Tukey depth is called a *Tukey median*. It is known that for any set V, the depth of the Tukey median is at least $\lceil n/(d+1) \rceil$ and at most $\lceil n/2 \rceil$. Wu *et al.* [19] showed that a point $p \in \mathbb{R}^d$ is a plurality point in the L_2 norm if and only if any open halfspace with p on its boundary contains at most $n/2$ voters. This is equivalent to saying that the Tukey depth of p is $\lceil n/2 \rceil$. They used this observation to present an algorithm that decides in $O(n^{d-1} \log n)$ time if a plurality point exists for a given set V of n voters in $\mathbb{R}^d$. A slightly better result can be obtained using a randomized algorithm by Chan [2], which computes a Tukey median (together with its depth) in $O(n \log n + n^{d-1})$ time.

As is clear from the relation to Tukey depth, a plurality point in the L_2 norm does not always exist. In fact, the set V of voters must, in a certain sense, be highly symmetric to admit a plurality point. This led Lin *et al.* [14] to study the *minimum-cost plurality problem*. Here each voter is assigned a cost, and the goal is to find a minimum-cost subset $W \subset V$ of voters such that if we ignore the voters in W – that is, if we consider $V \setminus W$ – then a plurality point exists. Lin *et al.* gave an $O(n^5 \log n)$ algorithm for the planar version of the problem; whether the problem in $\mathbb{R}^3$ can be solved in polynomial time was left as an open problem.

In the voting-theory literature plurality points in the L_1 norm have also been considered [15, 17, 18]. One advantage of the L_1 norm is that in $\mathbb{R}^2$ a plurality point always

exists and can easily be found in $O(n)$ time: any 2-dimensional median is a plurality point. Unfortunately, this is no longer true when $d > 2$ [15, 17]. We are not aware of any existing algorithms for deciding whether a given set V in $\mathbb{R}^d$ admits a plurality point in the L_1 norm.

Our results. Currently the fastest algorithm for deciding whether a plurality point exists runs in $O(n \log n + n^{d-1})$ randomized time and actually computes a Tukey median. However, in the case of plurality points we are only interested in the Tukey median if its depth is the maximum possible, namely $\lceil n/2 \rceil$. Wu *et al.* [19] exploited this to obtain a deterministic algorithm, but their running time is $O(n^{d-1} \log n)$. This raises the question: can we decide whether a plurality point exists faster than by computing the depth of the Tukey median? We show that this is indeed possible: we present a deterministic algorithm that decides if a plurality point exists (and, if so, computes one) in $O(n \log n)$ time.

We then turn our attention to the minimum-cost plurality problem. We solve the open problem of Lin *et al.* [14] by presenting an algorithm that solves the problem in $O(n^4)$ time, in any (fixed) dimension. Note that this even improves on the $O(n^5 \log n)$ running time for the planar case. We also consider the following problem for unit-cost voters in the plane: given a parameter k, find a minimum-cost set W of size at most k such that $V \setminus W$ admits a plurality point, if such a set exists. Our algorithm for this case runs in $O(k^3 n \log n)$ time.

Ignoring some voters in order to have a plurality point is undesirable when almost all voters must be ignored. Instead of ignoring voters we can work with *plurality balls*, as defined next. The idea is that if two points p and p' are very similar, then voters do not care much whether p or p' is chosen. Thus we define a ball $b(p, r)$ centered at p and of radius r to be a plurality ball if the following holds: there is no point p' outside $b(p, r)$ that is preferred by more voters than p. Note that a plurality point is a plurality ball of zero radius. We show that in the plane, the minimum-radius plurality ball can be computed in $O(T(n))$ time, where $T(n)$ is the time needed to compute the $\lfloor n/2 \rfloor$-level in an arrangement of n lines.

Recall that the different dimensions represent different issues on which the voters can express their preferences. It is then natural to allow the voters to give different weights to these issues. This leads us to introduce what we call the *personalized L_1 norm*. Here each voter $v \in V$ has a *preference vector* $\langle w_1(v), \ldots, w_d(v) \rangle$ of non-negative weights that specifies the relative importance of the various issues. The distance of a point $p \in \mathbb{R}^d$ to a voter v is now defined as $\mathrm{dist}_w(v, p) := \sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$, where $x_i(\cdot)$ denotes the i-th coordinate of a point. We present an algorithm that decides in $O(n^{d-1})$ time whether a set V of n voters admits a plurality point with respect to the personalized L_1 norm. For the special case when all preference vectors are identical – this case reduces to the normal case of using the L_1 norm – the running time improves to $O(n)$.

2 Plurality points in the L_2 norm

Let V be a set of n voters in $\mathbb{R}^d$. In this section we show how to compute a plurality point for V with respect to the L_2 norm in $O(n \log n)$ time, if it exists. We start by proving several properties of the plurality point in higher dimensions, which generalize similar properties that Lin *et al.* [14] proved in $\mathbb{R}^2$. These properties imply that if a plurality point exists, it is unique (unless all points are collinear). Our algorithm then consists of two steps: first it computes a single candidate point $p \in \mathbb{R}^d$, and then it decides if p is a plurality point.

2.1 Properties of plurality points in the L_2 norm

As remarked in the introduction, plurality points can be characterized as follows.

▶ **Fact 1** (Wu *et al.* [19])**.** A point p is a plurality point for a set V of n voters in $\mathbb{R}^d$ with respect to the L_2 norm if and only if every open halfspace with p on its boundary contains at most $n/2$ voters.

Verifying the condition in Fact 1 directly is not efficient. Hence, we will prove alternative conditions for a point p to be a plurality point in $\mathbb{R}^d$, which generalize the conditions Lin *et al.* [14] stated for the planar case. First, we define some concepts introduced by Lin *et al.*

Let V be a set of n voters in $\mathbb{R}^d$, and consider a point $p \in \mathbb{R}^d$. Let $L(p)$ be the set of all lines passing through p and at least one voter $v \neq p$. The point p partitions each line $\ell \in L(p)$ into two opposite rays, which we denote by $\rho(\ell)$ and $\overline{\rho}(\ell)$. (The point p itself is not part of these rays.) We say that a line $\ell \in L(p)$ is *balanced* if $|\rho(\ell) \cap V| = |\overline{\rho}(\ell) \cap V|$. When n is odd, then p turns out to be a plurality point if and only if every line $\ell \in L(p)$ is balanced (which implies that we must have $p \in V$). When n is even the situation is more complicated. Let $R(p)$ be the set of all rays $\rho(\ell)$ and $\overline{\rho}(\ell)$. Label each ray in $R(p)$ with an integer, which is the number of voters on the ray minus the number of voters from V on the opposite ray. Thus, a line ℓ is balanced if and only if its rays $\rho(\ell)$ and $\overline{\rho}(\ell)$ have label zero. Let $L^*(p)$ be the set of all unbalanced lines in $L(p)$ and let $R^*(p)$ be the corresponding set of rays. We now define the so-called alternating property, as introduced by Lin *et al.* [14]. This property is restricted to the 2-dimensional setting, where we can order the rays in $R^*(p)$ around p. In this setting, the point p is said to have the *alternating property* if the following holds: the circular sequence of labels of the rays in $R^*(p)$, which we obtain when we visit the rays in $R^*(p)$ in clockwise order around p, alternates between labels $+1$ and -1. Note that if p has the alternating property then the number of unbalanced lines must be odd.

▶ **Theorem 2.** *Let V be a set of n voters in $\mathbb{R}^d$, with $d \geqslant 1$, and let p be an arbitrary point.*

(a) *If n is odd, p is a plurality point if and only if $p \in V$ and every line in $L(p)$ is balanced.*

(b) *If n is even and $p \notin V$, then p is a plurality point if and only if every line in $L(p)$ is balanced.*

(c) *If n is even and $p \in V$, then p is a plurality point if and only if all unbalanced lines in $L(p)$ are contained in a single 2-dimensional flat f and p has the alternating property for the set $V \cap f$.*

For $d = 1$ the theorem is trivial, and for $d = 2$ – the condition in case 3 then simply states that p has the alternating property – the theorem was proved by Lin *et al.* Our contribution is the extension to higher dimensions. Before proving Theorem 2, we need the following lemma regarding the robustness of plurality points to dimension reduction.

▶ **Lemma 3.** *Let p be a plurality point for a set V in $\mathbb{R}^d$, with $d \geqslant 1$, and let f be any lower-dimensional flat containing p. Then p is a plurality point for $V \cap f$.*

Proof. We prove the statement by induction on d. For $d = 1$ the lemma is trivially true, so now consider the case $d > 1$. We consider two cases.

The first case is that f is a hyperplane, that is, $\dim(f) = d-1$. Let f^+ and f^- denote the open halfspaces bounded by f, and assume without loss of generality that $|f^+ \cap V| \geqslant |f^- \cap V|$. Suppose for a contradiction that p is not a plurality point for $f \cap V$. Then there must be a $(d-2)$-flat $g \subset f$ containing p such that, within the $(d-1)$-dimensional space f, the number of voters lying strictly to one side of g is greater than $|f \cap V|/2$. Let $g^+ \subset f$ denote the part of f lying to this side of g. Now imagine rotating f around g by an infinitesimal amount. Let $\hat{f}$ denote the rotated hyperplane. Then all voters in $f^+ \cap V$ end up in $\hat{f}^+$. Moreover, we can choose the direction of the rotation such that the voters in $g^+ \cap V$ end up in $\hat{f}^+$. But

then $|\hat{f}^+ \cap V| = |f^+ \cap V| + |g^+ \cap V| > |f^+ \cap V| + |f \cap V|/2 \geqslant n/2$, which contradicts the assumption that p is a plurality point.

The second case is that $\dim(f) < d-1$. Let h be a hyperplane that contains f. From the first case we know that p must be a plurality point for $h \cap V$. Hence, we can apply our induction hypothesis to conclude that p must be a plurality point for $f \cap V$. ◀

▶ **Corollary 4.** *Let V be a set of voters in $\mathbb{R}^d$, for $d \geqslant 2$, that are not collinear. Then V has at most one plurality point.*

Proof. Suppose for a contradiction that V has two distinct plurality points p_1 and p_2. Let f be a 2-flat containing p_1 and p_2, and a voter v not collinear with p_1 and p_2. By Lemma 3, both p_1 and p_2 are plurality points for $f \cap V$. But this contradicts the result by Wu *et al.* [19] that any set of voters in the plane admits at most one plurality point. ◀

Now we are ready to prove Theorem 2.

Proof of Theorem 2. Since the case $d = 2$ was already proved by Lin *et al.* [14], and the case $d = 1$ is trivial, we assume $d \geqslant 3$. Below we prove part 3 , the proof for parts 1 and 2 is given in the full version.

(3,⇐). Assume n is even and let p be a point such that all unbalanced lines in $L(p)$ are contained in a single 2-dimensional flat f and p has the alternating property for the set $V \cap f$. Consider an arbitrary open halfspace h^+ whose bounding hyperplane h contains p, and let h^- be the opposite open halfspace. If h contains f then all unbalanced lines lie in h and so $|h^+ \cap V| = |h^- \cap V|$, which implies $|h^+ \cap V| \leqslant n/2$. If h does not contain f, we can argue as follows. Let $\ell := h \cap f$. Since the theorem is true for $d = 2$ and we have the alternating property on f, we know that p is a plurality point on f. Hence, the number of voters on f on either side of ℓ is at most $|f \cap V|/2$. But then we have $|h^+ \cap V| \leqslant n/2$, because all voters not in f lie on balanced lines. We conclude that for any open halfspace h^+ we have $|h^+ \cap V| \leqslant n/2$, and so p is a plurality point.

(3,⇒). Assume n is even and let p be a plurality point. We first argue that all unbalanced lines must lie on a single 2-flat. Assume for a contradiction that there are three unbalanced lines that do not lie on a common 2-flat. Let g be the 3-flat spanned by these lines, and let $L^*_g(p) \subset L^*(p)$ be the set of all unbalanced lines contained in g. Let $f_1 \subset g$ be a 2-flat not containing p and not parallel to any of the lines in $L^*_g(p)$. Each of the lines in $L^*_g(p)$ intersects f_1 in a single point, and these intersection points are not all collinear. According to the Sylvester-Gallai Theorem [9] this implies there is an ordinary line in g', that is, a line containing exactly two of the intersection points. Thus we have an ordinary 2-flat in g, that is, a flat f_2 containing exactly two lines from $L^*(p)$. This implies that $f_2 \cap V$ does not have the alternating property, and since we know by the result of Lin *et al.* that the theorem holds when $d = 2$ this implies that p is not a plurality point in f_2. However, this contradicts Lemma 3.

We just argued that all unbalanced lines must lie on a single 2-flat f. By Lemma 3 the point p is a plurality point on f. Since the theorem holds for $d = 2$, we can conclude that $f \cap V$ has the alternating property. ◀

2.2 Finding plurality points in the L_2 norm

We now turn our attention to finding a plurality point. Our algorithm needs a subroutine for finding a *median hyperplane* h for V, which is a hyperplane such that $|h^+ \cap V| < n/2$ and $|h^- \cap V| < n/2$, where h^+ and h^- denote the two open halfspaces bounded by h. The following lemma is easy to prove.

▶ **Lemma 5.** *Let $v \in V$ be a voter that lies on a hyperplane h_0 such that all voters either lie on h_0 or in h_0^+. Then we can find a median hyperplane h containing v in $O(n)$ time.*

Recall that for $d \geqslant 2$ the plurality point is unique, if it exists. The algorithm below either reports a single candidate point p – we show later how to test if the candidate is actually a plurality point or not – or it returns $\emptyset$ to indicate that it already discovered that a plurality point does not exist. When called with a set V of n collinear voters, the algorithm will return the set of all plurality points; if n is even the set is a segment connecting the two median voters, if n is odd the set is a degenerate segment consisting of the (in this case unique) median voter. We call this segment the *median segment*.

FindCandidates(V)

1. If all voters in V are collinear, then return the median segment of V.
2. Otherwise, proceed as follow.
   a. Let $v_0 \in V$ be a voter with minimum x_d-coordinate. Find a median hyperplane h_0 containing v_0 using Lemma 5, and let $cand_0 :=$ FindCandidates($h_0 \cap V$).
   b. If $cand_0$ is a single point or $cand_0 = \emptyset$ then return $cand_0$.
   c. If $cand_0$ is a (non-degenerate) segment then let $v_1 \in V$ be a voter whose distance to h_0 is maximized. Find a median hyperplane h_1 containing v_1 using Lemma 5, and let $cand_1 :=$ FindCandidates($h_1 \cap V$). Return $cand_0 \cap cand_1$.

▶ **Lemma 6.** *Algorithm* FindCandidates(V) *returns in $O(n)$ time a set* cand *of candidate plurality points such that*

(i) *if all voters in V are collinear then* cand *is the set of all plurality points of V;*
(ii) *if not all voters in V are collinear then* cand *contains at most one point, and no other point can be a plurality point of V.*

Proof. We first prove the correctness of the algorithm, and then consider the time bound.

If all voters in V are collinear then the algorithm returns the correct result in Step 1, so assume not all voters are collinear. Consider the median hyperplane computed in Step 2a. Since $|h_0^+ \cap V| < n/2$ and $|h_0^- \cap V| < n/2$, for any point $p \notin h$ there is an open halfspace containing p and bounded by a hyperplane parallel to h_0 that contains more than $n/2$ voters. Hence, by Fact 1 any plurality point for V must lie on h_0. By Lemma 3, if a plurality point exists for V it must also be a plurality point for $h_0 \cap V$. By induction we can assume that FindCandidates($h_0 \cap V$) is correct. Hence, the result of the algorithm is correct when $cand_0$ is a single point or $cand_0 = \emptyset$. Note that when $cand_0$ is a (non-degenerate) segment – this only happens when all voters in $h_0 \cap V$ are collinear – we must have $V \neq h_0 \cap V$, otherwise V would be collinear and we would be done after Step 1. Hence, $v_1 \notin h_0$. By the same reasoning as above the median hyperplane h_1 must contain the plurality point of V (if it exists). But then the plurality point must lie in $cand_0 \cap cand_1$, and since $v_1 \notin h_0$ we know that $cand_0 \cap cand_1$ is either a single point or it is empty. This proves the correctness.

To prove the time bound, we note that we only have two recursive calls when the first recursive call reports a non-degenerate candidate segment. This only happens when all voters in $h_0 \cap V$ are collinear, which implies the recursive call just needs to compute a median segment in $O(n)$ time – it does not make further recursive calls. Thus we can imagine adding this time to the original call, so that we never make more than one recursive call. Since the recursion depth is at most d, and each call needs $O(n)$ time, the bound follows. ◀

Our algorithm to find a plurality point first calls FindCandidates(V). If all points in V are collinear we are done – FindCandidates(V) then reports the correct answer. Otherwise

we either get a single candidate point p, or we already know that a plurality point does not exist. It remains to test if a candidate point p is a plurality point or not.

▶ **Lemma 7.** *Given a set V of n voters in $\mathbb{R}^d$ and a candidate point p, we can test in $O(n \log n)$ time if p is a plurality point in the L_2 norm.*

Proof. First compute the set $L(p)$ of lines containing p and at least one voter. We can compute $L(v)$, and for each line $\ell \in L(p)$ the number of voters on the rays $\rho(\ell)$ and $\overline{\rho}(\ell)$, in $O(n \log n)$ time. (To this end, we take the line ℓ_v through p and v for each voter $v \neq p$, and group these into subsets of identical lines.) According to Theorem 2, we can now immediately decide if p is a plurality point when n is odd, or when n is even and $p \notin V$. When n is even and $p \in V$ we first check in $O(n)$ time if all unbalanced lines lie in a 2-flat f. If not, then p is not a plurality point, otherwise we check the alternating property in $O(n \log n)$ time. ◀

We obtain the following theorem. (See the full version for the $\Omega(n \log n)$ lower bound.)

▶ **Theorem 8.** *Let V be a set of n voters in $\mathbb{R}^d$, where $d \geqslant 2$ is a fixed constant. Then we can find in $O(n \log n)$ time the plurality point for V in the L_2 norm, if it exists, and this time bound is optimal.*

3 Dealing with point sets that do not admit a plurality point

Most point sets do not admit a plurality point in the L_2 norm. In this section we consider two ways of dealing with this: we present algorithms to compute a minimum-cost subset $W \subset V$ such that $V \setminus W$ admits a plurality point, and we present an algorithm for computing a minimum-radius plurality ball in $\mathbb{R}^2$.

3.1 The minimum-cost plurality problem

Let V be a set of n voters in $\mathbb{R}^d$, where each voter v has a cost $cost(v) > 0$ associated to it. For a candidate plurality point p – here we consider all points in $\mathbb{R}^d$ as candidates – we define W_p to be a minimum-cost subset of V such that p is a plurality point for $V \setminus W_p$. We define the *price* of p to be the cost of W_p. Our algorithm will report a pair (p, W_p), where p is a cheapest candidate plurality point. The algorithm has two main parts: one finds the cheapest candidate that does not coincide with one of the voters, the other finds the cheapest candidate that coincides with a voter.

Let $L(V)$ be the set of lines passing through at least two voters in V, and let $P(V)$ be the set of all intersection points of the lines in $L(V)$, excluding the intersection points coinciding with a voter. To find the cheapest candidate p that does not coincide with a voter, we only have to consider points in $P(V)$. Indeed, if all points in $V \setminus W_p$ are collinear then we can pick p to coincide with a voter; otherwise we know by Theorem 22 that p must be an intersection point of two lines in $L(V \setminus W_p)$, and so $p \in P(V)$. We will need the following lemma for the planar case.

▶ **Lemma 9** (Lin *et al.* [14])**.** *Let p be a candidate plurality point for a set V in $\mathbb{R}^2$. Then we can compute in $O(n \log n)$ time the price of p, together with the subset W_p.*

In the algorithm below, we use $L(p, V')$ to denote the set of lines through a point p and at least one voter in a set $V' \subseteq V$.

MinCostPluralityPoint(V)

1. Compute the set $L(V)$. If $|L(V)| = 1$, that is, all voters lie on a common line ℓ, then compute a median p along ℓ as a plurality point and report $(p, \emptyset)$.
2. Compute a cheapest candidate p that does not coincide with a voter, as follows. Compute the set $P(V)$. For each line $\ell \in L(V)$, sort the intersection points along ℓ. This can be done in $O(n^4)$ time in total, by projecting all lines onto an arbitrary 2-flat, constructing the arrangement in this 2-flat in $O(n^4)$ time [5], and then checking which intersections on the 2-flat correspond to actual intersections in $\mathbb{R}^d$. Let $C := \sum_{v \in V} cost(v)$ be the total cost of all voters. For each intersection point $p \in P(V)$, let $\gamma(p)$ be the total cost of all voters v for which there is no line in $L(v, V \setminus \{v\})$ that contains p; we can compute $\gamma(p)$ in $O(n^4)$ time in total by determining the total cost of all voters that *do* have a line in $L(v, V \setminus \{v\})$ that contains p, and then subtracting this cost from C.
   a. Traverse each line $\ell \in L(V)$, to visit the intersection points along ℓ in order. During the traversal, maintain the number of voters on ℓ on either side of the current intersection point p. Thus we know how many voters we have to remove to make ℓ balanced, and also from which side we should remove them. If we have to remove k voters, we have to remove the k cheapest voters on the relevant side. The subset $W_p(\ell)$ that we have to remove to make ℓ balanced only changes when p passes over a voter v on ℓ. When this happens we can compute the new $W_p(\ell)$ in linear time. In this way the traversal of ℓ takes $O(n^2)$ time in total, so over all $\ell \in L(V)$ we spend $O(n^4)$ time.
   b. For each intersection point p compute the price of p. This price has two components: the price to make every line $\ell \in L(V)$ that contains p balanced, and the price to remove any voter v for which the line $\ell(v, p)$ through v and p is not a line in $L(V)$. The first component equals $\sum_{\ell \ni p} cost(W_p(\ell))$. The second component equals $\gamma(p)$, which we precomputed.
3. Compute a cheapest candidate p that coincides with a voter $v \in V$. To this end, compute for each voter v the price of setting $p := v$ – below we describe how to do this in $O(n^3)$ time per voter – and take the cheapest of all n possibilities.

   Consider a candidate p coinciding with some $v \in V$. By Theorem 2 all unbalanced lines in $L_p(V \setminus W_p)$, if any, lie on a single 2-flat f. We will compute $price_p(f)$, the price to make p into a plurality point under the condition that all unbalanced lines lie in f, over all 2-flats f spanned by two lines from $L_p(V \setminus \{v\})$. Then we take the best of the results. Fix a 2-flat f spanned by two lines $\ell_1, \ell_2 \in L_p(V \setminus \{v\})$. Let L_f be the subset of lines from $L_p(V \setminus \{v\})$ contained in f, and let L'_f be the subset of lines not contained in f.
   a. Compute $price_p(L_f)$, the price of making p a plurality point on f, using Lemma 9. This takes $O(n_f \log n_f)$ time, where $n_f := |V \cap f|$.
   b. For each line $\ell \in L'_f$, compute $price_p(\ell)$, the price of making ℓ balanced. (This can easily be done in $O(n)$ time: we compute the smallest number, k, of voters we have to remove on ℓ to make ℓ balanced, and then find the k cheapest voters on the heavier of the two rays along ℓ and emanating from p.) Let $price_p(L'_f) := \sum_{\ell \in L'_f} price_p(\ell)$.
   c. Set $price_p(f) := price_p(L_f) + price_p(L'_f)$.
4. Let p be the cheaper of the two candidates found in Steps 2 and 3, respectively. Compute the set W_p for this candidate – this takes $O(n \log n)$ time – and report (p, W_p).

The correctness of the algorithm follows from Theorem 2 and the discussion above. As for the running time, we note that Steps 1, 2, and 4 all run in $O(n^4)$ time. For Step 3, the time needed to compute the price of a single voter v is $\sum_f O(n_f \log n_f + n)$, which is bounded by $O(n^3 + \sum_f n_f \log n_f)$. Because every voter $v' \in V$ lies on at most n of the

flats f (generated by the lines $\ell(v, v')$ and $\ell(v, v'')$ through v, v' and v, v'', respectively) we have $\sum_f n_f = O(n^2)$ and so $\sum_f n_f \log n_f = O(n^2 \log n)$. Hence, the whole algorithm runs in $O(n^4)$ time.

▶ **Theorem 10.** *Let V be a set of n voters in $\mathbb{R}^d$, each with a positive cost, where $d \geqslant 2$ is a fixed constant. Then we can compute in $O(n^4)$ time a minimum-cost subset $W \subset V$ such that $V \setminus W$ admits a plurality point in the L_2 norm.*

Our algorithm for finding a minimum-cost plurality point checks $O(n^4)$ candidate points. The algorithm from the previous section for deciding if a plurality point exists avoids this, resulting in a near-linear running time. An obvious question is if a faster algorithm is also possible for the minimum-cost plurality-point problem. While we do not have the answer to this question, we can show that, even in the plane and when all voters have unit cost, it is unlikely that the problem can be solved in truly subquadratic time. We do this by a reduction from the problem Three Concurrent Lines, which is to decide if a set of n lines has three or more lines meeting in a single point. Three Concurrent Lines is 3Sum-hard [8] and has an $\Omega(n^2)$ lower bound if only sidedness tests are used [7].

▶ **Theorem 11.** *Suppose we have an algorithm solving the minimum-cost plurality-point problem for any set of n unit-cost voters in the plane in time $T(n)$. Then there is a probabilistic algorithm solving* Three Concurrent Lines *with probability 1 in $O(n \log n + T(n))$ time.*

3.2 An output-sensitive algorithm for unit-cost voters in the plane

The proof of Theorem 11 uses a problem instance where many voters must be removed to obtain a plurality point. Below we show that a plurality point for which only a few voters have to be removed can be found in near-linear time, in the planar case and for unit-cost voters. More precisely, we consider the case where we are given a set V of unit-cost voters in the plane and a parameter k, and we want to compute a smallest subset $W \subset V$ of size at most k (if it exists) such that $V \setminus W$ admits a plurality point.

Define a *k-line* to be a line ℓ such that both open halfplanes bounded by ℓ contain at most $n/2 + k$ voters. For a voter v, let $L(v)$ be the set of lines containing v and at least one other voter, and let $L_k(v)$ be the set of all k-lines in $L(v)$.

▶ **Lemma 12.** *Let p be a plurality point of $V \setminus W$, for some subset W of size at most k. If $v \notin W$, then p lies on one of the lines in $L_k(v)$.*

Proof. Assume $v \notin W$. If $\ell(p, v)$, the line through p and v, is not a line in $L(v)$, then p does not coincide with a voter and $\ell(p, v)$ is unbalanced. But then p cannot be a plurality point, by Theorem 2. Hence, $\ell(p, v) \in L(v)$. If $\ell(p, v) \notin L_k(v)$ then there is a halfplane bounded by $\ell(p, v)$ containing more than $n/2 + k$ voters. But since $|W| \leqslant k$, such a point p cannot be a plurality point in $V \setminus W$, which implies we must have $\ell(p, v) \in L_k(v)$. ◀

The idea of our algorithm is now as follows. Consider a set $P := \{(v_{2i-1}, v_{2i}) : 1 \leqslant i \leqslant k+1\}$ of disjoint pairs of voters. Then there must be a pair $(v_{2i-1}, v_{2i}) \in P$ such that neither v_{2i-1} nor v_{2i} is in W. Hence, the point p we are looking for must lie on one of the lines in $L_k(v_{2i-1})$ and one of the lines in $L_k(v_{2i})$. So we check all intersection points between these lines, for every pair in P. The key to obtain an efficient algorithm is to generate P such that all sets $L_k(v_i)$ are small. There is one case that needs special attention, namely when there is a line – this must then be the line through v_{2i-1} and v_{2i} – that is present in both $L_k(v_{2i-1})$ and $L_k(v_{2i})$. This case is handled using the following lemma.

▶ **Lemma 13.** *Suppose we want to compute a cheapest plurality point on the line $\ell := \ell(v, v')$ through two voters $v, v' \in V$, where we are only interested in points of price at most k. Let ℓ^+ be any of the two open halfplanes bounded by ℓ, and let $V^+ := \ell^+ \cap V$. Then p is either an intersection point of ℓ and a line in $\bigcup_{v'' \in V^+} L_k(v'')$, or the point coinciding with any median of the voters located on ℓ is a cheapest plurality point.*

Proof. Let p be a cheapest plurality point and W_p the corresponding subset of voters to be removed, where $|W_p| \leqslant k$. If there is a voter $v'' \in V^+ \setminus W_p$ then $p \in L_k(v'')$ by Lemma 12. Otherwise all voters in V^+ are in W_p. But then any line through p and a voter in $\ell^- \cap V$ is unbalanced, which implies that all voters in $\ell^- \cap V$ except at most one must be in W_p. In this case the point coinciding with any of the at most two medians along ℓ is a cheapest plurality point on ℓ. ◀

We now present our algorithm. For technical reasons we assume $k \leqslant n/15$.

OutputSensitiveMinCostPluralityPoint(V, k)

1. Compute the set of convex layers of V. Let $V_1, V_2, \ldots$ be the sets of voters in these layers, where V_1 is the outermost layer, V_2 the next layer, and so on. Set $P := \emptyset$ and $i := 1$.
2. Visit the voters in V_i in clockwise order, starting at the lexicographically smallest voter in V_i. Put the first and second visited voters, the third and fourth visited voters, and so on as pairs into P until either P contains $k+1$ pairs or we run out of voters in V_i. In the former case we are done, in the latter case we start collecting pairs from the next layer, by setting $i := i + 1$ and repeating the process. This continues until we have collected $k+1$ pairs. (We are guaranteed we can collect this many pairs since $k \leqslant n/15$.)
3. Set $C := \emptyset$; the set C will contain candidates for the cheapest plurality points. For each pair $(v_{2i-1}, v_{2i}) \in P$, proceed as follows.
   a. Compute the sets $L_k(v_{2i-1})$ and $L_k(v_{2i})$, and put all intersection points between two lines in $L_k(v_{2i-1}) \cup L_k(v_{2i})$ as candidates into C.
   b. Let $\ell := \ell(v_{2i-1}, v_{2i})$. If ℓ is present in both $L_k(v_{2i-1})$ and $L_k(v_{2i})$, and ℓ contains at least $n/2 - 7k - 1$ voters, then proceed as follows. (We assume that the same line ℓ has not already been handled in this manner for a pair (v_{2j-1}, v_{2j}) with $j < i$, otherwise we can skip it now.) Assume without loss of generality that ℓ^+, the open halfplane above ℓ, contains at most as many voters as ℓ^-. For each voter $v \in \ell^+ \cap V$, compute $L_k(v)$ and add the intersection points of the lines in $L_k(v)$ with ℓ to the candidate set C. In addition, put a median voter along ℓ into C.
4. For each candidate point $p \in C$, compute a minimum-size subset W_p that makes p into a plurality point, using Lemma 9. Return the cheapest plurality point $p^* \in C$, provided $|W_{p^*}| \leqslant k$; if $|W_p| > k$ for all candidates, then report that it is not possible to obtain a plurality point by removing at most k voters.

The efficiency of our algorithm is based on the following lemma, using that we constructed P using the convex layers of V.

▶ **Lemma 14.** *Let v be a voter of some pair in P. Then $|L_k(v)| = O(k)$.*

We can now prove the following theorem.

▶ **Theorem 15.** *Let V be a set of n voters in the plane, and let k be a parameter with $k \leqslant n/15$. Then we can compute in $O(k^3 n \log n)$ time a minimum-size subset $W \subset V$ such that $V \setminus W$ admits a plurality point in the L_2 norm.*

Proof. To prove the time bound, we note that we can compute the convex layers in $O(n \log n)$ time [3]. Step 2 runs in $O(n)$ time. Computing the set $L_k(v)$ for a voter v can easily be done in $O(n \log n)$ time. By Lemma 14, Step 3a takes $O(n \log n + k^2)$ time. To bound the running time of Step 3b, we observe that there can be at most $O(1)$ pairs (v_{2i-1}, v_{2i}) to which this case applies. Indeed $\ell(v_{2i-1}, v_{2i})$ should contain at least $n/2 - 7k - 1$ voters, and since $k \leqslant n/15$ there can only be $O(1)$ such lines. Thus Step 3b needs $O(kn)$ time, and so Step 3 needs $O(k^2 n + kn \log n)$ time in total over all pairs in P, to generate $O(k^3)$ candidate points. Checking each of the candidates takes $O(n \log n)$, which proves the time bound.

The correctness of the algorithm follows from Lemmas 12 and 13, except for one thing: in Step 3b we only handle a line $\ell := \ell(v_{2i-1}, v_{2i})$ when it contains at least $n/2 - 7k - 1$ voters. This is allowed for the following reason. Note that ℓ is tangent to a convex hull $\mathrm{CH}(V_i)$ and on the side of ℓ that does not contain $\mathrm{CH}(V_i)$, say ℓ^+, there are at most $3k$ voters. Now consider a plurality point $p \in \ell$. Then there can be at most $3k + 1$ voters in $(V \setminus W_P) \cap \ell^-$, by Theorem 2. Since $|W_p| \leqslant k$, we thus have $|V \cap \ell^-| \leqslant 4k + 1$, which means that ℓ must contain at least $n - 7k - 1$ voters. ◀

3.3 The minimum-radius plurality-ball problem

Let V be a set of n voters in $\mathbb{R}^d$. A closed ball $b(p, r)$ of radius r and centered at a point p is a *plurality ball* if for any point $q \notin b(p, r)$ the number of voters who prefer p over q is at least the number of voters who prefer q over p. Note that for any point p the ball $b(p, r)$ is a plurality ball if r is sufficiently large, and that a plurality ball with $r = 0$ is a plurality point. Below we describe an algorithm to compute a minimum-radius plurality ball for V. If all voters are collinear then any point on the median segment of V is a plurality ball of radius 0, so in the remainder we assume not all voters are collinear.

We define the *core* of a ball $b(p, r)$ as $b(p, r/2)$. Fact 1 can be generalized as follows.

▶ **Fact 16.** A ball $b(p, r)$ is a plurality ball if and only if every open halfspace that does not intersect the core $b(p, r/2)$ contains at most $n/2$ voters.

To check this condition we use the concept of k-set and k-level and their duality. A *k-set* of V, for some $0 \leqslant k \leqslant n - d$, is defined as a subset $V' \subset V$ of size k such that there is an open halfspace h^+ with $h^+ \cap V = V'$ and with at least d points from V on its boundary. Let V^* be the set of hyperplanes dual to the voters in V, and consider the k-level in the arrangement $\mathcal{A}(V^*)$, that is, the set of points on the hyperplanes in V^* that have exactly k hyperplanes strictly below them. We associate each $(d-1)$-facet f of the k-level of $\mathcal{A}(V^*)$ to a cone in the primal space, as follows. Let $V^*(f)$ be the set of hyperplanes strictly below f, and consider the hyperplanes (in primal space) dual to the vertices of f. Then $cone(f)$ is the closed cone defined by these hyperplanes that contains the k voters whose dual hyperplanes are in $V^*(f)$ plus, at its apex, the voter whose dual hyperplane contains f. We call $cone(f)$ a *k-cone*. A k-cone contains exactly $k + 1$ voters including the voter at its apex, and the other $n - k - 1$ voters all lie in the opposite cone.

▶ **Lemma 17.** *A ball $b(p, r)$ is a plurality ball if and only if its core $b(p, r/2)$ intersects all $\lfloor n/2 \rfloor$-cones of V.*

Proof. Assume all $\lfloor n/2 \rfloor$-cones are intersected by $b(p, r/2)$ and suppose for a contradiction that p is not a plurality point. By Fact 16 there must be an open halfspace h^+ not intersecting $b(p, r/2)$ and containing more than $n/2$ voters. But then there must be a $\lfloor n/2 \rfloor$-cone contained inside the halfspace, which is a contradiction. On the other hand, if $b(p, r/2)$ does not intersect some $\lfloor n/2 \rfloor$-cone $cone(f)$ then there is an open halfspace h^+ not

intersecting $b(p, r/2)$ containing all the points in $cone(f)$. Therefore $|h^+ \cap V| \geqslant \lfloor n/2 \rfloor + 1$, and so $b(p, r)$ is not a plurality ball. ◀

Our algorithm is now easy: We compute all $\lfloor n/2 \rfloor$-cones of V by computing the $\lfloor n/2 \rfloor$-level in the dual arrangement $\mathcal{A}(V^*)$. Then we compute the minimum-radius ball $b(p, r/2)$ intersecting all these cones, and report $b(p, r)$ as the minimum-radius plurality ball. Since in $\mathbb{R}^2$ a minimum-radius disk intersecting all the cones is computable in linear time [12] we obtain the following result.

▶ **Theorem 18.** *Let V be a set of n voters in the plane. Then we can compute the minimum-radius plurality ball for V in $O(T(n))$ time, where $T(n)$ is the time needed to compute the $\lfloor n/2 \rfloor$-level in an arrangement of n lines in the plane.*

Computing the k-level in an arrangement of lines can be done in $O(n \log n + m \log^{1+\varepsilon} n)$ time, where m is the complexity of the level. Our algorithm then runs in $O(n^{4/3} \log^{1+\varepsilon} n)$ time. We believe the algorithm described above can be generalized to higher dimensions, using a standard algorithm for computing k-levels, and generalized linear programming for the second part of the algorithm. We are currently verifying the details.

4 Plurality points in the personalized L_1 norm

Let V be a set of n voters in $\mathbb{R}^d$, where each voter $v \in V$ has a preference vector $\langle w_1(v), \ldots, w_d(v) \rangle$ of non-negative weights. Define $\text{dist}_w(v, p) := \sum_{i=1}^{d} w_i(v) \cdot |x_i(v) - x_i(p)|$. In this section we study plurality points for this personalized L_1 distance. As mentioned in the introduction, a plurality point in the L_1 norm always exists in $\mathbb{R}^2$, but not in higher dimensions [17]. Interestingly, in the personalized L_1 norm the statement already fails in the plane: in the full version we given an example of a weighted point set V in $\mathbb{R}^2$ that does not admit a plurality point in the personalized L_1 norm.

4.1 Properties of plurality points in the personalized L_1 norm

Our goal is to formulate conditions that help us to find candidate plurality points and to decide if a given candidate is actually a plurality point. For the L_2 norm we used Theorem 2 and Lemma 3 for this. Here we need a different approach. Recall that a candidate point $p \in \mathbb{R}^d$ is a plurality point if, for any point $q \in \mathbb{R}^d$, the number of voters who prefer p is at least the number of voters who prefer q. From now on we refer to the point q as a *competitor*.

For two points p and q, define $V[p \succ q] := \{v \in V : \text{dist}_w(v, p) < \text{dist}_w(v, q)\}$. We also define $V[p \sim q] := \{v \in V : \text{dist}_w(v, p) = \text{dist}_w(v, q)\}$ and $V[p \succcurlyeq q] := V[p \succ q] \cup V[p \sim q]$. Let p be a candidate plurality point. We call a point q a *non-degenerate competitor for p* if $V[p \sim q] = \emptyset$, and we say that q is *ε-close* to p if $|pq| < \varepsilon$, where $|pq|$ denotes the Euclidean distance between p and q. The following lemma implies that to test if a point p is a plurality point, we only have to consider non-degenerate competitors that are ε-close to p.

▶ **Lemma 19.** *Let p be a candidate plurality point and let q be a competitor of p. For any $\varepsilon > 0$ there is a non-degenerate competitor q' that is ε-close to p such that $\big|V[q' \succ p]\big| \geqslant \big|V[q \succ p]\big| + \frac{1}{2} \cdot \big||V[q \sim p]\big|$.*

The following lemma helps us to narrow down our search for plurality points. Recall that a *multi-dimensional median* for V is a point $p \in \mathbb{R}^d$ such that, for all $1 \leqslant i \leqslant d$, we have $\big|\{v \in V : x_i(v) < x_i(p)\}\big| \leqslant n/2$ and $\big|\{v \in V : x_i(v) > x_i(p)\}\big| \leqslant n/2$.

▶ **Lemma 20.** *Let p be a plurality point for V in the personalized L_1 norm. Then p is a multi-dimensional median for V.*

The set M_V of all multi-dimensional medians for V is an axis-aligned hyperrectangle in $\mathbb{R}^d$, that is, it can be written as $M_V = I_1 \times \cdots \times I_d$, where each I_i is a closed interval that may degenerate into a single value. We call M_V the *median box* of V. Lemma 20 states that we only have to look at points in M_V when searching for plurality points. The next theorem implies that we only have to check which vertices of M_V are plurality points to fully classify the set of all plurality points. Let F be the set of all k-dimensional facets of M_V for $0 \leqslant k \leqslant d$, where each facet $f \in F$ is considered relatively open. Note that $|F| = 3^{d'}$, where d' is the number of non-degenerate intervals defining M_V.

▶ **Theorem 21.** *Let V be a set of voters in $\mathbb{R}^d$ and let f be a relatively open facet of the median box M_V of V.*

(1) *Either all points in f are plurality points in the personalized L_1 norm, or none of the points in f are.*

(2) *The points in f are plurality points in the personalized L_1 norm if and only if all vertices of f are plurality points.*

Proof. Part (1) follows immediately from part (2). Next we prove part (2).

(2,⇒). Suppose $p \in f$ is a plurality point, and consider a vertex p' of f. We need to prove that p' is also a plurality point. Let $\varepsilon > 0$ small enough so that for each $1 \leqslant i \leqslant d$ and every voter $v \in V$ with $x_i(v) \neq x_i(p)$ we have $|x_i(v) - x_i(p)| > \varepsilon$; define ε' similarly for p'. Let b and b' be Euclidean balls of radius $\min(\varepsilon, \varepsilon')$ and centered at p and p', respectively. Since p is a plurality point we know that $|V[p \succ q]| \geqslant |V[q \succ p]|$ for all $q \in b$. Now consider an arbitrary point $q' \in b'$, and define $q := q' + (p - p')$. Note that $q \in b$ and that the relative position of q and p is the same as the relative position of q' and p'. In particular, $x_i(p) - x_i(q) = x_i(p') - x_i(q')$ for all $1 \leqslant i \leqslant d$. As shown in the full version, this implies that $V[p \succ q] \subseteq V[p' \succ q']$ and $V[q' \succ p'] \subseteq V[q \succ p]$. Hence, $|V[p' \succ q']| \geqslant |V[p \succ q]| \geqslant |V[q \succ p]| \geqslant |V[q' \succ p']|$. Since q' is an arbitrary point in b', the point p' must be a plurality point according to Lemma 19.

(2,⇐). To prove this it suffices to show the following: if p is a point in the relative interior of f that is not a plurality point, then there is a vertex p' of f' that is not a plurality point. To this end let q be an ε-close competitor (for a sufficiently small $\varepsilon > 0$) who beats p. We will argue that the vertex p' on the opposite side of p as compared to q' – this is defined more precisely below – is not a plurality point.

Let $M_V = I_1 \times \cdots \times I_d$, where $I_i = [\min_i, \max_i]$ (possibly with $\min_i = \max_i$). For each $1 \leqslant i \leqslant d$, we pick $x_i(p')$ as follows. If $x_i(p) = \min_i$ or $x_i(p) = \max_i$ then we set $x_i(p') := x_i(p)$. Otherwise we have $\min_i < x_i(p) < \max_i$; then, if $x_i(p) \leqslant x_i(q)$ we set $x_i(p') := \min_i$, and if $x_i(p) > x_i(q)$ we set $x_i(p') := \max_i$. Now consider the competitor q' of p' defined by $q' := q + (p' - p)$, so that q' has the same positive relative to p as q has relative to p. We claim the following: for any voter v and any $1 \leqslant i \leqslant d$ we have $|x_i(v) - x_i(p')| - |x_i(v) - x_i(q')| = |x_i(v) - x_i(p)| - |x_i(v) - x_i(q)|$. Since q beats p, this claim implies that q' beats p', thus finishing the proof. Indeed, if $x_i(p') = x_i(p)$ we have $x_i(q') = x_i(q)$ and the claim holds. Otherwise assume without loss of generality that $x_i(p') < x_i(p)$, and recall that q is ε-close to p. By taking ε sufficiently small we can thus ensure that $\min_i = x_i(p') < x_i(q') < x_i(p) < x_i(q) < \max_i$. Since there are no voters v with $\min_i < x_i(v) < \max_i$, this implies the claim. ◀

4.2 Finding all the plurality points in the personalized L_1 norm

Our algorithm for finding the set of all plurality points is quite simple: we compute the median box M_V in $O(n)$ time, then we check for each vertex of M_V if it is a plurality point (as described in the proof of the theorem below), and finally we report the set of all plurality points using Theorem 21. The following theorem summarizes the result.

▶ **Theorem 22.** *Let V be a set of n voters in $\mathbb{R}^d$, where $d \geqslant 2$ is a fixed constant. Then we can compute in $O(n^{d-1})$ time the set of all plurality points for V in the personalized L_1 norm. When all voters have the same preferences the time bound reduces to $O(n)$.*

Proof. Below we show that we can test if a given vertex p of M_V is a plurality point in $O(n^{d-1})$ time (and in $O(n)$ time if all voters have the same preferences), from which the theorem readily follows.

Assume without loss of generality that p lies at the origin. We need to check if for any competitor q we have $|V[p \succ q]| \geqslant |V[q \succ p]|$. By Lemma 19 we only have to consider non-degenerate competitors. Such a competitor q beats p if $|V[q \succ p]| > n/2$. Because p is at the origin, a voter v is in $V[q \succ p]$ if $\sum_{i=1}^{d} w_i(v) \cdot \Big(|x_i(v) - x_i(q)| - |x_i(v)|\Big) < 0$. When q is an ε-close competitor of p we have $|x_i(q)| < \varepsilon$, and then $|x_i(v) - x_i(q)| - |x_i(v)| \in \{-x_i(q), x_i(q)\}$. Hence, whether or not $v \in V[q \succ p]$ depends on the position of q relative to the hyperplane $h := \sum_{i=1}^{d} w_i(v)\alpha_i x_i = 0$, where $\alpha_i = +1$ if $x_i(q) > x_i(v)$ and $\alpha_i = -1$ if $x_i(q) < x_i(v)$. Each voter $v \in V$ thus generates a set of 2^d hyperplanes. Let H be the total set of hyperplanes generated, that is, $H := \Big\{ \sum_{i=1}^{d} w_i(v)\alpha_i x_i = 0 : v \in V \text{ and } (\alpha_1, \ldots, \alpha_d) \in \{-1, +1\}^d \Big\}$. The discussion above implies that if two competitors q, q' have the same position relative to every hyperplane in H, then $V[q \succ p] = V[q' \succ p]$. Hence, we can proceed as follows.

We first compute the set H in $O(n)$ time. Next we compute the arrangement $\mathcal{A}(H)$ defined by the hyperplanes in H. Since all hyperplanes pass through the origin, $\mathcal{A}(H)$ is effectively a $(d-1)$-dimensional arrangement, so it has complexity $O(n^{d-1})$ and it can be constructed in $O(n^{d-1})$ time [5]. Note that for any cell C of $\mathcal{A}(H)$, we have $V[q \succ p] = V[q' \succ p]$ for any two competitors q, q' in C. With a slight abuse of notation we denote this set by $V[C \succ p]$. The sets $V[C \succ p]$ and $V[C' \succ p]$ for neighboring cells C, C' differ by at most one voter (corresponding to the hyperplane that separates the cells).[1] Hence, we can compute for each cell C of $\mathcal{A}(H)$ the size of $V[C \succ p]$ in $O(n^{d-1})$ time, by performing a depth-first search on the dual graph of $\mathcal{A}(H)$ and updating the size as we step from one cell to the next. When we find a cell C with $|V[C \succ p]| > n/2$ we report that p is not a plurality point, otherwise we report that p is a plurality point.

When all preferences are equal – after appropriate scaling this reduces to the case where we simply use the standard L_1 norm – then all voters $v \in V$ define the same set of 2^d hyperplanes, and so $|H| = 2^d$. Hence, the algorithm runs in $O(n)$ time. ◀

5 Concluding remarks

We presented efficient algorithms for a number of problems concerning plurality points. It would be interesting to generalize these to the setting where the voters have weights – not to

[1] Actually this is not quite true, as several voters could generate the same hyperplane. In this case the difference between $V[C \succ p]$ and $V[C' \succ p]$ can be more than one voter. Thus the time needed to step from C to C' is linear in the number of voters who generate the separating hyperplane of C and C'. It is easy to see that this does not influence the final time bound.

be confused with the weights defining the personal preferences – and a point is a plurality point if there is no other point that is preferred by a set of voters of higher total weight. This would also allow us to deal with multi-sets of voters, something which our current algorithms cannot do. Another direction for future research is to extend our output-sensitive algorithm for the minimum-cost problem and for plurality balls to higher dimensions, and to the personalized L_1 norm.

References

1 H. K. Ahn, S. W. Cheng, O. Cheong, M. Golin, and R. van Oostrum. Competitive facility location: the Voronoi game. *Theor. Comput. Sci.*, 310(1–3):457–467, 2004.

2 T. Chan. An optimal randomized algorithm for maximum tukey depth. In *Proc. 15th ACM-SIAM Symp. Discr. Alg. (SODA)*, pages 430–436, 2004.

3 B. Chazelle. On the convex layers of a planar set. *IEEE Trans. Inf. Theory*, 31(4):509–517, 1985.

4 O. Cheong, S. Har-Peled, N. Linial, and J. Matousek. The one-round voronoi game. *Discr. Comput. Geom.*, 31(1):125–138, 2004.

5 H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.*, 15(2):341–363, 1986.

6 H. A. Eiselt and G. Laporte. Sequential location problems. *Europ. J. Op. Res.*, 96(2):217–231, 1997.

7 J. Erickson. New lower bounds for convex hull problems in odd dimensions. *SIAM J. Comput.*, 28(4):1198–1214, 1999.

8 A. Gajentaan and M. H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, 5(3):165–185, 1995.

9 T. Gallai. Solution to problem number 4065. *The American Mathematical Monthly*, 51(3):169–171, 1944.

10 P. Hansen and J. F. Thisse. Outcomes of voting and planning: Condorcet, weber and rawls locations. *Journal of Public Economics*, 16(1):1–15, 1981.

11 P. Hansen, J. F. Thisse, and R. E. Wendell. Equivalence of solutions to network location problems. *Mathematics of Operations Research*, 11(4):672–678, 1986.

12 S. Jadhav, A. Mukhopadhyay, and B. Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. *J. Alg.*, 20(2):244–267, 1996.

13 D. Kress and E. Pesch. Sequential competitive location on networks. *Europ. J. Op. Res.*, 217(3):483–499, 2012.

14 W. Y. Lin, Y. W. Wu, H. L. Wang, and K. M. Chao. Forming plurality at minimum cost. In *Proc. 9th Int. Workshop Alg. Comput. (WALCOM), LNCS 8973*, pages 77–88, 2015.

15 R. D. McKelvey and R. E. .Wendell. Voting equilibria in multidimensional choice spaces. *Math. Op. Res.*, 1(2):144–158, 1976.

16 J. W. Tukey. Mathematics and the picturing of data. In *In* Proc. Int. Cong. Mathematicians, volume 2, pages 523–531, 1975.

17 R. E. Wendell and R. D. McKelvey. New perspectives in competitive location theory. *Europ. J. Op. Res.*, 6(2):174–182, 1981.

18 R. E. Wendell and S. J. Thorson. Some generalizations of social decisions under majority rule. *Econometrica*, 42(5):893–912, 1974.

19 Y. W. Wu, W. Y. Lin, H. L. Wang, and K. M. Chao. Computing plurality points and condorcet points in euclidean space. In *Proc. 24th Int. Symp. Alg. Comput. (ISAAC), LNCS 8283*, pages 688–698, 2013.

Qualitative Symbolic Perturbation*

Olivier Devillers[1], Menelaos Karavelas[†2], and Monique Teillaud[3]

1 **Inria, Centre de recherche Nancy – Grand Est, France; and**
CNRS, Loria, France; and
Université de Lorraine, France
2 **Mathematics and Applied Mathematics Department, University of Crete, Greece; and**
Institute of Applied and Computational Mathematics, FORTH, Greece
1 **Inria, Centre de recherche Nancy – Grand Est, France; and**
CNRS, Loria, France; and
Université de Lorraine, France

Abstract

In a classical *Symbolic Perturbation* scheme, degeneracies are handled by substituting some polynomials in ε for the inputs of a predicate. Instead of a single perturbation, we propose to use a sequence of (simpler) perturbations. Moreover, we look at their effects geometrically instead of algebraically; this allows us to tackle cases that were not tractable with the classical algebraic approach.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Robustness issues, Symbolic perturbations, Apollonius diagram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.33

1 Introduction

In earlier computational geometry papers, the treatment of degenerate configurations was mainly ignored. However, degenerate situations actually do occur in practice. When data are highly degenerate by nature, a direct handling of special cases in a particular algorithm can be efficient [3]. But in many situations, degeneracies happen only occasionally, and perturbation schemes are an easy and efficient generic solution. Controlled perturbations [11] combine increasing arithmetic precision together with actual displacement of the data, and eventually compute a non-degenerate configuration. On the other hand, the use of a symbolic perturbation allows a geometric algorithm or data structure that was originally designed without addressing degeneracies, to still operate on degenerate cases, without concretely modifying the input [7, 14, 15]. Actually, similar strategies were often used by earlier implementors of simple geometric algorithms, without identifying them as symbolic perturbations: for instance when incrementally computing a convex hull, when the new inserted point was lying on a facet of the convex hull, the point was decided to be inside the convex hull.

Let $G(\boldsymbol{u})$ be a geometric structure defined when the input data $\boldsymbol{u}$ satisfies some non-degeneracy assumptions, and let $\boldsymbol{u}_0$ be some input that is degenerate for G. A *symbolic*

* The work in this paper has been partially supported by the FP7-REGPOT-2009-1 project "Archimedes Center for Modeling, Analysis and Computation".
† M. Karavelas acknowledges support by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALIS – UOA (MIS 375891).


© Olivier Devillers, Menelaos Karavelas, and Monique Teillaud;
licensed under Creative Commons License CC-BY

perturbation consists in using as input $\boldsymbol{u}$ for G a continuous function $\pi(\boldsymbol{u}_0, \varepsilon)$ of a parameter ε. This is done in such a way that for $\varepsilon = 0$, $\pi(\boldsymbol{u}_0, 0)$ is equal to $\boldsymbol{u}_0$, and $\pi(\boldsymbol{u}_0, \varepsilon)$ is non-degenerate for G for sufficiently small positive values of ε. In that case the structure $G(\boldsymbol{u}_0)$ is defined as the limit of $G(\pi(\boldsymbol{u}_0, \varepsilon))$ when $\varepsilon \to 0^+$.

A symbolic perturbation allows an algorithm that computes $G(\boldsymbol{u})$ in generic situations to compute $G(\boldsymbol{u}_0)$ for the degenerate input $\boldsymbol{u}_0$. Most decisions made by the algorithm are usually made by looking at *geometric predicates*, which are combinations of elementary predicates. An elementary predicate is the sign of a continuous real function of the input. The general position assumption is that such a function p never returns 0. When applying a symbolic perturbation, a predicate $\text{sign}(p(\boldsymbol{u}))$ evaluated at $\boldsymbol{u}_0$ returns the limit of $\text{sign}(p(\pi(\boldsymbol{u}_0, \varepsilon)))$ as $\varepsilon \to 0^+$. The sign of $p(\boldsymbol{u}_0)$ can thus be evaluated, provided that $p(\pi(\boldsymbol{u}_0, \varepsilon))$ is not identically equal to 0 for ε in an open interval $(0, a^\star)$ for some $a^\star > 0$. A perturbation scheme is said to be *effective* for a predicate $\text{sign}(p(\boldsymbol{u}))$ if for any $\boldsymbol{u}_0$ the function $\varepsilon \mapsto p(\pi(\boldsymbol{u}_0, \varepsilon))$ is never the null function on any open interval $(0, a)$, with $a \leq a^\star$.

The main difficulty when designing a perturbation scheme for $G(\boldsymbol{u})$ is to find a function $\pi(\boldsymbol{u}_0, \varepsilon)$, such that the perturbation scheme can be proved to be effective for all relevant functions $p(\boldsymbol{u})$, and the perturbed predicates are easy to evaluate, e.g., using as few as possible arithmetic operations. The work of designing and proving the effectiveness of a perturbation for G is typically tailored to a specific algorithm for computing the geometric structure G.

In previous works [1, 6, 7, 8, 12], a predicate is the sign of a polynomial P in some input $\boldsymbol{u} \in \mathbb{R}^m$. The input $\boldsymbol{u}$ is perturbed as an element $\pi(\boldsymbol{u}_0, \varepsilon)$ of $\mathbb{R}^m$ whose coordinates are polynomials in $\boldsymbol{u}_0$ and ε, such that $\pi(\boldsymbol{u}_0, \varepsilon)$ goes to $\boldsymbol{u}_0$ when $\varepsilon \to 0^+$. In the perturbed setting, the predicate returns the sign of the limit $\lim_{\varepsilon \to 0^+} \text{sign}(P(\pi(\boldsymbol{u}_0, \varepsilon)))$. Since P is a polynomial, $P(\pi(\boldsymbol{u}_0, \varepsilon))$ can be rewritten as a polynomial in ε whose monomials in ε are ordered in terms of increasing degree. The constant monomial is actually $P(\boldsymbol{u}_0)$, while the signs of the remaining coefficients can be viewed as auxiliary predicates on $\boldsymbol{u}_0$. The coefficients of $P(\pi(\boldsymbol{u}_0, \varepsilon))$ are evaluated in increasing degrees in ε, until a non-vanishing coefficient is found. The sign of this coefficient is then returned as the value of the predicate $\text{sign}(P(\boldsymbol{u}_0))$.

Contribution

In this paper we propose QSP (Qualitative Symbolic Perturbation), a new framework for resolving degenerate configurations in geometric computing. Unlike classical symbolic perturbation techniques, QSP resolves degeneracies in a purely geometric manner, and independently of a specific algebraic formulation of the predicate. So, the technique is particularly suitable for predicates whose algebraic description is not unique or too complicated, such as the ones treated in this paper. In fact, QSP can even handle predicates that are signs of non-polynomial functions.

In addition, instead of having a single perturbation parameter that governs the way the input objects and/or predicates are modified, QSP allows for a sequence of perturbation parameters: conceptually, we symbolically perturb the input objects one-by-one, using a well-defined canonical ordering that corresponds to considering first the object that is perturbed most. To achieve termination, we must devise an appropriate sequence of perturbations which guarantees that eventually, i.e., after having perturbed sufficiently many input objects, the degenerate predicate is resolved in a non-degenerate manner. The number of objects that need to be perturbed depends on the specific predicate that we analyze. For example in the 2D Apollonius diagram, for a given predicate, perturbing a single object always suffices, whereas in its 3D counterpart, we may need to perturb two input objects.

Standard algebraic symbolic perturbation schemes [7, 8, 12] automatically provide us with the auxiliary predicates that we need to evaluate. These predicates are, by design, of at most the same algebraic degree as the original predicate, but evaluating them in an efficient manner (e.g., by factorizing the predicate) is far from being an obvious task. QSP schemes cannot guarantee that the auxiliary predicates are not more complicated algebraically (i.e., are of lower algebraic degree) from the original predicate; however, in principle, the auxiliary predicates that we have to deal with are expected to be more tractable, since their analysis is based on geometric considerations.

As for any perturbation scheme, QSP assumes exact arithmetic to detect degeneracies. Degeneracies are rare enough to allow high efficiency using the exact geometric computing paradigm [16].

In the next section of the paper we formally define the QSP framework. In Section 3 we describe QSP schemes for the main predicates of the 2D Apollonius diagram. In the full version of the paper [5], we apply our technique to the 3D Apollonius diagram and to the arrangement of circles. We end with Section 4, where we discuss the advantages and disadvantages of our framework, and indicate directions for future research.

2 General framework

Let us start with two easy observations about the limit of the sign of a function of two variables.

2.1 Preliminary observations

The first observation allows us to swap the order of evaluation of limits:

▶ **Observation 1.** *Let f be a continuous function of two variables (a, b) defined in a neighborhood of the origin. Let $\lim_{b\to 0^+} \operatorname{sign} f(0, b)$ be denoted as s. If $s \neq 0$ then*

$$\lim_{b\to 0^+} \lim_{a\to 0^+} \operatorname{sign} f(a, b) = s.$$

Proof. Let us assume that $s \neq 0$. There exists $\delta > 0$ such that $\forall b \in (0, \delta]$, $\operatorname{sign} f(0, b) = s$. For any b fixed in $(0, \delta]$ the function $f(a, b)$ is a continuous function in variable a, thus $\lim_{a\to 0^+} f(a, b) = f(0, b)$ and, since $s \neq 0$, f does not vanish when a is in a neighborhood of 0. We have $\lim_{a\to 0^+} \operatorname{sign} f(a, b) = \operatorname{sign} f(0, b) = s$. For any $b \in (0, \delta]$ the function $\lim_{a\to 0^+} \operatorname{sign} f(a, b)$ is the constant function of value s. So its limit is s when $b \to 0^+$. ◀

The second observation formalizes a situation that is in fact trivial.

▶ **Observation 2.** *Let f be a continuous function in two variables (a, b) defined in a neighborhood of the origin. Assume that $\forall b, b' \geq 0$, $\operatorname{sign} f(a, b) = \operatorname{sign} f(a, b')$. Then*

$$\lim_{b\to 0^+} \lim_{a\to 0^+} \operatorname{sign} f(a, b) = \lim_{a\to 0^+} \operatorname{sign} f(a, 0).$$

Proof. Let $s = \lim_{a\to 0^+} \operatorname{sign} f(a, 0)$. There exists $\delta > 0$ such that $\forall a \in (0, \delta]$, $\operatorname{sign} f(a, 0) = s$. By the hypothesis in the observation we have $\forall a \in (0, \delta]$, $\forall b \geq 0$, $\operatorname{sign} f(a, 0) = \operatorname{sign} f(a, b) = s$. The function has constant sign s on $(0, \delta] \times (0, \infty)$ so the limit is s. ◀

2.2 The QSP scheme

Let $G(\boldsymbol{u})$ be a geometric structure whose computation depends on a predicate $\text{sign}(p(\boldsymbol{u}))$, where p is a continuous real function. In this formal presentation, p appears as a function of the whole input $\boldsymbol{u}$; however, in practice, a predicate depends only on a constant size subset of $\boldsymbol{u}$.

We design the perturbation scheme π as a sequence of successive perturbations π_i, $0 \leq i < N$, with

$$\pi(\boldsymbol{u}, \boldsymbol{\varepsilon}) = \pi_0(\pi_1(\pi_2(\dots \pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1}) \dots, \varepsilon_2), \varepsilon_1), \varepsilon_0),$$

where $\boldsymbol{\varepsilon} = (\varepsilon_0, \varepsilon_1, \varepsilon_2, \dots, \varepsilon_{N-1}) \in \mathbb{R}^N$. The number of perturbations N is part of the perturbation scheme and usually depends on the input size. The perturbations are numbered by increasing order of magnitude, i.e., ε_i is considered much bigger than ε_j if $i > j$. Since $\boldsymbol{\varepsilon}$ is no longer a single real number, we have to determine how the limit is taken; we thus define $G(\boldsymbol{u})$ to be the limit:

$$G(\boldsymbol{u}) = \lim_{\varepsilon_{N-1} \to 0^+} \lim_{\varepsilon_{N-2} \to 0^+} \cdots \lim_{\varepsilon_1 \to 0^+} \lim_{\varepsilon_0 \to 0^+} G(\pi(\boldsymbol{u}, \boldsymbol{\varepsilon})).$$

QSP implies an evaluation strategy of this limit, as follows. The perturbed predicate

$$\lim_{\varepsilon_{N-1} \to 0^+} \lim_{\varepsilon_{N-2} \to 0^+} \cdots \lim_{\varepsilon_1 \to 0^+} \lim_{\varepsilon_0 \to 0^+} \text{sign}(p(\pi(\boldsymbol{u}, \boldsymbol{\varepsilon})))$$

is evaluated by first computing $p(\pi(\boldsymbol{u}, (0, 0, \dots, 0)))) = p(\boldsymbol{u})$, and returning its sign if it is non-zero. If $p(\boldsymbol{u}) = 0$, we look at the function $p(\pi(\boldsymbol{u}, (0, 0, \dots, \varepsilon_{N-1}))) = p(\pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1}))$; if this function is not vanishing when ε_{N-1} lies in a sufficiently small neighborhood to the right of 0, its sign can be returned. More formally, we compute the limit

$$\ell_1 = \lim_{\varepsilon_{N-1} \to 0^+} \text{sign}(p(\pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1}))). \tag{1}$$

If ℓ_1 is non-zero, using Observation 1, it is returned as the value of the predicate $\text{sign}(p(\boldsymbol{u}))$. Otherwise, we have to further perturb our geometric input; we examine the limit

$$\ell_2 = \lim_{\varepsilon_{N-1} \to 0^+} \lim_{\varepsilon_{N-2} \to 0^+} \text{sign}(p(\pi_{N-2}(\pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1}), \varepsilon_{N-2}))). \tag{2}$$

The expression in Eq.(2) can be simplified in cases that actually often occur in applications: if $p(\pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1})))$ is zero on $[0, \eta)$, it is often also the case that the sign of $p(\pi_{N-2}(\pi_{N-1}(\boldsymbol{u}, \varepsilon_{N-1}), \varepsilon_{N-2}))$ does not depend on ε_{N-1} in $[0, \eta)$. In such a case, and provided that this function is also non-zero, we can evaluate its sign using Observation 2: by taking $\varepsilon_{N-1} = 0$, Eq.(2) boils down to

$$\ell_2 = \lim_{\varepsilon_{N-2} \to 0^+} \text{sign}(p(\pi_{N-2}(\boldsymbol{u}, \varepsilon_{N-2}))). \tag{3}$$

The process is iterated until a non-zero limit is found. In very degenerate situations, when the $\mu - 1$ first limits evaluate to 0, i.e., $\ell_1 = \ell_2 = \ldots = \ell_{\mu-1} = 0$, we need to evaluate ℓ_μ. Its definition is

$$\ell_\mu = \lim_{\varepsilon_{N-1} \to 0^+} \lim_{\varepsilon_{N-2} \to 0^+} \dots \lim_{\varepsilon_{N-\mu} \to 0^+} \text{sign}(p(\pi(\boldsymbol{u}, (0, 0, \dots, 0, \varepsilon_{N-\mu}, \varepsilon_{N-\mu+1}, \dots, \varepsilon_{N-1})))).$$

Similarly to what we described for ℓ_2 above, it is frequently the case that the sign of $p(\pi(\boldsymbol{u}, (0, 0, \dots, 0, \varepsilon_{N-\mu+1}, \dots, \varepsilon_{N-1})))$ does not depend on $\varepsilon_{N-\mu+1}, \dots, \varepsilon_{N-1}$ in a neighborhood of 0 in $\mathbb{R}^{\mu-1}$; then the simplified evaluation allowed by Observation 2 gives:

$$\ell_\mu = \lim_{\varepsilon_{N-\mu} \to 0^+} \text{sign}(p(\pi(\boldsymbol{u}, (0, 0, \dots, 0, \varepsilon_{N-\mu}, \varepsilon_{N-\mu+1}, \dots, \varepsilon_{N-1})))).$$

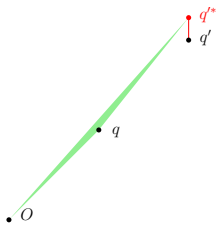
To assert that the perturbation scheme π is effective, we need to prove that one of these limits is indeed non-zero.

When the predicate is a polynomial, we get a sequence of successive evaluations as in algebraic symbolic perturbations; however, the expressions that need to be evaluated have been obtained in a different way and are *a priori* different. The main advantage of this approach is that we may use a very simple perturbation π_ν, since we do not need each perturbation π_ν to be effective, but rather the composed perturbation π. For geometric problems, the simplicity of π_ν allows us to look at the limit in a geometric manner, instead of algebraically computing some appropriate coefficient of $p(\pi(\boldsymbol{u}, \varepsilon))$.

2.3 Toy examples

We illustrate these principles with three toy examples. In all examples, we set $\boldsymbol{u} = (q, q') = ((x_0, x_1), (x_2, x_3))$, a pair of two 2D points and $\pi_i(\boldsymbol{u}, \varepsilon_i) = \boldsymbol{u} + \varepsilon_i \boldsymbol{e}_i$, where $\boldsymbol{e}_0 = ((1,0),(0,0))$, $\boldsymbol{e}_1 = ((0,1),(0,0))$, $\boldsymbol{e}_2 = ((0,0),(1,0))$, and $\boldsymbol{e}_3 = ((0,0),(0,1))$ form the canonical basis of $(\mathbb{R}^2)^2$. The differences between the examples below lie in the evaluated predicate $\mathrm{sign}(p(\boldsymbol{u}))$ and the degenerate position $\boldsymbol{u}_0$.

First example: orientation of a flat triangle



Let $p(\boldsymbol{u}) = x_0x_3 - x_1x_2$ and $\boldsymbol{u}_0 = (q, q') = ((1,1),(2,2))$. QSP defines the result for $\mathrm{sign}(p(\boldsymbol{u}_0))$, the orientation of Oqq', as

$$\mathrm{sign}(p(\boldsymbol{u}_0)) = \lim_{\varepsilon_3\to 0^+} \lim_{\varepsilon_2\to 0^+} \lim_{\varepsilon_1\to 0^+} \lim_{\varepsilon_0\to 0^+} \mathrm{sign}((1+\varepsilon_0)(2+\varepsilon_3) - (2+\varepsilon_1)(1+\varepsilon_2)).$$

A standard evaluation of this expression would consist in taking the limits in order:

$$\begin{aligned}\mathrm{sign}(p(\boldsymbol{u}_0)) &= \lim_{\varepsilon_3\to 0^+} \lim_{\varepsilon_2\to 0^+} \lim_{\varepsilon_1\to 0^+} \mathrm{sign}((2+\varepsilon_3) - (2+\varepsilon_1)(1+\varepsilon_2)) \\ &= \lim_{\varepsilon_3\to 0^+} \lim_{\varepsilon_2\to 0^+} \mathrm{sign}((2+\varepsilon_3) - 2(1+\varepsilon_2)) = \lim_{\varepsilon_3\to 0^+} \mathrm{sign}(\varepsilon_3) = 1.\end{aligned}$$

Following the QSP evaluation strategy instead, in such a case, the biggest perturbation, i.e., the perturbation on x_3, allows to quickly conclude. The only computed limit is the one in Eq. (1):

$$\ell_1 = \lim_{\varepsilon_3\to 0^+} \mathrm{sign}(p((1,1),(2,2+\varepsilon_3))) = \lim_{\varepsilon_3\to 0^+} \mathrm{sign}((2+\varepsilon_3) - 2) = \lim_{\varepsilon_3\to 0^+} \mathrm{sign}(\varepsilon_3) = 1.$$

The geometric interpretation is that we get the orientation of a triangle Oqq'^* for a point q'^* slightly above q'.

Second example: orientation of a vertical flat triangle

Let $p(\boldsymbol{u}) = x_0x_3 - x_1x_2$ and $\boldsymbol{u}_0 = (q, q') = ((0,1),(0,2))$. QSP defines the result for $\operatorname{sign}(p(\boldsymbol{u}_0))$, the orientation of Oqq', as

$$\operatorname{sign}(p(\boldsymbol{u}_0)) = \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\lim_{\varepsilon_1\to 0^+}\lim_{\varepsilon_0\to 0^+}\operatorname{sign}((0+\varepsilon_0)(2+\varepsilon_3)-(1+\varepsilon_1)(0+\varepsilon_2)).$$

Taking the limits in order leads to:

$$\begin{aligned}\operatorname{sign}(p(\boldsymbol{u}_0)) &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\lim_{\varepsilon_1\to 0^+}\operatorname{sign}(-(1+\varepsilon_1)\varepsilon_2)\\ &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\operatorname{sign}(-\varepsilon_2) = \lim_{\varepsilon_3\to 0^+}\operatorname{sign}(-1) = -1.\end{aligned}$$

In this case, the QSP evaluation strategy is to first compute

$$\ell_1 = \lim_{\varepsilon_3\to 0^+}\operatorname{sign}(p((0,1),(0,2+\varepsilon_3))) = \lim_{\varepsilon_3\to 0^+} 0 = 0,$$

which does not allow us to resolve the degeneracy. Then we observe that

$$\operatorname{sign}(p((0,1),(x_2,x_3)) = \operatorname{sign}(-x_2)$$

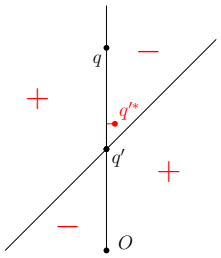
does not depend on x_3, thus we can evaluate ℓ_2 using Eq. (3):

$$\ell_2 = \lim_{\varepsilon_2\to 0^+}\operatorname{sign}(p((0,1),(\varepsilon_2,2))) = \lim_{\varepsilon_2\to 0^+}\operatorname{sign}(-\varepsilon_2) = -1.$$

Two perturbations π_3 and π_2 must be used, but the simplified evaluation of ℓ_2 suffices.

The geometric interpretation is that we look at the orientation of a triangle Oqq'^* for a moved point q'^*. Since moving q'^* slightly above q' doesn't change anything to the degeneracy, the point is moved to the right, which resolves the degeneracy.

Third example: points and quadratic form



Let $p(\boldsymbol{u}) = x_0(x_1-1) - x_0^2 - x_2(x_3-1) + x_2^2$ and $\boldsymbol{u}_0 = (q,q') = ((0,2),(0,1))$. The predicate p stands for the difference of a degenerate quadratic form evaluated at q and q'. QSP defines the result for $\operatorname{sign}(p(\boldsymbol{u}_0))$ as

$$\operatorname{sign}(p(\boldsymbol{u}_0)) = \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\lim_{\varepsilon_1\to 0^+}\lim_{\varepsilon_0\to 0^+}\operatorname{sign}\left(\varepsilon_0(1-2+\varepsilon_1) - \varepsilon_0^2 - \varepsilon_2(1-1+\varepsilon_3) + \varepsilon_2^2\right),$$

which could be evaluated as follows:

$$\begin{aligned}\operatorname{sign}(p(\boldsymbol{u}_0)) &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\lim_{\varepsilon_1\to 0^+}\operatorname{sign}\left(-\varepsilon_2\varepsilon_3+\varepsilon_2^2\right)\\ &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\operatorname{sign}\left(\varepsilon_2(\varepsilon_2-\varepsilon_3)\right) = \lim_{\varepsilon_3\to 0^+}\operatorname{sign}(-\varepsilon_3) = -1.\end{aligned}$$

Again the evaluation strategy first computes

$$\ell_1 = \lim_{\varepsilon_3\to 0^+}\operatorname{sign}(p((0,2),(0,1+\varepsilon_3))) = \lim_{\varepsilon_3\to 0^+} 0 = 0,$$

which does not allow us to resolve the degeneracy.

Then we observe that $\operatorname{sign}(p((0,2),(x_2,x_3))) = \operatorname{sign}(x_2(x_3-1)+x_2^2)$ actually depends on x_3, thus we must evaluate ℓ_2 using Eq. (2):

$$\begin{aligned}\ell_2 &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\operatorname{sign}(p((0,2),(\varepsilon_2,1+\varepsilon_3)))\\ &= \lim_{\varepsilon_3\to 0^+}\lim_{\varepsilon_2\to 0^+}\operatorname{sign}\left(\varepsilon_2(\varepsilon_2-\varepsilon_3)\right) = \lim_{\varepsilon_2\to 0^+}\operatorname{sign}(-\varepsilon_3) = -1.\end{aligned}$$

Notice that since $\operatorname{sign}(p((2,0),(x_2,x_3)))$ depends on x_3, the simplified evaluation of Eq. (3) would have given a wrong result:

$$\ell_2 \neq \lim_{\varepsilon_2\to 0^+}\operatorname{sign}(p((2,0),(\varepsilon_2,1))) = \lim_{\varepsilon_2\to 0^+}\operatorname{sign}(\varepsilon_2^2) = 1.$$

The geometric interpretation is that q and q' are both on one of the two lines defined by the quadratic equation $x(y-1)-x^2=0$. Point q' is first slightly moved upwards but this motion leaves it on that same line, then it is moved to the right, and the sign of the quadratic form depends on the vertical position of q' with respect to the other line.

A fourth toy example illustrating a non-polynomial predicate is given in the full version [5].

2.4 Discussion

Multiple epsilons

The idea of utilizing multiple perturbation parameters is already present in Yap's scheme [14], or very recently in Irving and Green's work [10], but without the geometric interpretation allowed by QSP. In other previous works, such as SoS [7], the algebraic symbolic perturbation framework was proved to be effective by a careful choice of the exponents for ε, depending on the choice of $G(\boldsymbol{u})$, so as to make some terms negligible. QSP can be forced to fit in such a traditional framework, with a single epsilon, by making all the variables ε_ν dependent on a single parameter κ that plays the traditional role of ε. For polynomial predicates, it is enough to take ε_ν exponentially increasing with respect to ν. For example one such choice can be to set $\varepsilon_{N-1}=\kappa$, and $\varepsilon_\nu = \left(\exp\left(\frac{1}{\varepsilon_{\nu+1}}\right)\right)^{-1}$, for $0\le\nu<N-1$. The interest of QSP, however, is not to use this traditional view, but rather have the variables ε_ν independent; this decoupling allows for additional flexibility, and, in particular, permits us to think of the sequence of perturbations in geometric terms.

Efficiency

The aim of a perturbation scheme is to solve degeneracies, and a common assumption is that such degeneracies are rare enough so that some extra time can be spent to make a reliable decision when a degeneracy happens. Another implicit assumption is that degeneracies are actually detected, that is, it is implicitly assumed that the original predicates are computed exactly, possibly with some filtering mechanism to ensure efficiency [16].

Nevertheless, the actual additional complexity in case of degeneracy must be addressed. Since QSP is geometrically defined and addresses very general problems, such a complexity analysis cannot be done at the general level. For the two applications described in this paper, the extra predicates needed to resolve the degeneracy have the same complexity as the original ones, while the number of epsilons used to perturb is not bigger than two.

QSP, as many other perturbation schemes, relies on an indexing of the input. However, as mentioned earlier, a given predicate usually depends on a constant number of input objects. It is important to keep in mind that the comparison of indices is necessary only for the few objects involved in a given predicate; sorting the whole input with respect to indices is not required.

Generality

In the first three toy examples above, SoS would have taken $\varepsilon_0 = \varepsilon^8$, $\varepsilon_1 = \varepsilon^4$, $\varepsilon_2 = \varepsilon^2$, and $\varepsilon_3 = \varepsilon$, which yields the same result as QSP. When it leads to a simple result, the classical algebraic view is a very good solution. However, if the original predicate is a bit intricate, the algebraic way will produce numerous extra predicates to resolve degeneracies. Moreover, as for any predicate, some custom work is often still needed on the polynomial to evaluate it efficiently, e.g., finding a good factorization.

QSP provides a very general approach that is able to handle various predicates, even non-polynomial (as in the fourth example described in [5]). Of course applying this scheme to a given problem requires some problem-specific work, but, as noted in the previous paragraph, this is also often the case for the above-mentioned algebraic approaches. In algebraic approaches, obtaining the coefficient of ε^i in a suitable way for an efficient evaluation is a non-trivial task; the task is even harder when the predicate does not boil down to evaluating a single polynomial, as it is the case for Apollonius predicates, which we present in the sequel.

We would not advise the use of QSP for simple cases such as Delaunay triangulations of points where other approaches work well [6], but rather only in cases where the predicates are very complex or non-polynomial. The applications below use high degree polynomials and QSP is a good solution. As far as we know, no other perturbation scheme has ever been proposed for Apollonius diagrams. Regarding intersections of circles, we successfully addressed the predicate comparing the abscissas of intersection of circles, using QSP [5]. The only other result that we know of for perturbing this predicate was recently obtained by Irving and Green [10]; it uses a pseudo-random scheme.

Meaningfulness

According to a classification by Seidel [13], QSP is (geometrically) meaningful, that is we have some control on the direction (in input data space) used to move away from the degeneracies. For example, for the Apollonius diagram we will choose to minimize the number of Apollonius vertices (it is also possible to choose to maximize it). QSP is not independent of indexing, but if this indexing is geometrically meaningful, then we can ensure invariance with respect to some geometric transformations.

3 The Apollonius diagram

3.1 Definition

The *Apollonius diagram*, also known as *additively weighted Voronoi diagram*, is defined on a set of weighted points in the Euclidean space $\mathbb{R}^d$. In the formalism of Section 2, $\boldsymbol{u}$ is a vector

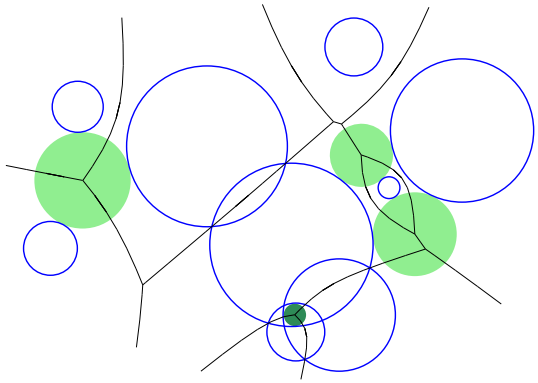


Figure 1 Planar Apollonius diagram. Weighted points are in light blue. A green disk is centered at an Apollonius vertex and its radius is the weighted distance of the center to its three closest sites. The darker green disk has a negative distance to its closest neighbors.

of coordinates and weights of a set of weighted points and $G(\boldsymbol{u})$ is the Apollonius diagram; in this section, we will use notations more adapted to our application. The Euclidean norm is denoted as $|\cdot|$. The *weighted distance* from a query point r to a weighted point (s, w), where s is a point in the Euclidean space and $w \in \mathbb{R}$, is $|rs| - w$. The Apollonius diagram is the closest point diagram for this distance. It generalizes the Voronoi diagram, defined on non-weighted points.

Given a set of weighted points, also called *sites*, it is clear that adding the same constant to all weights does not change the Apollonius diagram. Thus, in the sequel, we may freely translate the weights to ensure, for example, that all weights are positive, or that a particular weight is zero. A site $(s, w), w \geq 0$, can be identified with the sphere S centered at s and of radius w. The distance from a query point r to a site $S = (s, w)$ is the Euclidean distance from r to S, with a negative sign if r lies inside S.

An Apollonius vertex v is a point at the same distance from $d+1$ sites $S_0, S_1, \ldots, S_d$ in general position. We call the configuration *external* if v is outside sphere S_i, for all $i = 0, \ldots, d$, and *internal* if it is inside the spheres. If the configuration is external (resp., internal), v is the center of a sphere externally (resp., internally) tangent to the sites S_i (see green (resp., dark green) disks in figure above). It is always possible to ensure an external configuration locally by adding a suitable constant to the weights of all S_i, such that all weights are non-negative, while the smallest among them is equal to zero.

Let us show that $d+1$ sites in general position define 0, 1 or 2 Apollonius vertices. Assume, without loss of generality, that all weights are non-negative for $i = 1, \ldots, d$ and $w_0 = 0$, so as to have an external configuration. Consider now the inversion with point s_0 as the pole. The point s_0 goes to infinity, while each sphere S_i, $i = 1, \ldots, d$ becomes a new sphere $Z_i = (z_i, \rho_i)$. Determining the balls B_α (where α indexes the different solutions) tangent to the spheres S_i, $i = 0, \ldots, d$ is equivalent to determining halfspaces delimited by the hyperplanes T_α tangent to the spheres Z_i, $i = 1, \ldots, d$, with all spheres on the same side of T_α. Requiring that a given B_α is externally tangent to the spheres S_i is equivalent to requiring that T_α separates the spheres Z_i from the origin. The normalized equation of T_α: $\lambda_\alpha \cdot x + \delta_\alpha = 0$, with $\lambda_\alpha \in \mathbb{R}^d, |\lambda_\alpha| = 1$ and $\delta_\alpha \in \mathbb{R}$, gives the signed distance of a point $x \in \mathbb{R}^d$ to T_α. We have

$$T_\alpha \text{ tangent to } Z_i,\ 1 \leq i \leq d \iff \begin{cases} \lambda_\alpha \cdot z_i + \delta_\alpha = \rho_i,\ 1 \leq i \leq d \\ |\lambda_\alpha|^2 = 1 \end{cases}. \tag{4}$$

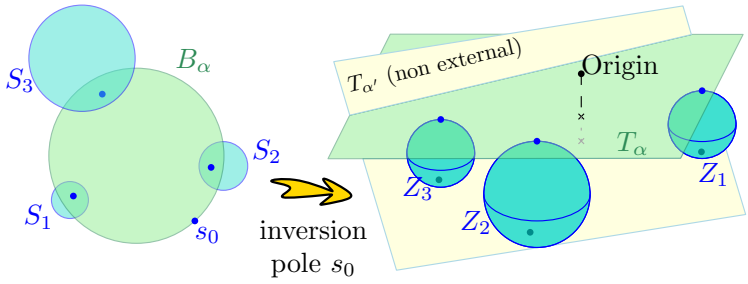


Figure 2 The spheres B_α externally tangent to the sites S_i, $i = 0, \ldots, d$, correspond, via the inversion transformation with s_0 as the pole, to hyperplanes T_α tangent to the spheres Z_i that separate them from the origin.

In the inverted space, the general position hypothesis means that the spheres Z_i do not have an infinity of tangent hyperplanes; the latter can occur only if the points z_i (and thus the points s_i) are affinely dependent. Therefore, the system (4) of one quadratic and d linear equations in $d+1$ unknowns ($\lambda_\alpha \in \mathbb{R}^d$ and $\delta_\alpha \in \mathbb{R}$) has at most two real solutions by Bézout's theorem, hence the first claim follows. Depending on the position of the origin with respect to T_α (or equivalently on the sign of δ_α), zero, one or both solutions may correspond to external configurations.

An Apollonius vertex is actually defined by a sequence of $d+1$ sites in general position, up to a positive permutation of the sequence. Indeed, in the previous paragraph, if there are two solutions T_α and $T_{\alpha'}$, we observe that they are symmetric with respect to the hyperplane spanned by the points z_i, thus the d-simplex formed by the tangency points and the origin has different orientations for the two solutions. This implies that the two solutions can be distinguished by the signature of the permutation of the spheres S_i.

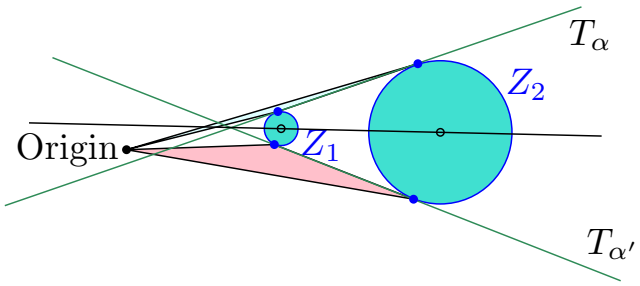


Figure 3 If T_α and $T_{\alpha'}$ are both external, the simplices formed by the tangency points and the origin have different orientations.

3.2 The VConflict predicate

Several predicates are necessary to compute an Apollonius diagram. We start with the *vertex conflict* predicate $\mathsf{VConflict}(\boldsymbol{S}^v, Q)$, which answers the following question:

> *Does an Apollonius vertex v defined, up to a positive permutation, by a $(d+1)$-tuple of sites $\boldsymbol{S}^v = (S_{i_0}, S_{i_1}, \ldots, S_{i_d})$ remain as a vertex of the diagram after another site Q is added?*

If the site centered at v and tangent to the sites of the tuple $\boldsymbol{S}^v$ is in internal configuration, we can add a negative constant to the radii of all spheres in $\boldsymbol{S}^v \cup \{Q\}$ so that the smallest

site in $\boldsymbol{S}^v$ has zero radius. Then the configuration of the common tangent sphere becomes external. In this manner, we can always restrict our analysis to the case where the Apollonius vertex we consider is in external configuration. Note that this may lead to a negative weight w_q for Q, which was *a priori* excluded above, but is treated below.

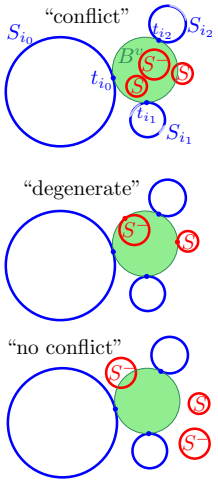
We denote by $B_{i_0 i_1 \ldots i_d}$ the open ball whose closure $\overline{B}_{i_0 i_1 \ldots i_d}$ is tangent to the sites of $\boldsymbol{S}^v$. The contact points $t_{i_0}, t_{i_1}, \ldots, t_{i_d}$ define a positively oriented d-simplex.

If $w_q \geq 0$, the predicate $\mathsf{VConflict}(\boldsymbol{S}^v, Q)$ answers

- "conflict" if Q intersects $B_{i_0 i_1 \ldots i_d}$,
- "no conflict" if Q and $\overline{B}_{i_0 i_1 \ldots i_d}$ are disjoint,
- "degenerate" if Q and $B_{i_0 i_1 \ldots i_d}$ do not intersect, while Q and $\overline{B}_{i_0 i_1 \ldots i_d}$ are tangent.

If $w_q < 0$, we define Q^- as the sphere with the same center s_q as Q and radius $-w_q$. Then $\mathsf{VConflict}(\boldsymbol{S}^v, Q)$ answers

- "conflict" if Q^- is included in $B_{i_0 i_1 \ldots i_d}$,
- "no conflict" if Q^- intersects the complement of $\overline{B}_{i_0 i_1 \ldots i_d}$,
- "degenerate" if Q^- is included in $\overline{B}_{i_0 i_1 \ldots i_d}$ and is tangent to its boundary.



Qualitative perturbation of the VConflict predicate

QSP relies on some ordering of the sites. Each site $S_\nu = (s_\nu, w_\nu)$ is perturbed to $S_\nu^\varepsilon = (s_\nu, w_\nu + \varepsilon_\nu)$, $\varepsilon_\nu \geq 0$, with S_λ perturbed more than S_ν if $\lambda > \nu$. Following the QSP framework, if the configuration is still degenerate after we have enlarged the site of maximum index, then we enlarge the site with the second largest index, and so on. As mentioned in the general presentation (Section 2.4), we need only consider the sites involved in the predicate, and enlarge them one-by-one until the resulting configuration is non-degenerate, in which case the predicate is resolved. Sites are sorted internally in the predicate, among a constant number of objects; there is no need for sorting the sites, with respect to their index, globally.

Any indexing can be used. We choose what we call the *max-weight* indexing that assigns a larger index to the site with larger weight. As a result, a site with larger weight is perturbed more, and in order to resolve the predicate we need to consider the sites in order of decreasing weights, until the degeneracy is resolved. To break ties between sites with the same weights, we use the lexicographic comparison of their centers: among two sites with the same weight, the site whose center is lexicographically smaller than the other is

assigned a smaller max-weight index. The max-weight indexing has the strong advantage of being geometrically meaningful. It favors sites with larger weights, so, if two sites are internally tangent, then the site with the larger weight will be perturbed more, in which case the site with the smallest weight will be inside the interior of the other site, and its Apollonius region will disappear in the perturbed diagram. As a first consequence, this indexing minimizes the number of Apollonius regions in the diagram, or, equivalently it maximizes the number of hidden sites in the diagram. Secondly, and most importantly, the tangency points of the sites with the Apollonius sites that they define in the diagram are pairwise distinct. This property makes the analysis of the perturbed predicates much simpler, whereas the Apollonius diagram computed does not exhibit pathological cases, such as Apollonius regions with empty interiors. Some inevitable degenerate constructions, such as zero-length Apollonius edges, are handled seamlessly by the method. As a final comment, the max-weight scheme can be used to resolve the degeneracies of all predicates described by Emiris and Karavelas for the 2D case [9].

3.3 Perturbing circles for the 2D Apollonius diagram

In two dimensions, the two main predicates for computing Apollonius diagrams are the VConflict predicate introduced in the previous section and the EdgeConflict predicate. Predicate EdgeConflict is presented in the full version of the paper [5].

Given S_i, S_j, S_k the three sites that define an Apollonius circle in the Apollonius diagram and $Q = S_q$ the query site, the algebraic formulation of predicate VConflict is a polynomial in the coordinates and weights of the four spheres. Emiris and Karavelas [9] proposed a degree 14 expression that can be factorized to reduce the degree to 8 [5, Appendix B].

Before using our qualitative symbolic perturbation framework to design the perturbed predicate, we briefly sketch how a standard algebraic perturbation framework could be applied.

3.3.1 Algebraic perturbation of the 2D VConflict predicate

If $S_\nu = (x_\nu, y_\nu, w_\nu)$ is perturbed in $S_\nu^\varepsilon = (x_\nu, y_\nu, w_\nu + \varepsilon_\nu)$ for $\nu \in \{i, j, k, q\}$, then developing the algebraic polynomial involved in VConflict will give a polynomial in ε_i, ε_j, ε_k and ε_q with hundreds of terms (see full version [5] for more details). Assigning ε_i, ε_j, ε_k and ε_q to be polynomial functions of a single variable ε (for example, we may set $\varepsilon_\nu = \varepsilon^{\alpha_\nu}$, $\nu \in \{i, j, k, q\}$) transforms the expression to a univariate polynomial in ε. When performing such an assignment, either some of the terms collapse making their geometric and algebraic interpretation difficult, or α_i, α_j, α_k and α_q have to be chosen carefully so that the coefficients of the various monomials of the variable ε_ν in the resulting polynomial do not collapse. Even if one could find an assignment that does not make the coefficients (of the originally different terms) collapse, we are still faced with the problem of analyzing the monomials, and, by employing algebraic and/or geometric arguments, showing that there is at least one coefficient of the polynomial that does not vanish.

3.3.2 Qualitative perturbation of the 2D VConflict predicate

We now precisely describe how the perturbation works on the VConflict predicate in dimension 2. Let us denote by q the max-weight index of Q, i.e., $Q = S_q$. We denote with superscript ε the perturbed version of objects, that is B_{ijk}^ε is a shorthand for the ball tangent

to S_i^ε, S_j^ε, and S_k^ε, and

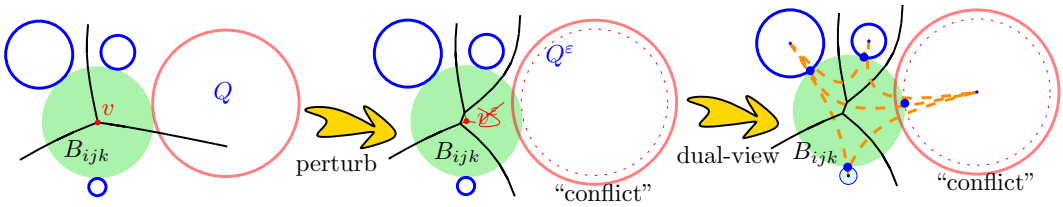
$$\mathsf{VConflict}^\varepsilon(S_i,S_j,S_k,S_q) = \lim_{\varepsilon_{i_3}\to 0^+}\lim_{\varepsilon_{i_2}\to 0^+}\lim_{\varepsilon_{i_1}\to 0^+}\lim_{\varepsilon_{i_0}\to 0^+}\mathsf{VConflict}(S_i^\varepsilon,S_j^\varepsilon,S_k^\varepsilon,Q^\varepsilon)$$

with $i_0 < i_1 < i_2 < i_3$, $\{i_0,i_1,i_2,i_3\} = \{i,j,k,q\}$.

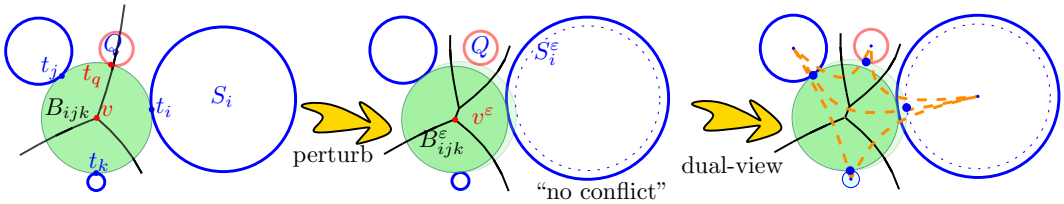
If $q > i,j,k$ and $\mathsf{VConflict}\ (S_i,S_j,S_k,Q)$ = "degenerate", we compute the limit given by Eq. (1)

$$\lim_{\varepsilon_q\to 0^+}\mathsf{VConflict}(S_i,S_j,S_k,Q^\varepsilon).$$

It is clear that this limit always evaluates to "conflict", since Q is growing while the open ball B_{ijk} whose closure is tangent to S_i, S_j, and S_k can be considered as fixed and we do not need to look at perturbations of smaller index.

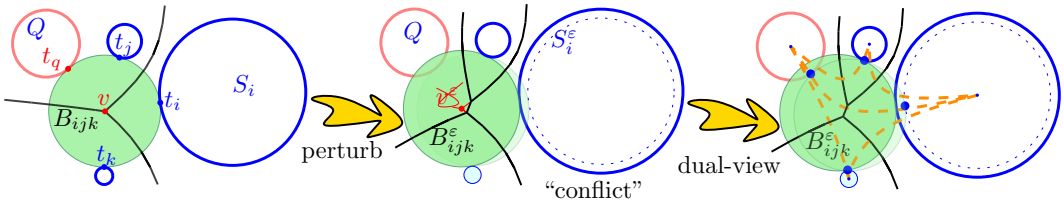


If q is not the largest index, then B_{ijk} can be viewed as defined by three other circles among S_i, S_j, S_k, and Q. Since $B_{ijk} = B_{jki} = B_{kij}$, we can assume, without loss of generality, that $i > j,k,q$. Moreover, B_{ijk} coincides with either B_{jkq} or B_{kjq}, depending on the orientation of the tangency points of S_j, S_k and Q with B_{ijk}.



In the perturbed setting, S_i^ε is in conflict with B_{jkq}^ε (or B_{kjq}^ε) since S_i^ε is growing, while B_{jkq}^ε can be considered as fixed. We simply need to determine if B_{ijk}^ε remains empty in the perturbed setting. Let t_ν (resp., t_q) be the tangency point of S_ν (resp., Q) with B_{ijk}, $\nu \in \{i,j,k\}$, and notice that $t_it_jt_k$ is a ccw triangle. We consider three cases depending on the position of t_q on ∂B_{ijk}. If t_q is different from t_i, t_j, and t_k, the four points form a convex quadrilateral. When perturbing S_i to become S_i^ε, the Apollonius vertex is split in two, which, in the dual,[1] corresponds to a triangulation of the quadrilateral with vertices S_i, S_j, S_k, S_q. Since S_i is the most perturbed circle, the quadrilateral will be triangulated by linking S_i to the other three vertices. If t_q is on the same side as t_i with respect to the line t_jt_k, then the triangulation contains triangle $S_iS_jS_k$ and, therefore, Q is not in conflict with B_{ijk} (see figure above), otherwise $S_iS_jS_k$ is not in the triangulation and Q has to be in conflict with B_{ijk} (see figure below).

[1] The dual of the Apollonius diagram is called Apollonius graph. The Apollonius region of S_i is associated to a vertex of the dual graph, thus S_i can be used to refer to the corresponding vertex in the dual graph.



If t_q is equal to t_i then, since $i > q$, Q is internally tangent to S_i and there is no conflict (S_i^ε contains Q in its interior, and thus Q has empty Apollonius region in the diagram). If t_q is equal to t_ν with $\nu \in \{j, k\}$ then either Q is internally tangent to S_ν, or S_ν is internally tangent to Q. In the former case, Q does not intersect the perturbed Apollonius disk B_{ijk}^ε and thus the result of the perturbed predicate is "no conflict"; in the latter case, Q intersects B_{ijk}^ε, and the perturbed predicate returns "conflict". Hence, in the case $t_q = t_\nu$, $\nu \in \{j, k\}$, the perturbed predicate returns "conflict" if an only if $q > \nu$.

3.3.3 Practical evaluation of the 2D VConflict$^\varepsilon$ predicate

Following the analysis in the previous section, $\mathsf{VConflict}^\varepsilon(S_i, S_j, S_k, Q)$ can be evaluated by the following procedure:

1. **if** $\mathsf{VConflict}(S_i, S_j, S_k, Q) \neq$ "degenerate" **then return** $\mathsf{VConflict}(S_i, S_j, S_k, Q)$;
2. **if** $q > \max\{i, j, k\}$ **then return** "conflict";
3. ensure that $i > \max\{j, k\}$ by a cyclic permutation of (i, j, k);
4. **if** $t_q = t_i$ **then return** "no conflict";
5. **if** $t_q = t_j$ **then** { **if** $q > j$ **then return** "conflict"; **else return** "no conflict"; };
6. **if** $t_q = t_k$ **then** { **if** $q > k$ **then return** "conflict"; **else return** "no conflict"; };
7. **if** $t_j t_k t_q$ is ccw **then return** "no conflict"; **else return** "conflict";

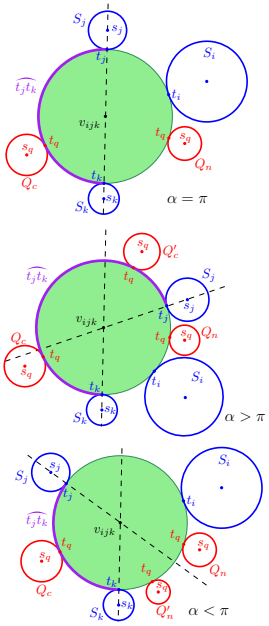
Step 1 is the unperturbed predicate and is evaluated algebraically, e.g., as described in [9, 5]. Steps 2 and 3 amount to sorting the indices of the four sites and determining if q is the largest, or, if this is not the case, finding the largest index. At Step 4, we already know that $i > q$, which implies that $w_i \geq w_q$, and hence the only possibility is that Q is internally tangent to S_i. So, in order to perform Step 4, we simply look at $p_q^* = (x_q - x_i)^2 + (y_q - y_i)^2 - (w_q - w_i)^2$: if $p_q^* = 0$, return "no conflict", otherwise continue with Step 5. Steps 5 and 6 can be resolved in a similar way: if $(x_q - x_\nu)^2 + (y_q - y_\nu)^2 - (w_q - w_\nu)^2 = 0$, then if $q > \nu$ (resp., $q < \nu$), we return "conflict" (resp., "no conflict"). Otherwise, we continue with the last step of the procedure.

We will now focus on this last step, Step 7, because it introduces a new geometric predicate, which is difficult to evaluate: $\mathsf{Orientation}(t_j, t_k, t_q)$, for three tangency points. Our aim is to reduce the complexity of the expressions to be evaluated, which is why we avoid computing the tangency points explicitly. The end of this section describes a method with algebraic degree 8, as in Step 1. This computation can be done in another way: in [5, Appendix A] we proposed an alternative method that requires very few arithmetic computations besides the quantities already computed in the unperturbed evaluation of Step 1, however these few computations have algebraic degree 12.

It has been shown in [9] that evaluating the orientation of three points where two are centers of sites and the third is an Apollonius vertex, can be performed using algebraic expressions of degree at most 14. In fact, this degree may be decreased to 8 (see [5, Appendix B]), in which case we resolve $\mathsf{Orientation}(t_j, t_k, t_q)$, without resorting to a higher degree predicate, as described below.

Firstly, we evaluate $o_1 = \mathsf{Orientation}(s_j, v_{ijk}, s_k)$, where v_{ijk} is the center of the Apollonius circles B_{ijk} of the three sites S_i, S_j, S_k. We perform this evaluation in order to determine whether the angle α of the ccw arc $\overset{\frown}{t_j t_k}$ on B_{ijk} is more or less than π. Secondly, we distinguish between the following cases:

- o_1 = "collinear". In this case $\alpha = \pi$, and the line through t_j and t_k coincides with the line through s_j and s_k. Hence: $\mathsf{Orientation}(t_j, t_k, t_q) = \mathsf{Orientation}(s_j, s_k, s_q)$ (see Q_n (resp., Q_c) in the figure (top) to illustrate a position of Q not in conflict (resp., in conflict)).
- o_1 = "ccw". In this case $\alpha > \pi$. We start by evaluating $o_2 = \mathsf{Orientation}(s_j, v_{ijk}, s_q)$. If $o_2 \neq$ "ccw" (see Q'_c in the figure (middle)), t_q lies to the right of the line through t_j and t_k, and thus $\mathsf{Orientation}(t_j, t_k, t_q) =$ "cw". Otherwise, we need to evaluate the orientation $o_3 = \mathsf{Orientation}(v_{ijk}, s_k, s_q)$; then $\mathsf{Orientation}(t_j, t_k, t_q) =$ "ccw" if and only if $o_3 =$ "ccw" (see Q_n and Q_c in the figure (middle)).
- o_1 = "cw". In this case $\alpha < \pi$. We start by evaluating $o_2 = \mathsf{Orientation}(s_j, v_{ijk}, s_q)$. If $o_2 \neq$ "cw", t_q lies to the left of the line through t_j and t_k, and thus $\mathsf{Orientation}(t_j, t_k, t_q) =$ "ccw" (see Q_n in the figure (bottom)). Otherwise, we need to evaluate the orientation $o_3 = \mathsf{Orientation}(v_{ijk}, s_k, s_q)$; then $\mathsf{Orientation}(t_j, t_k, t_q) =$ "cw" if and only if $o_3 =$ "cw" (see Q_c and Q'_n in the figure (bottom)).



To summarize, the evaluation of Step 7 requires at most three orientation tests involving an Apollonius vertex and two sites; one may be obtained as a subproduct of Step 1, while the other two require work similar to the work performed for Step 1. Thus, the evaluation of Step 7 does not increase the algebraic degree of the VConflict predicate.

4 Conclusion

In this extended abstract, a new framework for dealing with geometric degeneracies has been proposed: QSP, and its application to a predicate used in the computation of the 2D Apollonius diagram. In the full paper we extend the result to the 3D Apollonius diagram and to the computation of arrangement of arcs of circles in the plane. All these predicates are predicates of medium degree (8 to 28) and have complicated algebraic expressions that

make them difficult to combine with classical algebraic perturbation schemes. Conversely to usual approaches for symbolic perturbation, the new framework does not rely on a particular algebraic description of the predicate, but rather directly on its geometric description.

A QSP scheme consists of a sequence of perturbations, but given a specific predicate only a few of these perturbations are really *active*. The number of active perturbations used to resolve a specific predicate depends on the problem at hand. For the 2D Apollonius diagram perturbing one site always suffices. In its 3D counterpart we may need to perturb two sites, whereas in the case of circular arcs we may need perform a rotation (perturb the axes) and perturb up to one supporting circle per predicate. Minimizing the number of active perturbations is not necessarily desirable, since it might result in a more complicated design for the perturbed predicate (for example, trying to resolve degeneracies for the trapezoidal map of circular arcs with a single active perturbation seems much more complicated).

Besides the number of active perturbations, another important issue is the ordering of the perturbations: for the Apollonius diagram we consider sites by decreasing weight, whereas for the trapezoidal map of circular arcs we first consider a (global) rotation and then the circles by means of decreasing radius. Different perturbation sequences than the ones described in this paper are definitely possible; the analysis, however, can become unnecessarily more complicated.

Our qualitative symbolic perturbation framework, and in particular the schemes described in this paper, can also be applied to a variety of other problems, such as the 2D Voronoi diagram of disjoint convex objects under any L_p metric, as well as the Euclidean Voronoi diagram of certain disjoint convex objects in 3D (the objects can be, for example, non-intersecting lines, line segments or rays). It suffices to replace a site S_i with its Minkowski sum with a ball of radius ε_i, and then consider the limits $\varepsilon_i \to 0^+$, for an appropriately defined ordering of the sites. Another type of geometric problem, involving complex predicates, for which the QSP framework is relevant, is the computation of lines tangent to four given lines in 3D [2, 4].

References

1 P. Alliez, O. Devillers, and J. Snoeyink. Removing degeneracies by perturbing the problem or the world. *Reliable Computing*, 6:61–79, 2000. URL: `http://hal.inria.fr/inria-00338566/`.

2 H. Brönnimann, O. Devillers, V. Dujmović, H. Everett, M. Glisse, X. Goaoc, S. Lazard, H.-S. Na, and S. Whitesides. Lines and free line segments tangent to arbitrary three-dimensional convex polyhedra. *SIAM Journal on Computing*, 37:522–551, 2007. URL: `http://hal.inria.fr/inria-00103916`.

3 C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *5th ACM-SIAM Sympos. Discrete Algorithms*, pages 16–23, 1994. URL: `http://dl.acm.org/citation.cfm?id=314474`.

4 O. Devillers, M. Glisse, and S. Lazard. Predicates for line transversals to lines and line segments in three-dimensional space. In *Proc. 24th Annual Symposium on Computational Geometry*, pages 174–181, 2008. URL: `http://hal.inria.fr/inria-00336256/`.

5 O. Devillers, M. Karavelas, and M. Teillaud. Qualitative symbolic perturbation: two applications of a new geometry-based perturbation framework. Research Report 8153, INRIA, 2015. version 4. URL: `http://hal.inria.fr/hal-00758631/`.

6 O. Devillers and M. Teillaud. Perturbations for Delaunay and weighted Delaunay 3D triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011. `doi:10.1016/j.comgeo.2010.09.010`.

7 H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990. URL: http://dl.acm.org/citation.cfm?id=77639.

8 I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664, 1995. URL: http://epubs.siam.org/sicomp/resource/1/smjcat/v24/i3/p650_s1.

9 I. Emiris and M. Karavelas. The predicates of the Apollonius diagram: algorithmic analysis and implementation. *Computational Geometry: Theory and Applications*, 33(1-2):18–57, January 2006. doi:10.1016/j.comgeo.2004.02.006.

10 G. Irving and F. Green. A deterministic pseudorandom perturbation scheme for arbitrary polynomial predicates. Technical Report 1308.1986v1, arXiv, 2013. URL: http://arxiv.org/abs/1308.1986.

11 K. Mehlhorn, R. Osbild, and M. Sagraloff. A general approach to the analysis of controlled perturbation algorithms. *Comput. Geom. Theory Appl.*, 44:507–528, 2011. doi:10.1016/j.comgeo.2011.06.001.

12 R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998.

13 R. Seidel. Perturbations in geometric computing, 2013. Workshop on Geometric Computing, Heraklion. URL: http://www.acmac.uoc.gr/GC2013/files/Seidel-slides.pdf.

14 C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Syst. Sci.*, 40(1):2–18, 1990. URL: http://www.sciencedirect.com/science/article/pii/002200009090016E.

15 C. K. Yap. Symbolic treatment of geometric degeneracies. *J. Symbolic Comput.*, 10:349–370, 1990. URL: http://www.sciencedirect.com/science/article/pii/S0747717108800697.

16 C. K. Yap and T. Dubé. The exact computation paradigm. In *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, 1995. URL: http://www.cs.nyu.edu/~exact/doc/paradigm.ps.gz.

Finding Global Optimum for Truth Discovery: Entropy Based Geometric Variance*

Hu Ding[1], Jing Gao[2], and Jinhui Xu[3]

1 **Department of Computer Science and Engineering, Michigan State University, East Lansing, USA**
huding@msu.edu
2 **Department of Computer Science and Engineering, State University of New York, Buffalo, USA**
jing@buffalo.edu
3 **Department of Computer Science and Engineering, State University of New York, Buffalo, USA**
jinhui@buffalo.edu

Abstract

Truth Discovery is an important problem arising in data analytics related fields such as data mining, database, and big data. It concerns about finding the most trustworthy information from a dataset acquired from a number of unreliable sources. Due to its importance, the problem has been extensively studied in recent years and a number techniques have already been proposed. However, all of them are of heuristic nature and do not have any quality guarantee. In this paper, we formulate the problem as a high dimensional geometric optimization problem, called *Entropy based Geometric Variance*. Relying on a number of novel geometric techniques (such as Log-Partition and Modified Simplex Lemma), we further discover new insights to this problem. We show, for the first time, that the truth discovery problem can be solved with guaranteed quality of solution. Particularly, we show that it is possible to achieve a $(1+\epsilon)$-approximation within nearly linear time under some reasonable assumptions. We expect that our algorithm will be useful for other data related applications.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems - Geometrical problems and computations

Keywords and phrases geometric optimization, data mining, high dimension, entropy

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.34

1 Introduction

Truth discovery is an emerging topic in data analytics which has received a great deal of attentions in recent years [3, 6, 11, 13, 12, 16, 18, 19]. Despite its extensive studies in the fields of data mining, machine learning, database, and big data, it has yet to be seriously considered by the theory community (to our best knowledge). The problem arises in scenarios where data are acquired from multiple sources which may contain false or inconsistent information, and the truth discovery problem is to find the most trustworthy information from these sources. The problem finds many applications. For example, in online social networks, a user's information can be recorded by multiple websites which may not be always consistent;

* This work was partially supported by NSF through grants CCF-1422324, IIS-1422591, and CNS-1547167. Part of the work was done when the first author was in IIIS, Tsinghua University.


© Hu Ding, Jing Gao, and Jinhui Xu;
licensed under Creative Commons License CC-BY

	George Washington	Abraham Lincoln	Mahatma Gandhi	John Kennedy	Barack Obama	Franklin Roosevelt
Source 1	Virginia	Illinois	Delhi	Texas	Kenya	Georgia
Source 2	Virginia	Kentucky	Porbandar	Massachusetts	Hawaii	New York
Source 3	Maryland	Kentucky	Mumbai	Massachusetts	Kenya	New York
Majority Voting	Virginia	Kentucky	Delhi	Massachusetts	Kenya	New York
Truth Discovery	Virginia	Kentucky	Porbandar	Massachusetts	Hawaii	New York

Figure 1 Three sources are providing the birthplaces of 6 politicians. For *Mahatma Gandhi*, each source has an individual answer (*i.e.,* a tie case), and majority voting can only randomly pick one. More importantly, for *Barack Obama*, voting provides a totally wrong answer. However, truth discovery tries to distinguish reliable and unreliable sources and thus provide the right answer. In this example, the algorithm[6] finds that source 2 has a higher reliability than the other two.

thus it is desirable to find the most trustworthy information for each user. Similar problem also occurs in other areas, such as in healthcare where medical records of a patient may be acquired by multiple hospitals or laboratories.

The main challenge of truth discovery comes from its unsupervised nature, *i.e.,* the level of reliability of each source is unknown in advance. A straightforward way for solving the problem is to take the average if the data are continuous or conduct majority voting if the data are categorical. Such approaches are implicitly based on the assumption that all sources are equally reliable. However, in many applications the level of reliability of each source could be quite different which may make the yielded solution significantly different from the truth, due to the neglect of "the wisdom of minority" [6, 11]. See the example in Fig. 1 from [6]. Thus, estimating the reliability of each source should be taken into account when building the optimization model for truth discovery. In general, the two components, **reliability estimation** and **truth finding**, are tightly coupled and thus are expected to be solved simultaneously, where the truth should be closer to the source with higher reliability, and as a feedback, the source providing closer information should have a higher reliability. Another challenge of truth discovery is how to handle the large number of unreliable sources in the big data era. For example, in the case of monitoring human health, there may be tens of thousands of wearable devices distributed among a population, and each device represents a source which records various kinds of health data of a person for a possibly long period of time (e.g., months or years).

1.1 Problem Formulation and Related Works

We first introduce the problem formulation of truth discovery used in the data mining community, and then convert it to a new geometric optimization problem, called **entropy based geometric variance**.

To model the truth discovery problem, the data from each source can be represented as a (possibly high dimensional) vector, where each dimension corresponds to one attribute/property (*e.g.,* age, income, or temperature). For categorical data, we can reduce them to continuous data as follows [6]. Suppose that one attribute has t categories; then it can be represented as a t-dimensional binary sub-vector, where each dimension indicates the membership of one category. We can finally embed all these sub-vectors (corresponding to

the categorical attributes) into one unified vector in higher dimensional space[1]. Furthermore, we need a variable to represent the reliability of each source.

▶ **Definition 1** (Truth Discovery[13, 6]). Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of vectors in $\mathbb{R}^d$ space with each p_i representing the data from the i-th source (among a set of n sources). The truth discovery problem is to find the truth vector $p^* \in \mathbb{R}^d$ and the reliability (weight) w_i for each i-th source, such that the following objective function is minimized,

$$\sum_{i=1}^{n} w_i ||p^* - p_i||^2, \quad s.t. \quad \sum_{i=1}^{n} e^{-w_i} = 1. \tag{1}$$

In the above optimization problem (1), both p^* and the weights are variables. It is easy to see that when each w_i is fixed, p^* is simply the weighted mean, *i.e.,* $\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i p_i$. This means that the higher the weight of p_i, the closer it is to p^*, which is consistent with the principle of truth discovery.

Weight normalization function. In the above optimization problem, equation $\sum_{i=1}^{n} e^{-w_i} = 1$ is used to normalize the source weights. This way of normalization was initially introduced in [13] (with no justification) and has demonstrated experimentally its superior performance. To understand the rationale behind this, below we give a theoretical justification. Firstly, we notice that some straightforward ways, such as $\sum_{i=1}^{n} w_i^p = 1$ for some $p > 0$, are inappropriate for weight normalization [13], since otherwise, p^* can trivially choose any p_l as its solution and set $w_l = 1$ and $w_i = 0$ for all $i \neq l$ (in this way the objective value will always be equal to the smallest possible value 0). By using equation $\sum_{i=1}^{n} e^{-w_i} = 1$, we can easily avoid this issue. Secondly, this exponential normalization function ensures that the resulting solution minimizes the **entropy**, which implies that the solution contains more information from the input according to Shannon's information theory [17]. To see this, we first borrow the following lemma from [13], which can be shown by using the *Lagrange multipliers* method.

▶ **Lemma 2** ([13]). *If the truth vector p^* is fixed, the following value for each weight w_l minimizes the the objective function (1),*

$$w_l = \log\left(\frac{\sum_{i=1}^{n} ||p^* - p_i||^2}{||p^* - p_l||^2}\right). \tag{2}$$

Let S denote the total squared distance to p^* (*i.e.,* $S = \sum_{i=1}^{n} ||p^* - p_i||^2$), and f_l denote the contribution of each p_l to S (*i.e.,* $f_l = \frac{||p^* - p_l||^2}{S}$). Then the induced entropy is

$$H = -\sum_{l=1}^{n} f_l \log f_l = -\sum_{l=1}^{n} \frac{||p^* - p_l||^2}{S} \log \frac{||p^* - p_l||^2}{S} = \frac{1}{S} \sum_{l=1}^{n} ||p^* - p_l||^2 \log \frac{S}{||p^* - p_l||^2}. \tag{3}$$

Below, we define the *Entropy based Geometric Variance*.

▶ **Definition 3.** Given a set of points P and a point p^* in $\mathbb{R}^d$, the entropy base geometric variance induced by p^* is $H \times S$, where H and S are respectively the entropy and variance defined in the above discussion.

From Lemma 2 and the formula (3), we know that the objective function (1) is equal to the multiplication of S and H, *i.e.,* the entropy based geometric variance.

[1] Note that this representation for categorical data may cause fractional memberships in the final solution, which is often acceptable in practice (*e.g.,* we may claim that one object belongs to class 1, 2, and 3 with probabilities of 70%, 20%, and 10%, respectively).

▶ **Theorem 4.** *The optimization problem (1) is equivalent to finding a point p^* to minimize the entropy based geometric variance.*

Generally speaking, S represents the total variance from the sources to the truth vector, and the entropy H indicates how disorder the system is, where the higher the entropy, the greater disorder the system is. Since we minimize both of them, this implicitly explains the better performance of using the exponential normalization function in Definition 1.

Non-convexity. As shown in [13], when the truth vector or the weights are fixed, the optimization problem (1) is convex. However, when both of them are variables, the problem is non-convex in general. To see this, consider the following simple example. Suppose $n = 2$. Then the objective value is 0 when p^* coincides with either p_1 or p_2, according to Lemma 2 (note $\lim_{x\to 0} x \log(1/x) = 0$). This means that it is possible to have multiple isolated local or global optimal solutions for truth discovery, implying that truth discovery is non-convex.

Existing Approaches. To the best of our knowledge, all existing methods for truth discovery are based on some heuristic ideas, which achieve only a local optimal solution and have no quality guarantee on global optimality. A commonly used strategy is alternating minimization [13, 12, 15], which alternatively fixes either the weights or the truth vector, and optimizes the other. The optimization problem becomes convex when one of the two types of variables is fixed. This means that such approaches are guaranteed to converge to some local optima. Other approaches [19] follow similar ideas. The reader is referred to a recent survey [6] for a comprehensive introduction to these approaches.

1.2 Preliminaries and Our Main Results

Different from existing approaches, our goal is to achieve a quality guaranteed solution for the truth discovery problem. In practice, we can assume that the number of sources n and the size of the data in each source d are both large. As a starting point, the following theorem suggests that it is easy to generate a 2-approximation in quadratic time (due to space limit, we omit some of the proofs in our paper).

▶ **Theorem 5.** *If one tries every point in $\{p_i \mid 1 \le i \le n\}$ as a candidate for the truth vector, at least one yields a 2-approximation for the objective function in (1), and the total running time is $O(n^2 d)$.*

Theorem 5 implies that any further improvement needs to decrease either the approximation ratio or the running time. In this paper, we aim to achieve a $(1+\epsilon)$-approximation for truth discovery and also keep the time complexity as low as possible.

For ease of discussion, we use the following notations throughout the rest of this paper. Let $L_{\min} = \min\{||p_i - p_{i'}|| \mid 1 \le i \ne i' \le n\}$, $L_{\max} = \max\{||p_i - p_{i'}|| \mid 1 \le i \ne i' \le n\}$, and the spread ratio $\Delta = \frac{L_{\max}}{L_{\min}}$. To achieve a $(1+\epsilon)$-approximation for the truth discovery problem for any given small value $1 > \epsilon > 0$, we consider the following two cases.

Case 1. $\min_{1\le i\le n} ||p^* - p_i|| \le \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$, *i.e.,* some p_i locates very close to p^*.

Case 2. $\min_{1\le i\le n} ||p^* - p_i|| > \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$, *i.e.,* no p_i locates very close to p^*.

In following sections, we will present efficient algorithms to solve the two cases separately. For case 1, we show that the nearest point p_i to p^* is actually a $(1+\epsilon)$-approximation in Section 2. For case 2, we first give a simple linear time algorithm with large approximation ratio in Section 3, based on an analysis on the distribution of the weights; then in Section 4,

we reveal several new insights to the weights by using a novel *Log-Partition* technique, and perform a sequence of geometric operations to obtain a $(1+\epsilon)$-approximation. The time complexity depends on Δ. Finally, in Section 5 we show that when Δ is not too large, the time complexity for both cases can be improved to nearly linear ($O(nd \times poly(\log n))$); also through dimension reduction, the complexity can be further improved to linear ($O(nd)$). Note that spread ratio is commonly used as a parameter in many geometric algorithms and appears in the time complexity (such as [8]); it is usually not very large [4], especially in high dimensional space.

We introduce the following two folklore lemmas [2, 10] which are repeatedly used in our analysis. Let $Q = \{q_i \mid 1 \le i \le n\}$ be a set of n points in $\mathbb{R}^d$ with each q_i associated with a weight $w_i \ge 0$, $W = \sum_{i=1}^n w_i$, and $m(Q)$ be the weighted mean of Q *i.e.,* $m(Q) = \sum_{i=1}^n w_i q_i / W$.

▶ **Lemma 6.** *For an arbitrary point q, $\sum_{i=1}^n w_i||q-q_i||^2 = W||q-m(Q)||^2 + \sum_{i=1}^n w_i||m(Q)-q_i||^2$.*

▶ **Lemma 7.** *Let Q_1 be a subset of Q with a total weight of αW for some $0 < \alpha \le 1$. Let $m(Q_1)$ be the weighted mean point of Q_1. Then $||m(Q_1) - m(Q)|| \le \sqrt{\frac{1-\alpha}{\alpha}}\delta$, where $\delta^2 = \frac{1}{W}\sum_{i=1}^n w_i||q_i - m(Q)||^2$.*

2 A $(1+\epsilon)$-Approximation for Case 1

In this section, we consider case 1. Without loss of generality, we assume that $||p^* - p_{i_0}|| \le \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$, *i.e.,* p_{i_0} is the point very close to p^*. Then, we have:

▶ **Lemma 8.** *For any $i \ne i_0$, $||p^* - p_i|| \ge (1 - \frac{\epsilon}{4})||p_{i_0} - p_i||$.*

Proof. Since p^* is the weighted mean $\frac{\sum_{i=1}^n w_i p_i}{\sum_{i=1}^n w_i}$, we know that for any $1 \le l \le n$,

$$||p^* - p_l|| = ||\frac{\sum_{i=1}^n w_i p_i}{\sum_{i=1}^n w_i} - p_l|| \le \sum_{i=1}^n (\frac{w_i}{\sum_{i=1}^n w_i})||p_i - p_l|| \le L_{\max}. \tag{4}$$

Thus, we have $S \le nL_{\max}^2$, and consequently

$$||p^* - p_{i_0}|| \le \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta} \le \frac{\epsilon}{4}\frac{L_{\max}}{\Delta} = \frac{\epsilon}{4}L_{\min}. \tag{5}$$

Furthermore, due to triangle inequality, we have

$$||p^* - p_i|| \ge ||p_{i_0} - p_i|| - ||p^* - p_{i_0}|| \ge ||p_{i_0} - p_i|| - \frac{\epsilon}{4}L_{\min} \ge (1 - \frac{\epsilon}{4})||p_{i_0} - p_i|| \tag{6}$$

for any $i \ne i_0$. ◀

Now we can obtain a $(1+\epsilon)$-approximation for case 1.

▶ **Theorem 9.** *For case 1, if one tries every point in $\{p_i \mid 1 \le i \le n\}$ as a candidate for the truth vector, at least one yields a $(1+\epsilon)$-approximation for the objective function in (1), and the total time complexity is $O(n^2 d)$.*

Proof. We prove this theorem by showing how large the objective value will increase if p^* is moved to p_{i_0}. Firstly, we suppose that the weights are fixed temporarily. Then, by Lemma 8 and the fact that $0 < \epsilon < 1$, we have

$$\frac{||p_{i_0} - p_i||^2}{||p^* - p_i||^2} \leq \frac{1}{(1-\epsilon/4)^2} \leq 1 + \epsilon \tag{7}$$

for any i. This means that the objective value is increased by a factor no more than $1+\epsilon$. Once p^* is moved to p_{i_0}, we can further update the weights according to Lemma 2, and the objective value will not increase. Note that the contribution of p_{i_0} to the objective value will become 0 since $\lim_{x\to 0} x \log \frac{S}{x} = 0$.

Since we need to try every point to find out p_{i_0} (as the candidate for the truth vector) and each point takes $O(nd)$ time to evaluate the objective function, the total time complexity is thus $O(n^2 d)$. ◀

3 A Simple Linear Time Algorithm for Case 2

In this section, we present a simple linear time approximation algorithm for case 2. Although the approximation ratio is relatively large ($O(\log \frac{n\Delta}{\epsilon})$), the idea used in the algorithm sheds some lights on how to find a more refined solution, *e.g.,* $(1+\epsilon)$-approximation in Section 4. Additionally, we believe that this simple linear time algorithm is also of some independent interest.

We first estimate the range for the weights. Lemmas 10 and 11 provides the upper and lower bounds for each w_i, and Lemma 12 shows a lower bound on their summation.

▶ **Lemma 10.** *For case 2, each weight* $w_i \leq 2 \log \frac{n\Delta}{\epsilon}$.

Lemma 10 can be easily obtained from the assumption $\min_{1\leq i\leq n} ||p^* - p_i|| > \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$ and Lemma 2. Note that we assume $n \geq 16$ here; otherwise, we just need to increase the front constant "2" (of $2\log \frac{n\Delta}{\epsilon}$) a little bit.

▶ **Lemma 11.** *For any constant* $2 > c > 1$*, at least one of the following two events happens:*

1. $\min_{1\leq i\leq n} w_i \geq \log c$*;*
2. *all weights except one are at least* $\log \frac{c}{c-1}$.

Proof. Suppose that the first event does not happen, *i.e.,* $\min_{1\leq i\leq n} w_i < \log c$. Then, by Lemma 2 we know that there exists a p_l such that $\frac{||p^*-p_l||^2}{\sum_{i=1}^n ||p^*-p_i||^2} > \frac{1}{c}$. Since $\frac{1}{c} > \frac{1}{2}$, there is at most one such p_l, and each of the other points should have a weight at least $\log \frac{1}{1-1/c} = \log \frac{c}{c-1}$, *i.e.,* the second event happens. Thus the lemma is true. ◀

▶ **Lemma 12.** *The sum of the weights* $\sum_{i=1}^n w_i \geq n \log n$.

Proof. From Lemma 2, we know that $\sum_{i=1}^n w_i = \sum_{i=1}^n \log \frac{S}{||p^*-p_i||^2}$. It is easy to see that the function $f(x) = \log \frac{S}{x}$ is convex, since $f''(x) = \frac{1}{x^2} > 0$. By *Jensen's inequality*, we have

$$\sum_{i=1}^{n} \log \frac{S}{||p^* - p_i||^2} \geq n \log \frac{S}{\sum_{i=1}^n ||p^* - p_i||^2/n} = n \log n. \tag{8}$$

This completes the proof. ◀

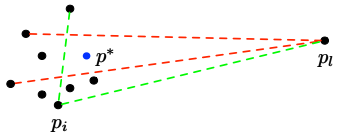


Figure 2 p_l and p_i are connected by dashed lines to their respective farthest and second farthest points; the blue point is p^*.

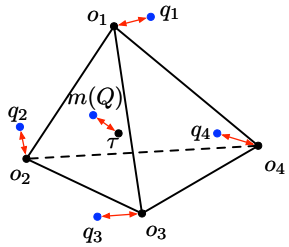


Figure 3 An illustration for Lemma 18 with $k = 4$; each o_j has a bounded distance to q_j, the corresponding exact weighted mean of Q_j, and the distance between τ and $m(Q)$ is also bounded.

Algorithm 1 Linear time algorithm for case 2

Input: $P = \{p_i, \mid 1 \le i \le n\} \subset \mathbb{R}^d$

1. Compute the mean of P , and denote it as p_1^*.
2. Arbitrarily pick one point from P and compute the ratio of the largest and second largest distances from other points to it,
   a. remove the selected point from P if the ratio is smaller than 1.618;
   b. or remove the farthest point to it otherwise.
3. Compute the mean of the remaining points, and denote it as p_2^*.
4. Take the one from $\{p_1^*, p_2^*\}$ with a smaller objective value as the truth vector.

Before introducing the algorithm, we first find a proper value for c in Lemma 11. For this purpose, we consider the second event in Lemma 11. If this event happens, we know that there exists one point, say p_l, with weight less than $\log c$, and all other weights are at least $\log \frac{c}{c-1}$. This implies that

$$||p^* - p_l||^2 > \frac{1}{c}S; \quad \textit{and} \quad ||p^* - p_i||^2 < (1 - \frac{1}{c})S \quad \forall i \neq l. \tag{9}$$

This means that p_l is farther away from p^* than all other points, and the smaller c, the larger difference is. To differentiate p_l from others, we consider **the ratio of the largest over the second largest distances from other points to** p_l (see Fig. 2), which is smaller than

$$(\sqrt{\frac{1}{c}} + \sqrt{1 - \frac{1}{c}})/(\sqrt{\frac{1}{c}} - \sqrt{1 - \frac{1}{c}}) \tag{10}$$

due to triangle inequality and the fact that when $x > y > 0$, the function $f(x, y) = \frac{x+y}{x-y}$ is decreasing on x and increasing on y. Similarly, the ratio for any other p_i for $i \neq l$ is bigger than

$$(\sqrt{\frac{1}{c}} - \sqrt{1 - \frac{1}{c}})/(2\sqrt{1 - \frac{1}{c}}). \tag{11}$$

From the above two inequalities, we know that in order to make (11) larger than (10), we need to have $c < 10 - 4\sqrt{5} \approx 1.056$, and (11) $=$ (10) ≈ 1.618 if $c = 10 - 4\sqrt{5}$. Consequently, we have the following lemma.

▶ **Lemma 13.** *It is possible to find the point p_l with the smallest weight in $O(nd)$ time, if the second event in Lemma 11 happens with $1 < c < 10 - 4\sqrt{5}$.*

Proof. To prove this lemma, we can arbitrarily pick one point from the input and compute the ratio of the largest over the second largest distances from other points to it. From the above analysis, we know that if the ratio is smaller than 1.618, this point is p_l; otherwise the farthest point to it is p_l. Obviously, the total time of the above procedure is linear, *i.e.*, $O(nd)$. ◀

Now we are ready to present our algorithm (see Algorithm 1).

▶ **Theorem 14.** *Algorithm 1 yields a* $(2\log\frac{n\Delta}{\epsilon}/\log c)$*-approximation for case 2, where* $c \approx 1.056$ *and the time complexity is* $O(nd)$*. In short, the approximation ratio is* $O(\log\frac{n}{\epsilon})$ *if* Δ *is a polynomial of* n*, or* $O(\log\frac{\Delta}{\epsilon})$ *otherwise.*

Proof. To prove this theorem, we consider the two events in Lemma 11 separately.

If the first event happens, we have $w_i \geq \log c$ for any i. Since p_1^* is the mean of P, we have $\sum_{i=1}^n ||p_1^* - p_i||^2 \leq \sum_{i=1}^n ||p^* - p_i||^2$. Consequently, if we fix the weights and move p^* to p_1^* (note that we can use Lemma 2 to update the weights and further reduce the objective value), the objective value of (1) will be

$$\begin{aligned}\sum_{i=1}^n w_i||p_1^* - p_i||^2 &\leq 2\log\frac{n\Delta}{\epsilon}\sum_{i=1}^n ||p_1^* - p_i||^2 \\ &\leq 2\log\frac{n\Delta}{\epsilon}\sum_{i=1}^n ||p^* - p_i||^2 \\ &\leq 2\log\frac{n\Delta}{\epsilon}\sum_{i=1}^n \frac{w_i}{\log c}||p^* - p_i||^2 = (2\log\frac{n\Delta}{\epsilon}/\log c)\sum_{i=1}^n w_i||p^* - p_i||^2 \end{aligned} \quad (12)$$

based on Lemma 10 & 11, which implies that p_1^* is a $(2\log\frac{n\Delta}{\epsilon}/\log c)$-approximation.

Now we consider the second event. Let p_l denote the point removed in step 2(a) or 2(b). From the proof of Lemma 13, we know that p_l has the smallest weight. Let $\tilde{p}^*$ be the weighted mean of $P \setminus \{p_l\}$. Suppose that the total weight of $P \setminus \{p_l\}$ is $\alpha\sum_{i=1}^n w_i$, then from Lemma 12 and the fact that $w_l \leq \log c < 1$, we have $\alpha > \frac{n\log n - 1}{n\log n}$. As a consequence, by Lemma 7 we have

$$||\tilde{p}^* - p^*||^2 < \frac{1}{n\log n - 1}\frac{\sum_{i=1}^n w_i||p^* - p_i||^2}{\sum_{i=1}^n w_i}. \quad (13)$$

Then applying Lemma 6 in Section 1.2, we get

$$\sum_{i=1}^n w_i||\tilde{p}^* - p_i||^2 \leq \frac{n\log n}{n\log n - 1}\sum_{i=1}^n w_i||p^* - p_i||^2. \quad (14)$$

(14) indicates that $\tilde{p}^*$ can replace p^* without causing much increase on the objective value. If we continue to move $\tilde{p}^*$ to p_2^*, the objective value becomes

$$\sum_{i=1}^n w_i||p_2^* - p_i||^2 = \sum_{i\neq l} w_i||p_2^* - p_i||^2 + w_l||p_2^* - p_l||^2. \quad (15)$$

For the first term in the right hand side of (15), by a similar calculation to (12), we know that $\sum_{i\neq l} w_i||p_2^* - p_i||^2 < (2\log\frac{n\Delta}{\epsilon}/\log\frac{c}{c-1})\sum_{i\neq l} w_i||\tilde{p}^* - p_i||^2$. For the second term in the right hand side of (15), by an estimation similar to (10), we have $\frac{||p_2^* - p_l||}{||\tilde{p}^* - p_l||} \leq (\sqrt{\frac{1}{c}} + \sqrt{1-\frac{1}{c}})/(\sqrt{\frac{1}{c}} - \sqrt{1-\frac{1}{c}}) \approx 1.618$. (Note that both p_2^* and $\tilde{p}^*$ are a convex combination of

$P \setminus p_l$). Putting (14) and (15) together, we know that p_2^* is a solution with approximation ratio

$$\frac{n \log n}{n \log n - 1} \times \max\{2 \log \frac{n\Delta}{\epsilon} / \log \frac{c}{c-1}, 1.618\} \leq 2 \log \frac{n\Delta}{\epsilon} / \log c. \tag{16}$$

Finally, it is easy to know that the time complexity is $O(nd)$. ◀

4 A $(1+\epsilon)$-Approximation for Case 2

In this section, we present a $(1+\epsilon)$-approximation for case 2. In Theorem 14, we consider only two groups of the points, *i.e.,* the point with the smallest weight and all others. In this section we show that by further partitioning the input points into more groups, it is possible to obtain a much better solution.

▶ **Definition 15** (Log-Partition)**.** In case 2, let p_l and $p_{l'}$ be the points with the smallest and the second smallest weights, respectively. Then the log-partition is to divide the points in $\{p_i \mid 1 \leq i \leq n\}$ into $k = \lceil \log_{1+\beta} \frac{2\log(n\Delta/\epsilon)}{w_{l'}} \rceil + 1$ (where β is a small positive number that will be determined later) groups as follows:

- $\mathcal{G}_1 = \{p_l\}$.
- $\mathcal{G}_j = \{p_i \mid (1+\beta)^{j-2} w_{l'} \leq w_i < (1+\beta)^{j-1} w_{l'}\}$ for $j \geq 2$.

Note that we cannot explicitly obtain the log-partition since we do not know the weights in advance. We can only assume that such a partition exists, which will be useful in the following analysis.

From Lemmas 10 and 11 and the fact that $\log(1+\beta) \approx \beta$ when β is a small positive number, we can easily have the following lemma.

▶ **Lemma 16.** *In the log-partition,* $k = O(\frac{1}{\beta} \log\log \frac{n\Delta}{\epsilon})$.

In each $\mathcal{G}_j$, their weight difference is no more than a factor of $(1+\beta)$; as a consequence, their weighted mean and weighted standard deviation are very close to their mean and standard deviation respectively. In the remaining parts, we denote the mean and weighted mean of each $\mathcal{G}_j$ by $\hat{m}_j$ and m_j, the standard deviation and weighted standard deviation by $\hat{\delta}_j$ and δ_j, [2] respectively.

▶ **Lemma 17.** *For each* $\mathcal{G}_j$ *in the log-partition,* $||\hat{m}_j - m_j|| \leq \beta\sqrt{1+\beta}\delta_j$, *and* $\delta_j \in [\frac{1}{\sqrt{1+\beta}}\hat{\delta}_j, \sqrt{1+\beta}\hat{\delta}_j]$.

Using Lemma 17, we can obtain a $(1+\epsilon)$-approximation algorithm for case 2. Below is the sketch of our idea.

Synopsis. The essential task of truth discovery is to find the weighted mean without knowing the weights in advance. Using log-partition, we can first divide the input points implicitly into k groups, and Lemma 17 enables us to ignore the weights inside each group. Then by applying random sampling techniques, we can estimate the weighed mean of each group, and find the weighted mean of the whole input using *simplex lemma*. We elaborate our ideas in the following subsections.

[2] $\hat{\delta}_j = \sqrt{\frac{1}{|\mathcal{G}_j|}\sum_{p_i \in \mathcal{G}_j} ||p_i - \hat{m}_j||^2}$ and $\delta_j = \sqrt{\frac{1}{\sum_{p_i \in \mathcal{G}_j} w_i}\sum_{p_i \in \mathcal{G}_j} w_i ||p_i - m_j||^2}$.

4.1 Modified Simplex Lemma

In [2], Ding and Xu introduced a simplex lemma for solving a large class of constrained clustering problems in high dimensional space. In this subsection, we show that despite developed for a different purpose, the simplex lemma is still applicable to our truth discovery problem.

▶ **Lemma 18** (Simplex Lemma [2]). *Given an unknown weighted point-set $Q \subset \mathbb{R}^d$, which is implicitly divided into k mutually exclusive groups $\{Q_j \mid 1 \leq j \leq k\}$, and k points $\{o_j \mid 1 \leq j \leq k\}$ satisfying the condition that for each j, the distance between o_j and the weighted mean of the unknown Q_j is no more than a fixed value $L \geq 0$, it is possible to construct a grid of size $((8k/\epsilon)^k)$ inside the simplex determined by $\{o_j \mid 1 \leq j \leq k\}$ such that at least one grid point τ satisfies the following inequality.*

$$||\tau - m(Q)|| \leq \sqrt{\epsilon}\delta(Q) + (1+\epsilon)L, \tag{17}$$

where $m(Q)$ and $\delta(Q)$ are the weighted mean and standard deviation of Q, respectively.

Simplex lemma shows that it is possible to find an approximate weighted mean of an unknown point-set. The only known information is the approximate weighted mean of each unknown subset. L is a slack parameter to control the error bound in (17). See Fig. 3. Also, a nice feature of the simplex lemma is that it needs to consider only a low dimensional subspace determined by the simplex ($k \ll d$), and thus can be applied to problems in high dimensional space.

It is easy to see that the simplex lemma is immediately applicable to the truth discovery problem for finding the weighted mean, if we are able to obtain the weighted mean (or only the mean due to Lemma 17) of each $\mathcal{G}_j$. The difficulty is that since some $\mathcal{G}_j$ could be quite small in its cardinality, it is extremely challenging to estimate the mean by using random sampling techniques. The following modified simplex lemma shows that it is actually possible to ignore such small-size groups.

▶ **Lemma 19** (Modified Simplex Lemma). *Let Q, Q_j, ϵ, δ, and k be defined as in Lemma 18, and $\Gamma = \{j \mid \frac{w(Q_j)}{w(Q)} \geq \frac{\epsilon}{k}\}$, where $w(\cdot)$ is the total weight of a point-set. Then it is possible to construct a grid of size $((8k/\epsilon)^k)$ inside the simplex determined by $\{o_j \mid j \in \Gamma\}$ such that at least one grid point τ satisfies the following*

$$||\tau - m(Q)|| \leq 2\sqrt{\frac{\epsilon}{1-\epsilon}}\delta(Q) + (1+\epsilon)L. \tag{18}$$

Proof. Let $Q_\Gamma = \cup_{j\in\Gamma} Q_j$. Then by Lemma 18, we immediately have the following inequality.

$$||\tau - m(Q_\Gamma)|| \leq \sqrt{\epsilon}\delta(Q_\Gamma) + (1+\epsilon)L, \tag{19}$$

where Q is simply replaced by Q_Γ. Now, we consider the differences between $m(Q)$, $\delta(Q)$ and $m(Q_\Gamma)$, $\delta(Q_\Gamma)$, respectively. Similar to (13) in Theorem 14 for proving the distance between $\tilde{p}^*$ and p^*, based on Lemma 7 we have

$$||m(Q_\Gamma) - m(Q)||^2 \leq \frac{w(Q \setminus Q_\Gamma)}{w(Q_\Gamma)}\delta^2(Q) \leq \frac{\epsilon}{1-\epsilon}\delta^2(Q), \tag{20}$$

where the last inequality comes from the facts that $w(Q \setminus Q_\Gamma) \leq k \times \frac{\epsilon}{k} w(Q)$ and $w(Q_\Gamma) \geq (1-\epsilon)w(Q)$. Furthermore, since $w(Q)\delta^2(Q) \geq w(Q_\Gamma)\delta^2(Q_\Gamma)$, we have

$$\delta^2(Q_\Gamma) \leq \frac{w(Q)}{w(Q_\Gamma)}\delta^2(Q) \leq \frac{1}{1-\epsilon}\delta^2(Q). \tag{21}$$

Plugging (20) and (21) into (19), we have

$$\begin{aligned} ||\tau - m(Q)|| &\leq ||\tau - m(Q_\Gamma)|| + ||m(Q_\Gamma) - m(Q)|| \\ &\leq \sqrt{\epsilon}\delta(Q_\Gamma) + (1+\epsilon)L + \sqrt{\frac{\epsilon}{1-\epsilon}}\delta(Q) \\ &\leq \sqrt{\epsilon}\frac{1}{\sqrt{1-\epsilon}}\delta(Q) + (1+\epsilon)L + \sqrt{\frac{\epsilon}{1-\epsilon}}\delta(Q) \\ &= 2\sqrt{\frac{\epsilon}{1-\epsilon}}\delta(Q) + (1+\epsilon)L. \end{aligned} \tag{22}$$

This completes the proof. ◀

4.2 The Algorithm Using Modified Simplex Lemma

The following two lemmas are commonly used random sampling techniques in Euclidean space. Lemma 20 shows that in order to estimate the mean of a point-set, one just needs to take the mean of a randomly selected sample. Lemma 21 further shows how to sample points in order to ensure that there are enough number of points in the sample from a hidden subset.

▶ **Lemma 20** ([5]). *Let T be a set of n points in $\mathbb{R}^d$ space, T' be a randomly selected subset of size t from T, and $\hat{m}(T)$, $\hat{m}(T')$ be the mean points of T and T' respectively. With probability $1-\eta$, $||\hat{m}(T) - \hat{m}(T')||^2 < \frac{1}{\eta t}\hat{\delta}^2(T)$, where $\hat{\delta}^2(T) = \frac{1}{n}\sum_{s\in T}||s - m(T)||^2$ and $0 < \eta < 1$.*

▶ **Lemma 21** ([2]). *Let Ω be a set of elements, and T be a subset of Ω with $\frac{|T|}{|\Omega|} = \alpha$ for some $\alpha \in (0,1)$. If randomly select $\frac{t\log\frac{t}{\eta}}{\log(1+\alpha)} = O(\frac{t}{\alpha}\log\frac{t}{\eta})$ elements from Ω, then with probability at least $1-\eta$, the sample contains t or more elements from T for $0 < \eta < 1$ and $t \in \mathbb{Z}^+$.*

By Lemma 19, we know that only those groups with large enough weight need to be considered. The following lemma further shows that each of such groups contains a significant fraction of the input points. This means that we can directly apply Lemmas 20 and 21 to estimate their means.

▶ **Lemma 22.** *In the log-partition for case 2, if a group $\mathcal{G}_j$ has a total weight no less than $\frac{\epsilon}{k}\sum_{i=1}^n w_i$, it contains at least $\frac{\epsilon\log n}{2k\log(n\Delta/\epsilon)}n$ points,* i.e., $|\mathcal{G}_j|/|P| \geq \frac{\epsilon\log n}{2k\log(n\Delta/\epsilon)}$.

Proof. Let $|\mathcal{G}_j|$ and $w(\mathcal{G}_j)$ denote the number of points and the total weight in $\mathcal{G}_j$, respectively. From previous discussion (*i.e.,* Lemmas 10 and 12), we know that

$$w(\mathcal{G}_j) \leq 2\log\frac{n\Delta}{\epsilon}|\mathcal{G}_j|; \qquad \sum_{i=1}^n w_i \geq n\log n. \tag{23}$$

With the assumption $w(\mathcal{G}_j) \geq \frac{\epsilon}{k}\sum_{i=1}^n w_i$, we have $|\mathcal{G}_j| \geq (\frac{\epsilon}{k}\sum_{i=1}^n w_i)/(2\log\frac{n\Delta}{\epsilon}) \geq \frac{\epsilon\log n}{2k\log(n\Delta/\epsilon)}n$. ◀

Now we are ready to present our refined algorithm for truth discovery. Firstly, we use Lemmas 21 and 22 to sample an enough number of points from each group with large enough weight. Then, we apply Lemma 20 to obtain their approximate means. Finally, we use the modified simplex lemma (*i.e.,* Lemma 19) to obtain the desired $(1+\epsilon)$-approximation. See Algorithm 2. Below, we analyze the correctness of the algorithm. For convenience, we denote by $\delta(P)$ the weighted standard deviation induced by p^*, *i.e.,* $\sqrt{\frac{1}{\sum_{i=1}^n w_i}\sum_{i=1}^n w_i||p^* - p_i||^2}$.

Algorithm 2 $(1+\epsilon)$-algorithm for case 2

Input: $P = \{p_i, \mid 1 \le i \le n\} \subset \mathbb{R}^d$, and $\alpha = \frac{\epsilon^3 \log n}{2(\log\log(n\Delta/\epsilon))^2 \log(n\Delta/\epsilon)}$.

1. Randomly take a sample N from the input with size $\frac{4k}{\alpha\beta^2} \log \frac{16k^2}{\beta^2}$.
2. Enumerate all the subsets having $4k/\beta^2$ points from N, compute their means, and put them into a set U.
3. For any k'-tuple from U, where k' is enumerated from $\{1, 2, \cdots, k\}$, apply Lemma 19 to build a grid inside the simplex determined by the k'-tuple.
4. Try all the grid points, and output the one with the smallest objective value of (1) in Definition 1.

A key step for analyzing the correctness of the algorithm is to determine the value of β for log-partition. When applying the modified simplex lemma, we have to keep the value of L to be roughly $O(\sqrt{\epsilon}\delta(P))$, such that the obtained grid point τ can result in a $(1+O(1)\epsilon)$-approximation solution by Lemma 6. Note that the value of L depends on two factors, the distance between m_j and $\hat{m}_j$ (Lemma 17), and the error for estimating the position of $\hat{m}_j$ (Lemma 20). For simplicity, we only consider the first factor; the following analysis will show that the first factor actually dominates the value of L. Firstly, when $j \in \Gamma$ (see Lemma 19), we have the upper bound of $||m_j - \hat{m}_j||$,

$$\beta\sqrt{1+\beta}\delta_j < 2\beta\delta_j \le 2\beta\sqrt{\frac{k}{\epsilon}}\delta(P) \tag{24}$$

by Lemma 17. Meanwhile, we know that $k = O(\frac{1}{\beta}\log\log\frac{n\Delta}{\epsilon})$ by Lemma 16. Thus, we need to set

$$\beta = \frac{\epsilon^2}{\log\log\frac{n\Delta}{\epsilon}} \tag{25}$$

to guarantee that $L = O(\sqrt{\epsilon}\delta(P))$. As a consequence, we have

$$k = \frac{1}{\epsilon^2}(\log\log\frac{n\Delta}{\epsilon})^2. \tag{26}$$

(26) together with Lemma 22 implies that $|\mathcal{G}_j|/|P| \ge \frac{\epsilon^3 \log n}{2(\log\log(n\Delta/\epsilon))^2 \log(n\Delta/\epsilon)}$ for each $j \in \Gamma$. By simple calculations and Lemmas 21, we know that with probability $(1-\frac{1}{4k})^k \ge 1 - 1/4 = 3/4$ the sample N contains at least $4k/\beta^2$ points from each of such group $\mathcal{G}_j$. From Lemma 20, we know that with probability $(1-\frac{1}{4k})^k \ge 3/4$, for each of such $\mathcal{G}_j$ the mean of the corresponding $4k/\beta^2$ points has a distance no more than $\beta\hat{\delta}_j \le \beta\sqrt{1+\beta}\delta_j = O(\sqrt{\epsilon}\delta(P))$ to its mean (the inequality comes from Lemma 17). In total, L is bounded by $O(\sqrt{\epsilon}\delta(P))$, and we have a $(1+O(1)\epsilon)$-approximation (by Lemma 6).

As for the running time, we note that $k = \frac{1}{\epsilon^2}(\log\log\frac{n\Delta}{\epsilon})^2$. The total number of grid points is

$$(|N|^{4k/\beta^2})^k (8k/\epsilon)^k = 2^{O(\frac{1}{\epsilon^8}(\log\log n\Delta)^7)}. \tag{27}$$

Since $\frac{1}{\epsilon^8}(\log\log n\Delta)^7 < \sigma \log n\Delta$ for any small positive σ if ϵ is fixed and $n\Delta$ is large enough, the time complexity is $O(2^{\sigma\log n\Delta} nd) = O((n\Delta)^\sigma nd)$.

From the above analysis, we have the following theorem.

▶ **Theorem 23.** *With probability* 9/16*, Algorithm 2 outputs a* $(1+\epsilon)$*-approximation for case 2, and the time complexity is* $O((n\Delta)^\sigma nd)$*, where* σ *is a small positive number. In short, the time complexity is* $O(n^{1+\sigma}d)$ *if* Δ *is a polynomial of* n*, or* $O(\Delta^\sigma nd)$ *otherwise.*

5 Improving the Time Complexity

A common strategy adopted by the $(1+\epsilon)$-approximation algorithms in Section 2 and 4 for the two cases is to first identify a set of candidates for the truth vector, then compute the objective value for each candidate, and finally output the candidate with the smallest objective value. Since computing the objective value for each candidate costs $O(nd)$ time, the total time is thus $O(z \times nd)$, where z is the number of candidates (*i.e.,* $z = n$ for case 1 and $z = (n\Delta)^\sigma$ for case 2). In this section, we show that when the spread ratio Δ is not too large, the **amortized time complexity** for computing all the objective values of the candidates can be reduced to sub-linear, and consequently the overall time complexity is nearly linear.

Recall that Theorem 4 tells us that the objective value is equal to the entropy based geometric variance $S \times H$ induced by p^*. In order to reduce the time complexity for computing the objective value, below we show how to efficiently compute S and H, respectively.

▶ **Lemma 24.** *The value of S for all the z candidates can be computed in a total of $O((n+z)d)$ time,* i.e., *$O(\frac{nd}{z} + d)$ amortized time complexity for each candidate.*

Proof. Let $\hat{m}(P)$ be the (unit weighted) mean of the point-set P, *i.e.,* $\hat{m}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$. Then in $O(nd)$ time, we can compute the value of $\hat{S} = \sum_{i=1}^{n} ||\hat{m}(P) - p_i||^2$. For each candidate p^*, we know that its total variance $S = \sum_{i=1}^{n} ||p^* - p_i||^2 = n||p^* - \hat{m}(P)||^2 + \hat{S}$ (by Lemma 6 in Section 1.2). Clearly, the variance of p^* can be computed in $O(d)$ time by using the value of $\hat{S}$. This implies that the total time for computing the value of S for all z candidates is $O(nd + zd)$. ◀

From Lemma 24, we know that it is possible to compute the total variance S in an amortized sub-linear time. Below we discuss how to efficiently compute the entropy H. The following lemma comes from [7] for entropy estimation.

▶ **Lemma 25** ([7])**.** *Let $F = \{f_i \mid 1 \le i \le n\}$ be a discrete probabilistic distribution with the entropy $H = \sum_{i=1}^{n} -f_i \log f_i$, and two parameters $\epsilon, \delta \in (0,1)$. There exists an algorithm outputting a value $\tilde{H} \in [(1-\epsilon)H, (1+\epsilon)H]$ with probability $1-\delta$, which makes at most $O(\frac{1}{\epsilon^2 H} \log n \log(\frac{1}{\delta}))$ queries on F.*

To estimate H, the algorithm presented in [7] does not read all the values in F. Instead, it takes only a subset of $O(\frac{1}{\epsilon^2 H} \log n \log(\frac{1}{\delta}))$ samples (*i.e.,* queries) from F. From the above lemma, we know that if H is small, the number of needed queries could be quite large, and consequently the time complexity could be high. To avoid this issue, we show in the following lemma that H can actually be lower bounded in our problem if Δ is not too large. Also note that in our problem each query costs only $O(d)$ time, since it can be computed by equation $f_i = \frac{||p^* - p_i||^2}{S}$, where S is the total variance already obtained in Lemma 24.

▶ **Lemma 26.** *If $\Delta = \tilde{O}(\sqrt{n})$ $(= O(\sqrt{n} \times poly(\log n)))$, $H \ge \frac{1}{poly(\log n)}$.*

For simplicity, we let $l_i^2 = ||p^* - p_i||^2$. Then $H = \sum_{i=1}^{n} \frac{l_i^2}{S} \log \frac{S}{l_i^2}$. Since we only need to care about the ratio $\frac{l_i^2}{S}$, without loss of generality we can assume that $\min_{i<j} ||p_i - p_j||^2 = 1$ and $\max_{i<j} ||p_i - p_j||^2 = \Delta^2$. Before proving Lemma 26, we first have the following lemma.

▶ **Lemma 27.** *Except for the smallest value in $\{l_i^2 \mid 1 \le i \le n\}$, all other values are between $\frac{1}{4}$ and Δ^2. Furthermore, $S > \frac{1}{4}\Delta^2$.*

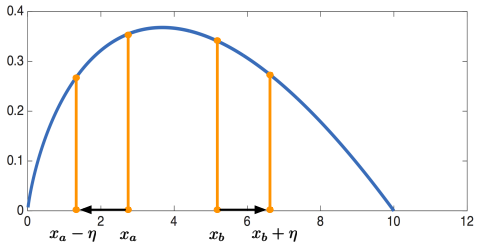


Figure 4 The curve of $g(x)$ with $S = 10$.

Proof of Lemma 26. Let $h = \min\{S, \Delta^2\}$, and $g(x) = \frac{x}{S} \log \frac{S}{x}$ for any $x \in [1/4, h]$, which is concave (as $g''(x) = -\frac{1}{Sx} < 0$). Considering two values $1/4 < x_a \le x_b < h$, we know that

$$g(x_a) + g(x_b) > g(x_a - \eta) + g(x_b + \eta), \tag{28}$$

where $\eta = \min\{x_a - 1/4, h - x_b\}$ (see Fig. 4), and the sum of $x_a - \eta$ and $x_b + \eta$ is always $x_a + x_b$. This suggests that to find a lower bound of H for a fixed S, we can first identify two values $1/4 < l_{i_1}^2 \le l_{i_2}^2 < h$, and then decrease $l_{i_1}^2$ and increase $l_{i_2}^2$ in the same speed until either $l_{i_1}^2 = 1/4$ or $l_{i_2}^2 = h$. After repeating the above operation at most $n-1$ times, we have at most one $l_i^2 \in (1/4, h)$, one smaller than $1/4$ (recall Lemma 27), and all the others are either $1/4$ or h. Suppose that t of them have a value of $1/4$ and $n-2-t$ of them have a value of h in $\{l_i^2 \mid 1 \le i \le n\}$, where $0 \le t \le n-2$. Then we have:

$$H \ge \frac{t}{4S} \log(4S) + \frac{(n-2-t)h}{S} \log \frac{S}{h}; \tag{29}$$

$$(t+1)\frac{1}{4} + (n-2-t)h \le S \le (t+1)\frac{1}{4} + (n-1-t)h. \tag{30}$$

If $S \le 2\Delta^2$, we know that at most two items equal to h (*i.e.*, $n-2-t \le 2$). Consequently, we know that the right hand side of (29) is at least $\frac{t}{4S} \log(4S) \ge \frac{n-4}{4S} \log(4S)$. Also notice that $S \in (\Delta^2/4, 2\Delta^2]$ in this case (where $S > \Delta^2/4$ comes from Lemma 27). Thus, we have

$$H \ge \frac{n-4}{4S} \log(4S) \ge \frac{n-4}{8\Delta^2} \log \Delta^2. \tag{31}$$

To bound the right hand side of the above inequality, we can actually assume that $\Delta^2 \ge 2$. Otherwise, from Lemma 27 and some simple calculations, we know that $H = \sum_{i=1}^n \frac{l_i^2}{S} \log \frac{S}{l_i^2} = \Theta(\log n)$. Thus, (31) becomes $H \ge \frac{n-4}{8\Delta^2} \log 2 \ge \frac{1}{poly(\log n)}$ and the lemma is true.

Now, we consider the case of $S > 2\Delta^2$, which immediately implies that $h = \Delta^2$. Note that $t \in [0, n-2]$. From (29) and (30), we have

$$\begin{aligned} H &\ge \frac{t}{(t+1)+4(n-1-t)\Delta^2} \log(4S) + \frac{4(n-2-t)\Delta^2}{(t+1)+4(n-1-t)\Delta^2} \log \frac{S}{\Delta^2} \\ &\ge \left(\frac{t}{(t+1)+4(n-1-t)\Delta^2} + \frac{4(n-2-t)\Delta^2}{(t+1)+4(n-1-t)\Delta^2}\right) \log 2 \\ &= \frac{t+4(n-2-t)\Delta^2}{(t+1)+4(n-1-t)\Delta^2} \log 2 \\ &= \frac{t+4(n-2-t)\Delta^2}{t+4(n-2-t)\Delta^2+1+4\Delta^2} \log 2 \ge \frac{n-2}{n-2+1+4\Delta^2} \log 2 \ge \frac{1}{poly(\log n)}. \end{aligned} \tag{32}$$

The second inequality follows from $S \ge 2\Delta^2$.

This completes the proof of Lemma 26. ◀

From previous discussion, we know that the time complexity can be improved to nearly linear if $\Delta = \tilde{O}(\sqrt{n})$. We can take the union of the candidates in both case 1 and case 2 (*i.e.,* $\min_{1\le i\le n} ||p^* - p_i|| \le \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$ or $\min_{1\le i\le n} ||p^* - p_i|| > \frac{\epsilon\sqrt{S}}{4\sqrt{n}\Delta}$), and denote it as $\mathcal{Z}$, where $|\mathcal{Z}| = n + (n\Delta)^\sigma = O(n)$ since $\Delta = \tilde{O}(\sqrt{n})$. Note that in case 2, the candidates are obtained by using random sampling which takes sub-linear time (see Section 4.2). Consequently, finding such a set $\mathcal{Z}$ needs only $O(|\mathcal{Z}|d) = O(nd)$ time. By Lemmas 24, 25[3], and 26, we have the following theorem.

▶ **Theorem 28.** *Given an instance P of the truth discovery problem with $\Delta = \tilde{O}(\sqrt{n})$ and two parameters $\epsilon, \delta \in (0,1)$, there exists an algorithm yielding an $(1+\epsilon)$-approximation with success probability $\frac{9}{16}(1-\delta)$. The time complexity is $\tilde{O}(nd)$, where the hiding constant in the big-O notation depends on ϵ and δ.*

In some real applications, the dimensionality d (which is the maximum size of the data from each source) could be much larger than n. For this case, we can first apply the well known *JL-Lemma* [9] to reduce the dimensionality from d to $O(\frac{\log n}{\epsilon^2})$ and then run our algorithm. Note that this can only slightly increase the objective value, since p^* is the weighted mean and consequently

$$\sum_{i=1}^{n} w_i ||p_i - p^*||^2 = \frac{1}{2\sum_{i=1}^{n} w_i} \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j ||p_i - p_j||^2 \tag{33}$$

where JL-Lemma based dimension reduction approximately preserves the pairwise distances. For the running time, we know that a straightforward approach is just multiplying the data matrix $A \in \mathbb{R}^{d\times n}$ to the random projection matrix $R \in \mathbb{R}^{O(\frac{\log n}{\epsilon^2})\times d}$ which costs $O(nd\log n/\epsilon^2)$ time in total. We can also let R be a rescaled random sign matrix [1] and use the technique in [14] to further reduce the time complexity to $O(nd\frac{\log n}{\epsilon^2 \log d})$.

▶ **Corollary 29.** *When $d = \Omega(n^c)$ with some constant $c > 0$, the time complexity in Theorem 28 can be improved to $O(nd\frac{\log n}{\epsilon^2 \log d} + n \times poly(\log n)) = O(nd)$, where the hiding constant depends on c, ϵ and δ.*

References

1. D. Achlioptas, "Database-friendly Random Projections: Johnson-Lindenstrauss with Binary Coins," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 671–687, 2003.
2. H. Ding and J. Xu, "A Unified Framework for Clustering Constrained Data without Locality Property", *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '15)*, pp. 1471–1490, 2015.
3. X. L. Dong, L. Berti-Equille, and D. Srivastava, "Integrating conflicting data: The role of source dependence", *PVLDB*, 2(1):550–561, 2009.
4. H. Edelsbrunner, P. Valtr, and E. Welzl, "Cutting Dense Point Sets in Half", *Proc. of the Tenth Annual Symposium on Computational Geometry (SoCG'94)*, pp. 203–209, 1994.
5. M. Inaba, N. Katoh, and H. Imai, *Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering,* Proc. 10th ACM Symposium on Computational Geometry (SoCG'94), pp. 332–339, 1994.
6. Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A Survey on Truth Discovery", *CoRR* abs/1505.02463, 2015.

[3] When applying Lemma 25, we should replace the parameter δ by $O(\frac{\delta}{n})$ since we have to guarantee the success for all the $O(n)$ candidates. The number of queries increases only by a factor of $\log n$.

7 S. Guha, A. McGregor, and S. Venkatasubramanian, “Sublinear Estimation of Entropy and Information Distances", *ACM Transactions on Algorithms* 5(4) (2009).

8 P. Indyk, R. Motwani, and S. Venkatasubramanian, “Geometric Matching Under Noise: Combinatorial Bounds and Algorithms", *Proc. of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA'99)*, pp. 457–465, 1999.

9 W. Johnson and J. Lindenstrauss, “Extensions of Lipschitz mappings into a Hilbert space," *Conference in Modern Analysis and Probability (New Haven, Conn., 1982)*, Contemporary Mathematics 26, Providence, RI: American Mathematical Society, 189–06.

10 A. Kumar, Y. Sabharwal, and S. Sen, “Linear-time Approximation Schemes for Clustering Problems in Any Dimensions", *J. ACM*, 57(2), 2010.

11 H. Li, B. Zhao, and A. Fuxman, “The Wisdom of Minority: Discovering And Targeting The Right Group of Workers for Crowdsourcing", *Proc. of the International Conference on World Wide Web (WWW'14)*, pp. 165–176, 2014.

12 Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, “A Confidence-Aware Approach for Truth Discovery on Long-Tail Data", *PVLDB* 8(4):425–436, 2014.

13 Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, “Resolving Conflicts in Heterogeneous Data by Truth Discovery and Source Reliability Estimation", *Proc. the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*, pp. 1187–1198, 2014.

14 E. Liberty and S. W. Zucker, ’“The Mailman Algorithm: A Note on Matrix Vector Multiplication," *Inf. Process. Lett.*, vol. 109, no. 3, pp. 179–182, 2009.

15 L. Su, Q. Li, S. Hu, S. Wang, J. Gao, H. Liu, T. F. Abdelzaher, J. Han, X. Liu, Y. Gao, and L. M. Kaplan, “Generalized Decision Aggregation in Distributed Sensing Systems", *Proc. 35rd IEEE Real-Time Systems Symposium (RTSS'14)*, pp. 1-10, 2014.

16 J. Pasternack and D. Roth, “Knowing what to believe (when you already know something)", *Proc. of the International Conference on Computational Linguistics (COLING'10)*, pp. 877–885, 2010.

17 C. E. Shannon, “A Mathematical Theory of Communication", *Bell System Technical Journal*, 27, pp. 379–423 & 623-656, July & October, 1948.

18 J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan, “Whose Vote Should Count More: Optimal Integration of Labelers of Unknown Expertise", *Advances in Neural Information Processing Systems (NIPS'09)*, pp. 2035–2043, 2009.

19 X. Yin, J. Han, and P. S. Yu, “Truth discovery with multiple conflicting information providers on the web", *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pp. 1048–1052, 2007.

On Expansion and Topological Overlap*

Dominic Dotterrer[1], Tali Kaufman[2], and Uli Wagner[3]

1 **Department of Mathematics, University of Chicago, Chicago, USA**
d.dotterrer@math.uchicago.edu
2 **Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel**
kaufmant@macs.biu.ac.il
3 **IST Austria, Klosterneuburg, Austria**
uli@ist.ac.at

Abstract

We give a detailed and easily accessible proof of Gromov's *Topological Overlap Theorem*. Let X be a finite simplicial complex or, more generally, a finite polyhedral cell complex of dimension d. Informally, the theorem states that if X has sufficiently strong *higher-dimensional expansion properties* (which generalize edge expansion of graphs and are defined in terms of cellular cochains of X) then X has the following *topological overlap property*: for every continuous map $X \to \mathbb{R}^d$ there exists a point $p \in \mathbb{R}^d$ whose preimage intersects a positive fraction $\mu > 0$ of the d-cells of X. More generally, the conclusion holds if $\mathbb{R}^d$ is replaced by any d-dimensional piecewise-linear (PL) manifold M, with a constant μ that depends only on d and on the expansion properties of X, but not on M.

1998 ACM Subject Classification G. Mathematics of Computing.

Keywords and phrases Combinatorial Topology, Selection Lemmas, Higher-Dimensional Expanders

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.35

1 Introduction

Let X be a finite polyhedral cell complex of dimension $\dim X = d$. Gromov [6] recently showed that if X has sufficiently strong *higher-dimensional expansion properties* (which generalize edge expansion of graphs, see below for the precise definition) then X has the following *topological overlap property*: For every every continuous map $f\colon X \to \mathbb{R}^d$, there exists a point $p \in \mathbb{R}^d$ that is contained in the images of some positive fraction of the d-cells of X, i.e.,

$$|\{\sigma \in \Sigma_d(X)\colon p \in f(\sigma)\}| \geq \mu \cdot |\Sigma_d(X)|, \tag{1}$$

where $\mu > 0$ and $\Sigma_k(X)$ denotes the set of k-dimensional cells of X, $0 \leq k \leq d$. More generally, the same conclusion holds if the target space $\mathbb{R}^d$ is replaced by a d-dimensional manifold M, and the *overlap constant* $\mu > 0$ depends only on the dimension d and on the constants quantifying the expansion properties of X, but not on M. For technical reasons, we will assume that the manifold M admits a *piecewise-linear* (*PL*) triangulation, so that we can apply standard tools to perturburb a given map to *general position*. We refer to the book by Rourke and Sanderson [13] or to the lecture notes by Zeeman [14] for background and standard facts about piecewise-linear topology.

* Research supported by the Swiss National Science Foundation (Project SNSF-PP00P2-138948).

© Dominic Dotterrer, Tali Kaufman, and Uli Wagner;
licensed under Creative Commons License CC-BY

In the special case that X is the d-dimensional skeleton of the n-dimensional simplex Δ^n, determining the optimal overlap constant for maps $\Delta^n \to \mathbb{R}^d$ is a classical problem in discrete geometry, also known as the *point selection problem* [2, 1] and originally only considered for affine maps. Apart from the generalization from affine to arbitrary continuous maps, Gromov's proof also led to improved estimates for the point selection problem, and a number of papers have appeared with expositions and simplified proofs of Gromov's result in this special case $X = \Delta^n$, see [7, 11] and [4, Sec. 7.8].

The goal of the present paper is to provide a detailed and easily accessible proof of Gromov's result for general complexes X, see Theorem 8 below, which is a crucial ingredient for obtaining examples of simplicial complexes X of *bounded degree* (i.e., such that every vertex is incident to a bounded number of simplices) that have the topological overlap property [5, 3]. The basic idea of the proof is the same as Gromov's, but we present a simplified and streamlined version of the proof that uses only elementary topological notions (general position for piecewise-linear maps, algebraic intersection numbers, cellular chains and cochains, and chain homotopies) and avoids much of the machinery used in Gromov's original paper (in particular, the simplicial set of cocycles).

For stating the result formally, we need to discuss higher-dimensional expansion properties of cell complexes. The relevant notion of expansion originated in the work of Linial and Meshulam [8] and of Gromov [6] and generalizes edge expansion of graphs (which corresponds to 1-dimensional expansion). To define k-dimensional expansion, we need two ingredients: first, information about incidences between cells of dimensions k and $k-1$ and, second, a notion of *discrete volumes* in X. To define these, it is convenient to use the language of *cellular cochains* of X.

Cellular Cochains

Let $\Sigma_k(X)$ denote the set of k-dimensional cells of X, and let $C^k(X) := C^k(X;\mathbb{F}_2) := \mathbb{F}_2^{\Sigma_k(X)}$ be the space of k-dimensional cellular cochains with coefficients in the field $\mathbb{F}_2$; in other words $C^k(X)$ is the space of functions $a\colon \Sigma_k(X) \to \mathbb{F}_2 = \{0,1\}$. For a pair $(\sigma,\tau) \in \Sigma_k(X) \times \Sigma_{k-1}(X)$, let $[\sigma\colon \tau]$ be 1 or 0 depending on whether τ is incident to σ (i.e., whether τ is contained in the boundary $\partial\sigma$) or not. This incidence information is recorded in the *coboundary operator*, which is a linear map $\delta\colon C^{k-1}(X) \to C^k(X)$ given by $\delta a(\sigma) := \sum_{\tau\in\Sigma_{k-1}(X)} [\sigma\colon \tau] a(\tau)$.

The elements of the subspaces $Z^k(X) := \ker(\delta\colon C^k(X) \to C^{k+1}(X))$ and $B^k(X) := \operatorname{im}(\delta\colon C^{k-1}(X) \to C^k(X))$ are called k-dimensional *cocycles* and *coboundaries*, respectively. The composition of consecutive coboundary operators is zero, i.e., $B^k(X) \subseteq Z^k(X)$, and $H^k(X) = Z^k(X)/B^k(X)$ is the k-dimensional *homology group* (with $\mathbb{F}_2$-coefficients) of X. This information is customarily recorded in the *cellular cochain complex*[1] of X

$$0 \longrightarrow \mathbb{F}_2 = C^{-1}(X) \xrightarrow{\delta} C^0(X) \xrightarrow{\delta} C^1(X) \xrightarrow{\delta} \cdots \xrightarrow{\delta} C^{d-1}(X) \xrightarrow{\delta} C^d(X) \longrightarrow 0 \quad (2)$$

Norm, cofilling, expansion and systoles

For $\alpha \in C^k(X)$, let $|\alpha|$ denote the *Hamming norm* of α, i.e., the cardinality of the *support* $\operatorname{supp}(\alpha) := \{\sigma \in \Sigma_k(X)\colon \alpha(\sigma) \neq 0\}$, which we think of as a measure of "discrete k-dimensional

[1] More precisely, we work with the *augmented* cellular cochain complex of X, unless stated otherwise, i.e., we consider X to have a unique (-1)-dimensional cell, the empty cell $\emptyset$, which is incident to every vertex of X.

volume." In fact, it will be convenient to allow more general norms on cochains; the following definition summarizes the properties that we will need.

▶ **Definition 1** (Norm on cochains)**.** A *norm* on the group $C^*(X) = \bigoplus_{k=0}^{d} C^k(X)$ of cellular cochains of X with $\mathbb{F}_2$-coefficients is a function $\|\cdot\| \colon C^*(X;\mathbb{F}_2) \to \mathbb{R}_{\geq 0}$ that satisfies the following properties for all cochains $\alpha, \beta \in C^k(X)$, $0 \leq k \leq d$:

1. $\|0\| = 0$.
2. *Triangle inequality*: $\|\alpha + \beta\| \leq \|\alpha\| + \|\beta\|$.

Furthermore, we will assume throughout that the norm satisfies the following:

3. *Monotonicty*: $\|\alpha\| \leq \|\beta\|$ whenever $\mathrm{supp}(\alpha) \subseteq \mathrm{supp}(\beta)$.

From now on, we work with a fixed norm on the cochains of X. We assume that the norm is normalized such that $\|\mathbb{1}_X^k\| = 1$ for $0 \leq k \leq d$, where $\mathbb{1}_X^k \in C^k(X)$ assigns 1 to every k-cell of X. In particular, when working with the Hamming norm, we will consider its normalized version

$$\|\alpha\|_H := \frac{|\alpha|}{|\Sigma_k(X)|}.$$

Given $\beta \in B^k(X)$, we say that $\alpha \in C^{k-1}(X)$ *cofills* b if $\beta = \delta\alpha$. Once we have a notion of discrete volumes, we can consider the following *(co)isoperimetric question*: Can we bound the minimum norm of a cofilling for a coboundary β in terms of the norm of β?

▶ **Definition 2** (Cofilling/Coisoperimetric Inequality)**.** Let $L > 0$. We say that X satisfies a *L-cofilling inequality* (or *coisoperimetric inequality*) in dimension k if, for every $\beta \in B^k(X)$, there exists some $\alpha \in C^{k-1}(X)$ such that $\delta\alpha = \beta$ and $\|\alpha\| \leq L\|\beta\|$.

Any two cofilings of a given coboundary differ by a cocycle. Thus, X satisfies an L-cofilling inequality in dimension k if and only if

$$\|\delta\alpha\| \geq \frac{1}{L} \cdot \min\{\|\alpha + \zeta\| : \zeta \in Z^{k-1}(X)\} \qquad \text{for all} \quad \alpha \in C^{k-1}(X). \tag{3}$$

We can strengthen (3) by replacing cocycles with coboundaries and obtain a condition that also allows us to draw conclusions about the cohomology of X. For $\alpha \in C^{k-1}(X)$, let

$$\|[\alpha]\| := \min\{\|\alpha + \beta\| : \beta \in B^{k-1}(X)\} \tag{4}$$

denote the distance (with respect to the norm $\|\cdot\|$) of α to the space $B^{k-1}(X)$ of coboundaries.

▶ **Definition 3** (Coboundary Expansion)**.** Let $\eta > 0$. We say that X is *η-expanding* in dimension k, if for every $(k-1)$-cochain $\alpha \in C^{k-1}(X)$,

$$\|\delta\alpha\| \geq \eta \cdot \|[\alpha]\|. \tag{5}$$

▶ **Lemma 4.** *Let $\eta > 0$. A complex X is η-expanding in dimension k if and only if $H^{k-1}(X) = 0$ and X satisfies a $1/\eta$-coisoperimetric inequality in dimension k.*

Proof. Suppose that X is η-expanding in dimension k. Clearly, (5) implies (3), i.e., X satisfies a $1/\eta$-cofilling inequality. Moreover, if $\alpha \in C^{k-1}(X) \setminus B^{k-1}(X)$ then $\|[\alpha]\| > 0$, hence $\|\delta\alpha\| > 0$, hence $\alpha \notin Z^{k-1}(X)$. Thus, $Z^{k-1}(X) = B^{k-1}(X)$, i.e., $H^{k-1}(X) = 0$.

Conversely, assume that $H^{k-1}(X) = 0$. Then $Z^{k-1}(X) = B^{k-1}(X)$, so (5) and (3) are equivalent. ◀

In some cases, however, vanishing of $H^{k-1}(X)$ turns out to be too stringent a requirement, and we can replace it by the condition that every nontrivial cocycle has large norm:

▶ **Definition 5** (Large Cosystoles)**.** Let $\vartheta > 0$. We say that X has *ϑ-large cosystoles* in dimension j if $\|\alpha\| \geq \vartheta$ for every $\alpha \in Z^j(X) \setminus B^j(X)$.

▶ **Example 6.** Consider the case $k = 1$, with the normalized Hamming norm. In this case, η-expansion in dimension 1 corresponds to η-edge expansion of a graph (the 1-skeleton of the complex). An L-cofilling inequality in dimension 1 means that every connected component of the graph is $1/L$-edge expanding. Having ϑ-large cosystoles in dimension 0 means that every connected component contains at least a ϑ-fraction of the vertices.

Local Sparsity of X

For the formal statement of the overlap theorem, we need one more technical condition on X. For a cell τ of X, let ι_τ^k be the k-dimensional cochain that assigns 1 to k-cells of X that have nonempty intersection with τ and 0 otherwise.

▶ **Definition 7** (Local Sparsity)**.** Let $\varepsilon > 0$. We say that X is *locally ε-sparse* (with respect to a given norm $\|\cdot\|$) if $\|\iota_\tau^k\| \leq \varepsilon$ for every nonempty cell τ of X and every k, $0 \leq k \leq d$.

For example, in the case of the normalized Hamming norm $\|\cdot\|_H$, local sparsity means that

$$|\{\sigma \in \Sigma_k(X) \colon \tau \cap \sigma \neq \emptyset\}| \leq \varepsilon |\Sigma_k(X)|,$$

for every nonempty cell τ of X.

Formal Statement of the Theorem

We are now ready to state Gromov's theorem.

▶ **Theorem 8** (Gromov's Topological Overlap Theorem [6])**.** *For every $d \geq 1$ and $L, \vartheta > 0$ there exists $\varepsilon_0 = \varepsilon_0(d, L, \vartheta) > 0$ such that the following holds:*

Let X be a finite cell complex of dimension d, and let $\|\cdot\|$ be a norm on the cochains of X. Suppose that

1. *X satisfies a L-cofilling inequality in dimensions $1, \ldots, d$;*
2. *X has ϑ-large cosystoles in dimensions $0, \ldots, d-1$; and*
3. *X is locally ε-sparse for some $\varepsilon \leq \varepsilon_0$.*

Then for every continuous map $f \colon X \to M$ into a compact connected d-dimensional piecewise-linear (PL) manifold M, there exists a point $p \in M$ such that[2]

$$\|\{\sigma \in \Sigma_d(X) \mid p \in f(\sigma)\}\| \geq \mu, \tag{6}$$

where $\mu = \mu(d, \varepsilon, L, \vartheta) > 0$.

▶ Remark. The assumption that the manifold M is compact is not essential; moreover, we may assume without loss of generality that M has no boundary. Indeed, since X is compact, the image $f(X)$ is compact and hence contained in a compact submanifold N of M with boundary ∂N; we can turn N into a compact manifold without boundary by doubling, i.e., by glueing two copies of N along their boundary.

[2] Here, we use that a subset of $\Sigma_k(X)$ can be identified with a k-dimensional cellular cochain, its indicator function.

If a complex X satisfies the conclusion of the theorem, we also say that X is *topologically μ-overlapping* for maps into d-dimensional PL manifolds. If the conclusion holds true just for *affine maps* and $M = \mathbb{R}^d$, we say that X is *geometrically μ-overlapping*.

2 Piecewise-Linear Topology – Preliminaries

2.1 Assumptions on M

We assume that M is a compact connected piecewise-linear (PL) d-dimensional manifold, without boundary. That is, we assume that M admits a triangulation[3] T with the property that the link of every nonempty simplex τ of T is a PL sphere of dimension $d - 1 - \dim(\tau)$; throughout this paper, we only consider triangulations of M that have this property.

2.2 Approximation by PL maps

We can fix a metric on M, e.g., by fixing a triangulation T of M and metrizing M by considering each simplex of T as a regular simplex with edge length 1. By subdividing a given triangulation T sufficiently often, we can pass to a new triangulation T' in which each simplex has diameter at most $\rho > 0$, for a given ρ (see, e.g., [10, Sec. 1.7]).

By the standard *simplicial approximation theorem* [12], given the triangulation T' of M and a continuous map $f\colon X \to M$, there is a simplicial approximation of f, i.e., there is a subdivision X' of X and a simplicial map $g\colon X' \to T'$ such that, for each point $x \in X$, the image $g(x)$ belongs to the (uniquely defined) simplex of T' whose relative interior contains $f(x)$. (In fact, g is even homotopic to f, but we will not need that.) This map g is a PL map $X \to M$ and the distance between $g(x)$ and $f(x)$ is at most the maximum diameter of any simplex in T', hence at most ρ, for every $x \in X$.

Thus, by the preceding discussion and the following lemma, it suffices to prove Thm. 8 for PL maps:

▶ **Lemma 9.** *Let $f\colon X \to M$ be a continuous map, and let $g_n\colon X \to M$ be a sequence of continuous maps that converges to f pointwise, i.e., $g_n(x) \to f(x)$ as $n \to \infty$, for every $x \in X$. Suppose that for every g_n there exists a point $p_n \in M$ such that $\|\{\sigma \in \Sigma_d(X) \mid p_n \in g_n(\sigma)\}\| \geq \mu$. Then there exists a point $p \in M$ such that (6) holds.*

Proof. By compactness, there is a subsequence of the points p_n that converges to a point p. We claim that p is the desired point. Since there are only finitely many cells in X, there is some $\rho > 0$ such that for every d-cell σ of X with $p \notin f(\sigma)$, the distance between p and $f(\sigma)$ is at lest ρ. Choose n sufficiently large so that the distance between p_n and p is less than $\rho/2$, and the distance between $f(x)$ and $g_n(x)$ is at most $\rho/2$, for every $x \in X$. If $p_n \in g_n(\sigma)$, then the distance between p and $f(\sigma)$ is less than ρ, so by the choice of ρ, we have $p \in f(\sigma)$. Therefore, $\{\sigma \in \Sigma_d(X) \mid p \in f(\sigma)\} \subseteq \{\sigma \in \Sigma_d(X) \mid p_n \in g_n(\sigma)\}$, and the desired conclusion follows by the monotonicity property of the norm. ◀

2.3 General Position

We refer to [14, Ch. VI] for a comprehensive treatment of general position for PL maps. The following definition summarizes the properties that we will need.

[3] The triangulation is necessarily finite, since M is compact.

▶ **Definition 10.** Let X be a finite polyhedral cell complex and let $f\colon X \to M$ be a PL map.

1. We say that f is in *strongly general position* (with respect to the given decomposition of X into polyhedral cells) if, for every $r \geq 1$ and pairwise disjoint cells $\sigma_1, \ldots, \sigma_r$ of X,
$$\dim\left(\bigcap_{i=1}^{r} f(\sigma_i)\right) \leq \max\left\{-1, \left(\sum_{i=1}^{r} \dim \sigma_i\right) - d(r-1)\right\}. \tag{7}$$
In particular, if the number of the right-hand side is -1, then the intersection is empty.
2. Given a triangulation T of M, we that that f is in *general position with respect to* T if, for every simplex σ of X and every simplex τ of T, $\dim(f(\sigma)\cap\tau) \leq \max\{-1, \dim\sigma+\dim\tau-d\}$; moreover, if $\dim\sigma + \dim\tau = d$ then we require that $f(\sigma)$ and τ intersect transversely (either the intersection is empty, or they intersect locally like complementary linear subspaces).

The main fact that we will need is that any map $f\colon X \to M$ can be approximated arbitrarily closely by a PL map that is in general position:

▶ **Lemma 11** ([14, Ch. VI])**.** *Let $f\colon X \to M$ be a PL map and let T be a triangulation of M. Then, up to a small perturbation, we may assume that f is general position with respect to T and in strongly general position.*

Moreover, we will need the following notion of *sufficiently fine triangulations*:

▶ **Definition 12.** Let T be a triangulation of M and let $f\colon X \to M$ be a PL map in general position with respect to T. We say that T is *sufficiently fine* with respect to f if, for every $k > 0$ and every k-simplex τ of T,
$$\|\{\sigma \in \Sigma_{d-k}(X)\colon f(\sigma) \cap \tau \neq \emptyset\}\| \leq \frac{d}{k} \max_{\sigma' \in \Sigma_{d-k}(X)} \|\iota_{\sigma'}^{d-k}\|.$$

▶ **Lemma 13.** *Suppose that $f\colon X \to M$ be a PL map in strongly general position and in general position with respect to a triangulation T of M. Then (by refining T, if necessary), we may assume furthermore that T is sufficiently fine with respect to f.*

Proof. If f is in general position with respect to T, then by choosing points at which we subdivide T in a sufficiently generic way, we can assume that f is also in general position with respect to the subdivision T'. Thus, we may assume that T already has the property that every simplex of T has diameter smaller than some specified parameter $\rho > 0$.

Now suppose that $\sigma_1, \ldots, \sigma_r$ are pairwise distinct simplices of X with $f(\sigma_1)\cap\ldots\cap f(\sigma_r) = \emptyset$. By compactness, there exists $\rho = \rho(\sigma_1, \ldots, \sigma_r) > 0$ such that no matter how we select $x_i \in f(\sigma_i)$, some pair x_i, x_j has distance at least ρ. Since X is finite, there is some $\rho > 0$ that works for all finite collections of simplices whose images do not have a common point of intersection. Suppose now that we have chosen T such that all simplices in T have diameter at most $\rho/2$.

Given $\tau \in T$ of dimension $k > 0$ consider $S(\tau) := \{\sigma \in \Sigma_{d-k}(X)\colon f(\sigma) \cap \tau \neq \emptyset\}$. We claim that $\bigcap_{\sigma\in S(\tau)} f(\sigma) \neq \emptyset$. Otherwise, for every choice of points $x_\sigma \in f(\sigma)$, $\sigma \in S(\tau)$, there would be some pair σ, σ' such that x_σ and $x_{\sigma'}$ have distance at least ρ. However, by the definition of $S(\tau)$, we can choose each x_σ to lie in the intersection $f(\sigma) \cap \tau$, from which it follows that for every pair $\sigma, \sigma' \in S(\tau)$, the distance between x_σ and $x_{\sigma'}$ is at most the diameter of τ, i.e., at most $\rho/2$.

Let $\{\sigma_1, \ldots, \sigma_r\} \subseteq S(\tau)$ be an inclusion-maximal subset with $\sigma_i \cap \sigma_j = \emptyset$ (i.e., the σ_i are pairwise vertex-disjoint; we can pick this subset greedily). Since f is in strongly general position and $\bigcap_{\sigma\in S(\tau)} f(\sigma) \neq \emptyset$, it follows that $\sum_{i=1}^{r}(d-k) - d(r-1) \geq 0$; this implies $r \leq d/k$. Now, every other simplex $\sigma \in S(\tau)$ intersects one of the σ_i. Thus, by monotonicity of the norm and by the triangle inequality, $\|S(\tau)\| \leq \frac{d}{k} \max_{1\leq i\leq r} \|\iota_{\sigma_i}^{d-k}\|$. ◀

2.4 Intersection Numbers

▶ **Definition 14** (Intersection numbers)**.** If T is a PL triangulation of M and if $f\colon X \to M$ is a PL map in general position with respect to T, then for every pair of chains $a \in C_{d-k}(X;\mathbb{F}_2)$ and $b \in C_k(T;\mathbb{F}_2)$, we can define their (algebraic) *intersection number*

$$f(a)\cdot b \in \mathbb{F}_2$$

as follows: If σ is a $(d-k)$-dimensional cell of X and if τ is a k-dimensional simplex of T, then by general position, the intersection $f(\sigma)\cap\tau$ consists of a finite number of points, and the intersection number $f(\sigma)\cdot\tau$ is defined as the number of intersections[4] modulo 2. This definition is extended by linearity (over $\mathbb{F}_2$) to arbitrary chains.

This yields, for $0 \le k \le d$, an *intersection number homomorphism*

$$f^{\pitchfork}\colon C_k(T) \to C^{d-k}(X), \tag{8}$$

defined by $f^{\pitchfork}(b)(a) = f(a)\cdot b$ for each $a \in C_{d-k}(X)$.

It is well-known that the intersection number homomorphism is a *chain-cochain map*, i.e., it commutes with the boundary and coboundary operators in the following sense (see, e.g., [9, Sec. 2.2] for a detailed review of this and other properties of intersection numbers).

▶ **Lemma 15.**

$$f^{\pitchfork}(\partial a) = \delta f^{\pitchfork}(a).$$

For the proof of the main theorem, we need the following definition:

▶ **Definition 16** (Chain-cochain homtopy)**.** Consider two chain-cochain maps $\varphi,\psi\colon C_k(M)\to C^{d-k}(X)$ from the (non-augmented) chain complex of M to the cochain complex of X. A *chain-cochain homtopy* between φ and ψ is a family of linear maps $h\colon C_k(M)\to C^{d-k-1}(X)$ such that $\varphi-\psi = h\partial+\delta h$. To keep track of the various maps, it is convenient to keep in mind the following diagram:

$$\begin{array}{ccccccccccccc}
0 & \longrightarrow & C_d(M) & \xrightarrow{\partial} & C_{d-1}(M) & \xrightarrow{\partial} & \cdots & \xrightarrow{\partial} & C_1(M) & \xrightarrow{\partial} & C_0(M) & \longrightarrow & 0 \\
 & & \swarrow^{h} \;\varphi\Downarrow\psi & & \swarrow^{h} \;\varphi\Downarrow\psi & & \swarrow^{h} & & \swarrow^{h} \;\varphi\Downarrow\psi & & \swarrow^{h} \;\varphi\Downarrow\psi & & \\
0 & \longrightarrow & C^0(X) & \xrightarrow[\delta]{} & C^1(X) & \xrightarrow[\delta]{} & \cdots & \xrightarrow[\delta]{} & C^{d-1}(X) & \xrightarrow[\delta]{} & C^d(X) & \longrightarrow & 0
\end{array} \tag{9}$$

3 Proof of the Overlap Theorem

Proof of Theorem 8. Let μ and ε_0 be parameters that we will determine in the course of the proof. We assume that X satisfies the assumptions of the theorem, in particular that it is locally ε-sparse for some $\varepsilon \le \varepsilon_0$.

Let $f\colon X \to M$ be a map. By the discussion in Sec. 2.2 and by Lemmas 11 and 13, we may assume that f is PL and in general position with respect to a sufficiently fine PL triangulation T of M.

[4] There is a small caveat: In the case $k=0$, an intersection point in $f(\sigma)\cdot\tau$ may have several preimages in σ and should be counted with the corresponding multiplicity; equivalently, the intersection number is defined as the number of points in $\sigma\cap f^{-1}(\tau)$ modulo 2.

We wish to show that there is a vertex v of T such that the intersection number cochain $f^\pitchfork(v) \in C^d(X)$ satisfies $\|f^\pitchfork(v)\| \geq \mu$. We assume that this is not the case and we proceed to derive a contradiction.

Let v_0 be a fixed vertex of T; by assumption, $\|f^\pitchfork(v_0)\| < \mu$. (Note that if f is not surjective then we can choose the triangulation T and v_0 so that $\|f^\pitchfork(v_0)\| = 0$.)

We define a chain-cochain map[5]

$$G\colon C_*(T) \to C^{d-*}(X)$$

by setting $G(v) := f^\pitchfork(v_0)$ for every vertex v of T and $G(c) = 0$ for every $c \in C_k(T; \mathbb{F}_2)$, $k > 0$.

We will construct a *chain-cochain homotopy* $H\colon C_*(T) \to C^{d-1-*}(X)$ between $f^\pitchfork$ and G; that is, for every k, we construct a homomorphism

$$H\colon C_k(T) \to C^{d-1-k}(X)$$

such that

$$f^\pitchfork(c) - G(c) = H(\partial c) + \delta H(c) \tag{10}$$

for $c \in C_k(T)$. Note that $G(c) = 0$ for $k > 0$ and that $H(c) = 0$ for $c \in C_d(M)$ (by convention, $C^{-1}(X) = 0$, since we work with the so-called *non-augmented cochain complex*, as in (9)).

This yields the desired contradiction: Given the triangulation T of M, the formal sum of all d-dimensional simplices of T is a d-dimensional cycle ζ_M (here we use that M has no boundary). Note that $f^\pitchfork(\zeta_M) = \mathbb{1}^0_X$ (every vertex v of X is mapped into the interior of a unique d-simplex of M) but $G(\zeta_M) = 0$. This is a contradiction, since

$$0 \neq \mathbb{1}^0_X = f^\pitchfork(\zeta_M) - G(\zeta_M) = \underbrace{H(\partial\zeta_M)}_{=0 \text{ since } \partial\zeta_M = 0} + \delta \underbrace{H(\zeta_M)}_{=0} = 0.$$

To complete the proof, we construct the chain-cochain homotopy H by induction on k.

For $k = 0$, we observe that for every vertex v of T, the cochains $f^\pitchfork(v)$ and $G(v) = f^\pitchfork(v_0)$ are cohomologous, i.e., their difference is a coboundary: Since we assume that M is connected, there is a 1-chain (indeed, a path) c in T with $\partial c = v - v_0$, and so $f^\pitchfork(v) - G(v) = f^\pitchfork(v - v_0) = \delta f^\pitchfork(c)$. For every vertex v of T, we set $H(v)$ to be a cofilling of $f^\pitchfork(v) - G(v)$ of minimal norm (if there is more than one minimal cofilling, we choose one arbitrarily). Thus, the homotopy condition (10) is satisfied for 0-chains (since chains and cochains of dimension less than zero or larger than d are, by convention, zero).

By choice of $H(v)$ and the coisoperimetric assumption on X, we have

$$\|H(v)\| \leq L \underbrace{\|f^\pitchfork(v) - f^\pitchfork(v_0)\|}_{<2\mu} < s_0 := 2L\mu.$$

Inductively, assume that we have already defined H on chains of dimension less than k and that $\|H(\rho)\| < s_i$ for every i-simplex of T, $i < k$, where s_i is a parameter that we will determine inductively. Thus, if τ is a k-simplex of T, then $H(\partial\tau)$ is already defined and has norm less than $(k+1)s_{k-1}$.

Moreover, we have $\|f^\pitchfork(\tau)\| \leq \frac{d}{k}\varepsilon \leq d\varepsilon$, by the sparsity assumption on X and since the triangulation T is sufficiently fine.

---

[5] That is, a homomorphism $G\colon C_k(T) \to C^{d-k}(X)$ for every k such that $G(\partial c) = \delta G(c)$ for $c \in C_k(T)$.

By construction, $z := f^{\pitchfork}(\tau) - H(\partial\tau)$ is a $(d-k)$-dimensional cocycle, and

$$\|z\| \leq \|f^{\pitchfork}(\tau) - H(\partial\tau)\| < d\varepsilon + (k+1)s_{k-1}. \tag{11}$$

If z is cohomologically trivial, i.e., $z \in B^{d-k}(X)$, then we define $H(\tau)$ to be a minimal cofilling of z and extend H to $C_k(T)$ by linearity. By assumption on X, we get

$$\|H(\tau)\| < s_k := L\left(d\varepsilon + (k+1)s_{k-1}\right).$$

Note that this recursion yields $s_k = d\varepsilon(L + \cdots + L^k) + (k+1)!L^{k+1}2\mu$.

If z is nontrivial,[6] then by the assumption on large cosystoles and (11),

$$\vartheta \leq \|z\| < d\varepsilon + (k+1)s_{k-1},$$

which is a contradiction if we choose μ and ε_0 (and hence ε) sufficiently small with respect to d, L and ϑ. ◀

▶ **Remarks 17.**

1. In many interesting cases, X belongs to an infinite family of complexes for which the local sparsity parameter ε tends to zero as the size of the complex increases. For instance, if X is the d-skeleton of the n-simplex, $n \to \infty$, then we have $\varepsilon = O(1/n)$. For complexes with local sparsity $\varepsilon = o(1)$, the the above proof yields $\mu \geq \frac{\vartheta}{2(k+1)!L^k} + o(1)$. If M is unbounded, then, as remarked in the proof, we can take the vertex v_0 to satisfy $f^{\pitchfork}(v_0) = 0$, which improves the estimate by a factor of 2.
   More quantitative information and better bounds on the overlap constant (which are of interest for specific families of complexes, e.g., skeleta of simplices) can be gleaned from the proof by a more refined analysis through the *cofilling profiles* of X [6], which estimate the size of a minimal cofilling of a cocycle b as a possibly nonlinear function of $\|b\|$. Further improvements in the estimates are possible trough the notion of *pagodas* [11].
2. The proof of the overlap theorem is very robust and easily generalizes to other settings, in particular to other coefficient rings and other norms. Suppose that R is a fixed ring of coefficients (commutative, with 1), and consider (co)chains and (co)homology with R-coefficients. If R is not of characteristic 2, we need to add some minor assumptions to deal with orientations. First, we need to assume that he target manifold M is R-orientable, i.e., that $H_d(M;R) \cong R$, generated by a fundamental homology class $[M]$. The definition of the intersection number changes slightly: if two oriented linear simplices σ, τ of complementary dimensions in M intersect transversely in a single point, then their orientations determine a local orientation of M, and we set the intersection number $\sigma \cdot \tau$ to be $+1$ or -1 depending on whether this orientation agrees with the chosen global orientation of M or not.
   Second, we need to assume that the norm of a cochain is invariant under sign changes in the values of the cochain, i.e., if two k-cochains $c, c' \in C^k(X;R)$ satisfy $c(\sigma) = \pm c'(\sigma)$ for every orientated k-cell σ of X (the signs may be different for different σ), then $\|c\| = \|c\|$. With these additional assumptions, the proof of Theorem 8 goes through also for R-coefficients and yields that for every $f\colon X \to M$, there exists $p \in M$ such (6) holds.

[6] Note that in the special case that X is connected and $k = d$, the only nontrivial 0-cocycle is $z = \mathbb{1}^0_X$, hence $\|z\| = 1$.

3. For norms other than the normalized Hamming norm, $\|f^{\pitchfork}(p)\| \geq \mu$ does not necessarily imply that (1) holds. For instance, suppose that $R = \mathbb{R}$ and that we work with the ℓ_2-norm. In this case, large norm $\|f^{\pitchfork}(p)\|$ might be caused by a single d-simplex σ such that $f^{\pitchfork}(p)(\sigma)$ is a large integer, i.e., $f(\sigma)$ intersects p with large multiplicity. However, this problem does not occur if we impose additional assumptions on the map f, e.g., that $f^{\pitchfork}(p)(\sigma)$ is bounded by some constant K in absolute value (e.g., if f is linear, then we can take $K = 1$).
4. We used the assumption that M is piecewise-linear in order to apply standard general position arguments from piecewise-linear topology. We believe that the result holds more generally if M is a homology manifold. General position arguments for homology manifolds are much more subtle, but for the proof we do not really need to perturb the map f to general position (which may not be possible), we only need a general position chain map that is close to the chain map induced by f. We plan to investigate this in more detail in a future paper.

Acknowledgement. We would like to thank the anonymous referees for many helpful remarks concerning the presentation.

References

1 I. Bárány. *A generalization of Carathéodory's theorem.* Discrete Math. 40(2–3):141–152, 1982.
2 E. Boros and Z. Füredi. *The number of triangles covering the center of an n-set.* Geom. Dedicata 17, 69–77, 1984.
3 S. Evra and T. Kaufman. *Systolic Expanders of Every Dimension.* Preprint, `http://arxiv.org/abs/1510.00839v2`.
4 D. Burago, Y. Eliashberg, M. Bestvina, F. Forstnerič, L. Guth, A. Nabutovsky, A. Phillips, J. Roe, and A. Vershik. *A Few Snapshots from the Work of Mikhail Gromov.* In: H. Holden and R. Piene (Eds.). *The Abel Prize, 2008–2012*, Springer-Verlag, Berlin 2014, pp. 139–234.
5 T. Kaufman, D. Kazhadan, A. Lubotzky. *Isoperimetric Inequalities for Ramanujan Complexes and Topological Expanders*, Proceedings of the 55th Annual Symposium on Foundations of Computer Science (FOCS), 2014, pp. 484–493.
6 M. Gromov. *Singularities, expanders and topology of maps. Part 2: From combinatorics to topology via algebraic isoperimetry,* Geom. and Funct. Analysis, 20(2):416–526, 2010.
7 R. Karasev. *A simpler proof of the Boros–Füredi–Bárány–Pach–Gromov theorem.* Discrete & Computational Geometry 47(3), 492–495, 2012.
8 N. Linial and R. Meshulam. *Homological connectivity of random 2-complexes,* Combinatorica, 26(4):475–487, 2006.
9 I. Mabillard and U. Wagner. *Eliminating higher-multiplicity intersections, I. A Whitney trick for Tverberg-type problems.* Preprint, `http://arxiv.org/abs/1508.02349v1`.
10 J. Matoušek. *Using the Borsuk–Ulam theorem.* Springer-Verlag, Berlin, 2003.
11 J. Matoušek and U. Wagner. *On Gromov's method of selecting heavily covered points.* Discrete & Computational Geometry 52(1), 1–33, 2014.
12 V. Prasolov. *Elements of combinatorial and differential topology.* American Mathematical Society, Providence, RI, 2006.
13 C. P. Rourke and B. J. Sanderson, *Introduction to piecewise-linear topology.* Springer-Verlag, New York, 1972.
14 E. C. Zeeman. Seminar on Combinatorial Topology. Lecture Notes, Institut des Hautes Études Scientifiques, 1963. Scanned notes available at `http://www.maths.ed.ac.uk/~aar/papers/zeemanpl.pdf`.

On the Number of Maximum Empty Boxes Amidst n Points

Adrian Dumitrescu[1] **and Minghui Jiang**[2]

1 **Department of Computer Science, University of Wisconsin, Milwaukee, USA**
dumitres@uwm.edu

2 **Department of Computer Science, Utah State University, Logan, USA**
mjiang@cc.usu.edu

Abstract

We revisit the following problem (along with its higher dimensional variant): Given a set S of n points inside an axis-parallel rectangle U in the plane, find a maximum-area axis-parallel sub-rectangle that is contained in U but contains no points of S.

1. We prove that the number of maximum-area empty rectangles amidst n points in the plane is $O(n \log n\, 2^{\alpha(n)})$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackermann's function. The previous best bound, $O(n^2)$, is due to Naamad, Lee, and Hsu (1984).
2. For any $d \geq 3$, we prove that the number of maximum-volume empty boxes amidst n points in $\mathbb{R}^d$ is always $O(n^d)$ and sometimes $\Omega(n^{\lfloor d/2 \rfloor})$. This is the first superlinear lower bound derived for this problem.
3. We discuss some algorithmic aspects regarding the search for a maximum empty box in $\mathbb{R}^3$. In particular, we present an algorithm that finds a $(1-\varepsilon)$-approximation of the maximum empty box amidst n points in $O(\varepsilon^{-2}\, n^{5/3} \log^2 n)$ time.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Maximum empty box, Davenport-Schinzel sequence, approximation algorithm,data mining.

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.36

1 Introduction

Given an axis-parallel rectangle U in the plane containing n points, Maximum Empty Rectangle is the problem of computing a maximum-area axis-parallel empty sub-rectangle contained in U. This problem is one of the oldest in computational geometry, with multiple applications, e.g., in facility location problems [31]. In higher dimensions, finding the largest empty box has applications in data mining, such as finding large gaps in a multi-dimensional data set [21].

A *box* in $\mathbb{R}^d$, $d \geq 2$, is an open axis-parallel hyperrectangle $(a_1, b_1) \times \cdots \times (a_d, b_d)$ with $a_i < b_i$ for $1 \leq i \leq d$. Due to the fact that the volume ratio of any box inside another box is invariant under scaling, the problem can be reduced to the case when the enclosing box is a hypercube. Given a set S of n points in the unit hypercube $U_d = [0,1]^d$, $d \geq 2$, an *empty box* is a box empty of points in S and contained in U_d, and Maximum Empty Box is the problem of finding an empty box with the *maximum* volume. Note that an empty box of maximum volume must be *maximal* with respect to inclusion. Some planar examples of maximal empty rectangles are shown in Fig. 1. All rectangles and boxes considered in this paper are axis-parallel.


© Adrian Dumitrescu and Minghui Jiang;
licensed under Creative Commons License CC-BY

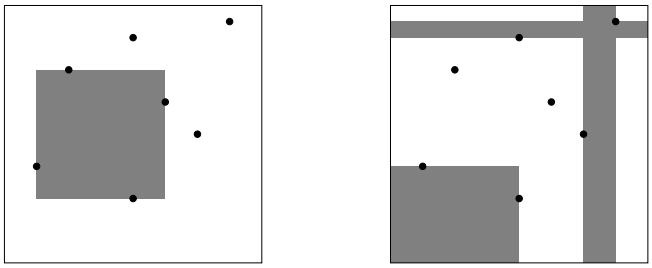


Figure 1 A maximal empty rectangle supported by one point on each side (left), and three maximal empty rectangles supported by both points and sides of $[0,1]^2$ (right).

By slicing the hypercube with n parallel hyperplanes, each incident to one of the n points, the largest slice gives an empty box of volume at least $\frac{1}{n+1}$. For a fixed dimension d, one can exhibit point sets for which the largest empty box is $O\left(\frac{1}{n}\right)$, and so the above estimate is the best possible, apart from constant factors (that depend on d) [34, 17, 5]. Knowing what to hope for in regard to the maximum volume, the next natural question is: how many such extremal boxes can there be?

The study of extremal problems on triangle areas determined by n points was initiated in a series of papers by Erdős and Purdy in the early 1970s [22]. It is known for instance that for n points in the plane, the maximum area of a triangle can occur at most n times [10].

In Section 3 we show that the number of maximum empty rectangles amidst n points is $O(n \log n\, 2^{\alpha(n)})$, where $\alpha(n)$ is the extremely slowly growing inverse of Ackermann's function. The previous best upper bound, due to Naamad, Lee, and Hsu from 1984 [31], is $O(n^2)$; this quadratic upper bound has been revisited numerous times [2, 3, 4, 6, 11, 14, 27, 34] during the last 30 years without any progress being made.

In Section 4 we show that the number of maximum empty boxes amidst n points in $\mathbb{R}^d$ is always $O(n^d)$ and sometimes $\Omega(n^{\lfloor d/2 \rfloor})$. For $d \geq 4$, this is the first superlinear lower bound derived for the empty box problem.

In Section 5 we discuss two algorithmic applications: (i) Given n points inside a hyperrectangle U in $\mathbb{R}^3$, a $(1-\varepsilon)$-approximation of the maximum volume of an empty box contained in U can be computed in $O(\varepsilon^{-2}\, n^{5/3} \log^2 n)$ time. (ii) If the data structure of Kaplan et al. [27] for the maximum empty rectangle containing a query point has a semi-dynamic extension that allows the deletion of points, with $o(n)$ amortized time per point deletion, then there is a subcubic-time exact algorithm for the Maximum Empty Box problem in $\mathbb{R}^3$.

2 Background and related work

Bounds on the volume of a maximum empty box. Given a set S of n points in the unit hypercube $U_d = [0,1]^d$, where $d \geq 2$, let $A_d(S)$ be the maximum volume of an empty box contained in U_d, and let $A_d(n)$ be the minimum value of $A_d(S)$ over all sets S of n points in U_d.

Rote and Tichy [34] proved that $A_d(n) = \Theta\left(\frac{1}{n}\right)$ for any fixed $d \geq 2$. From one direction, for any $d \geq 2$, we have

$$A_d(n) < \left(2^{d-1} \prod_{i=1}^{d-1} p_i\right) \cdot \frac{1}{n}, \tag{1}$$

where p_i is the ith prime, as shown in [34, 17] using Halton-Hammersley generalizations [24, 25] of the van der Corput point set [12, 13]; see also [29, Ch. 2.1].

From the other direction, by slicing the hypercube with n parallel hyperplanes, each incident to one of the n points, the largest slice yields the lower bound $A_d(n) \geq \frac{1}{n+1}$ for each d. This trivial estimate can be improved using the following observation [17, 19] that relates $A_d(n)$ to $A_d(b)$ for fixed d and b:

▶ **Lemma 1** ([19]). *For any $d \geq 2$ and $b \geq 2$, $A_d(n) \geq [(b+1)A_d(b) - o(1)] \cdot \frac{1}{n}$.*

In particular, with $b = 4$ in Lemma 1, the following bound[1] was obtained in [17]:

$$A_d(n) \geq A_2(n) \geq (5A_2(4) - o(1)) \cdot \frac{1}{n} = (1.25 - o(1)) \cdot \frac{1}{n}.$$

By exploiting the above observation in a more subtle (and fruitful) way, Aistleitner et al. [5] recently showed that the dependence on d in the volume bound is necessary, that is, the maximum volume grows with the dimension d:

$$A_d(n) \geq \frac{\log d}{4(n + \log d)}. \tag{2}$$

Further, one can show that their proof leads to an efficient algorithm that computes such a box. A broader analysis of this approach will be the focus of a forthcoming paper [20].

Complexity of computing a maximum empty box. Several algorithms have been proposed in the 1980s for the Maximum Empty Box problem in the plane; see [17, 19] and the references therein. The fastest one, due to Aggarwal and Suri [4], runs in $O(n \log^2 n)$ time and $O(n)$ space. On the other hand, a lower bound of $\Omega(n \log n)$ in the algebraic computation tree model already holds for the one-dimensional variant [33, pp. 260–262]: Given n points in a line, find the largest gap between two consecutive points.

For $d \geq 3$, the only known approach for computing a largest empty box is by examining *all* candidates, i.e., all maximal empty boxes. As noted, an empty box of maximum volume must be maximal with respect to inclusion. Given a set S of n points in the unit hypercube $U_d = [0,1]^d$, $d \geq 2$, let $k = k(S)$ and $\kappa = \kappa(S)$, respectively, denote the number of *maximal* empty boxes and the number of empty boxes of *maximum* volume contained in U_d. Clearly $\kappa(S) \leq k(S)$.

Chazelle et al. [11] show that in the plane, *all* largest empty rectangles can be reported in $O(k + n \log n)$ time (note the dependence on k, not on κ, in all time-bounds). Nandy and Bhattacharya [32] show that in $\mathbb{R}^3$, *all* maximum empty boxes can be reported in $O(k + n^2 \log n)$ time. A result of Kaplan, Rubin, Sharir, and Verbin [28] implies an output-sensitive algorithm for Maximum Empty Box running in $O((n+k) \log^{2d-1} n)$ time, for any fixed $d \geq 2$. Subsequently, Backer and Keil [8, 9] reported an output-sensitive algorithm running in $O(k \log^{d-2} n)$ time, for any fixed $d \geq 3$.

One can easily exhibit point sets for which the number k of maximal empty boxes is $\Omega(n^d)$, for any fixed $d \geq 2$; see [28, 8, 17]. Naamad, Lee, and Hsu [31] showed that for $d = 2$, the maximum number of maximal empty rectangles is $O(n^2)$, and that this bound is tight. Kaplan et al. [28] proved (indirectly, see [18, p. 479]) that the maximum number of maximal empty boxes is $O(n^d)$ for each fixed d, confirming an earlier conjecture of Datta and Soundaralakshmi [14].

Since the maximum number of maximal empty boxes is $\Theta(n^d)$ for each fixed d, any algorithm that computes a maximum-volume empty box by enumerating all maximal empty

[1] A weaker bound with $b = 3$ was inadvertently labeled as an improvement over this bound in [19].

boxes is bound to be inefficient in the worst case. In fact, Backer and Keil [8, 9] proved that Maximum Empty Box is NP-hard when the dimension d is part of the input, and Giannopoulos, Knauer, Wahlström, and Werner [23] further showed that the problem is W[1]-hard with the dimension d as the parameter. Further, the W[1]-hardness of the problem implies [23, Corollary 1] that the existence of an exact algorithm running in $n^{o(d)}$ time is unlikely. On the other hand, the output-sensitive algorithms of Kaplan et al. [28] and Backer and Keil [9] would run faster when there are only a few maximal empty boxes, and indeed, when d is fixed, Dumitrescu and Jiang [18] proved that the expected number of maximal empty boxes amidst n random points in the unit hypercube $[0,1]^d$ in $\mathbb{R}^d$ is

$$(1 \pm o(1))\, \frac{(2d-2)!}{(d-1)!}\, n \ln^{d-1} n.$$

So while for fixed d, the expected running times of the output-sensitive algorithms in [9, 28] on random point sets are $O(n \log^{O(d)} n)$, their worst-case running times remain $\Omega(n^d)$.

In terms of approximation, Dumitrescu and Jiang [17] gave an algorithm that finds an empty box whose volume is at least $1-\varepsilon$ times the optimal in $O\big((\frac{8ed}{\varepsilon^2})^d \cdot n \log^d n\big)$ time; a faster approximation algorithm for the related Maximum Empty Cube problem was also provided. This approximation algorithm is fixed parameter tractable with both d and $1/\varepsilon$ as parameters [23, Section 1.5].

Notations. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. As usually, Θ, O, Ω notation is used to describe the asymptotic growth of functions. Given a set S of n points in the unit hypercube $U_d = [0,1]^d$, where $d \geq 2$, let $X_d(n)$ be the maximum value of $\kappa(S)$ over all sets S of n points in U_d (where $\kappa(S)$ denotes the number of empty boxes of maximum volume contained in U_d, amidst the points in S).

3 The number of empty rectangles of maximum area

As mentioned earlier, the number of maximal empty rectangles is $O(n^2)$, and this bound can be attained. In contrast, we prove (see Theorem 2) that the number of empty rectangles of maximum area is linear or nearly linear. Refer for brevity to any empty rectangle of maximum area as an *maximum empty rectangle*; and *maximum empty box* for $d \geq 3$. The arguments we need are based on Davenport-Schinzel sequences.

A sequence $a_1, a_2, \ldots$ over the alphabet $\Sigma = \{1, 2, \ldots, m\}$ is called an (m, s)-*Davenport-Schinzel sequence of order* s, if (i) it has no two consecutive elements which are the same, and (ii) it has no *alternating* subsequence of length $s+2$, i.e., there are no indices $1 \leq i_1 < i_2 < \ldots < i_{s+2}$ such that

$$a_{i_1} = a_{i_3} = a_{i_5} = \ldots = a, \quad a_{i_2} = a_{i_4} = a_{i_6} = \ldots = b,$$

where $a \neq b$. Let $\lambda_s(m)$ denote the maximum length of an (m, s)-Davenport-Schinzel sequence [15]; see also [35] and [30, Ch. 7]. Obviously, we have $\lambda_1(m) = m$, and one can verify that $\lambda_2(m) = 2m - 1$, for every m. It was shown by Hart and Sharir [26] that $\lambda_3(m) = O(m\alpha(m))$ and $\lambda_4(m) = O(m\, 2^{\alpha(m)})$, where $\alpha(m)$ is the extremely slowly growing inverse of Ackermann's function, and that this estimate is asymptotically tight. They also proved that $\lambda_s(m)$ is only slightly superlinear in m, for every fixed $s \geq 4$. For the currently best bounds of this type, see [1].

▶ **Theorem 2.** *We have* $X_2(n) = \Omega(n)$ *and* $X_2(n) = O(\lambda_4(n) \log n) = O(n \log n\, 2^{\alpha(n)})$, *where* $\alpha(n)$ *is the (slowly growing) inverse of Ackermann's function.*

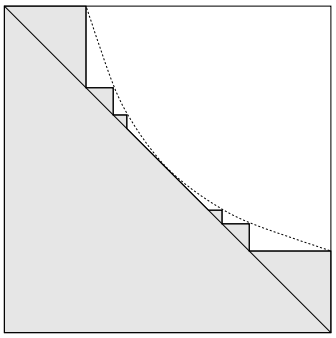


Figure 2 A hyperbola construction for the linear lower bounds on $X_2(n)$.

3.1 Proof of Theorem 2

Lower bound. We start with a construction that yields a lower bound of $X_2(n) \geq n+1$. Consider the line $x+y = 2\sqrt{c'}$ and the hyperbola $xy = c$ inside $U = [0,1]^2$, where $1/4 \leq c' \leq c < 1$. They are disjoint when $c' < c$, and tangent at $(\sqrt{c'}, \sqrt{c'})$ when $c' = c$. For any c' and c, $1/4 \leq c' \leq c < 1$, we can obtain a sequence $S = S(c', c)$ of points inside U on the line $x+y = 2\sqrt{c'}$ as follows. Start at the right end $(1, c)$ of the hyperbola, shoot a ray to the left until it hits the line at $p_1 = (2\sqrt{c'} - c, c)$. While the x-coordinate of p_i is greater than that of the left end $(c, 1)$ of the hyperbola, let the ray continue upward until it hits the hyperbola, and then to the left until it hits the line again at p_{i+1}. Refer to Fig. 2 for the limiting case when $c \to c' = 1/4$.

Fix $1/2 \leq c < 1$. As c' continuously increases from $1/4$ to c, the number of points p_i in the sequence S gradually increases from 1 and tends to ∞. For each $n \geq 1$, there is a range of c' such that S contains exactly n points inside U on the line $x+y = 2\sqrt{c'}$. Moreover, as c' increases continuously in this range, the n points move continuously on this line. There is a particular value of c', $1/4 \leq c' < c$, such that the n points are placed symmetrically about the center $(\sqrt{c'}, \sqrt{c'})$ of the line inside U. Observe that all (final) point coordinates are at least c. With these n points in U, there are exactly $n+1$ maximum empty rectangles with area c, all having lower-left corners at $(0,0)$ and upper-right corners on the hyperbola.

We next give a more refined construction which yields a better, but still linear, lower bound of $X_2(n) \geq 2n+2$. Again consider the line $x+y = 2\sqrt{c'}$ and the hyperbola $xy = c$ inside U, and refer to Fig. 2.

Fix $c' = 1/4$. Then the line $x+y = 2\sqrt{c'}$ is the diagonal $x+y = 1$ of U. As c continuously decreases from 1 to c', the number of points p_i in the sequence S gradually increases from 1 and tends to ∞. For each $n \geq 1$, there is a range of c such that S contains exactly n points inside U on the diagonal $x+y = 1$. Moreover, as c decreases continuously in this range, the n points move continuously on this diagonal. There is a particular value of c, $1/4 < c \leq 1/2$, such that the n points are placed symmetrically about the center $(1/2, 1/2)$ of the diagonal. With these n points in U, there are exactly $2n+2$ maximum empty rectangles with area c, among which $n+1$ have lower-left corners at $(0,0)$, and the other $n+1$ have upper-right corners at $(1,1)$, symmetric about the diagonal $x+y = 1$.

Upper bound. We proceed by induction on n and develop a recurrence on $X(n) = X_2(n)$. Consider a *balanced partition* of the points by a vertical line into two sets, with $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ points, respectively: $S = L \cup R$ (points on the line are assigned as needed to any one of the two parts). We may suppose that the vertical line is the y-axis.

Following the terminology in [11], a *support* of a maximum empty rectangle R is either a point in S or an edge of U that limits the expansion of R in that direction: leftward, rightward, upward or downward. Every maximum empty rectangle has 4 supports. Every maximum empty rectangle is either contained in L, or contained in R, or crosses the y-axis. Let Y denote the set of maximum empty rectangles crossing the y-axis. A rectangle in Y can have either (i) 2 supports on the left side of the y-axis and 2 supports on the right side of the y-axis; or (ii) 3 supports on one side of the y-axis and 1 support on the other side of the y-axis.

Naamad et al. [31] showed that the number of maximal empty rectangles with at least one edge support is $O(n)$. Thus we immediately get:

▶ **Lemma 3.** *The number of maximum empty rectangles in Y with at least one edge support is $O(n)$.*

We next establish upper bounds for the remaining types of maximum empty rectangles in Y, with 4 points of support (a point of support is also referred to as *defining point* in [27]). For the remainder of this section we only consider this type of maximum empty rectangles.

Chazelle et al. [11] showed that the number of maximal empty rectangles with 3 points of support on one side of an axis and one point of support on the other side of that axis is $O(n)$; see also [27, p. 346]. Thus we also have:

▶ **Lemma 4.** *The number of rectangles in Y with 3 points of support on one side of the y-axis and one point of support on the other side of the y-axis is $O(n)$.*

The key argument is provided by the following.

▶ **Lemma 5.** *The number of rectangles in Y with two points of support on each side of the y-axis is $O(\lambda_4(n))$.*

Proof. By symmetry, it suffices to show that the number h of maximum empty rectangles with bottom and left supports in L and with top and right supports in R is $O(\lambda_4(n))$.

Consider a maximum empty rectangle H amidst the points $L \cup R$ in U. Let U_- and U_+, respectively, be the two subrectangles of U on the left and the right side of the y-axis. Then the bottom-left corner of H is the bottom-left corner of a maximal empty rectangle amidst L in U_- with the y-axis as the right support, and the top-right corner of H is the top-right corner of a maximal empty rectangle amidst R in U_+ with the y-axis as the left support. Naamad et al. [31] showed that amidst any n points in a rectangle, the number of maximal empty rectangles with at least one edge support is $O(n)$. We thus obtain l candidate bottom-left corners $p_1, \ldots, p_l$ on the left side of the y-axis, and r candidate top-right corners $q_1, \ldots, q_r$ on the right side of the y-axis, where $l + r = O(n)$, such that each maximum empty rectangle under our consideration has some p_i as the bottom-left corner and some q_j as the top-right corner.

Order the candidate corners on the left such that $y(p_1) \geq \ldots \geq y(p_l)$, and corners with the same y-coordinate are ordered from left to right. Similarly, order the candidate corners on the right such that $y(q_1) \geq \ldots \geq y(q_r)$, and corners with the same y-coordinate are ordered from left to right.

We next construct a sequence w over the alphabet $\Sigma = \{p_1, \ldots, p_l, q_1, \ldots, q_r\}$ of $m = l + r$ symbols; recall that $m = O(n)$. Initialize w to an empty string. For each $i = 1, \ldots, l$, first append the symbol p_i to w, then for each $j = 1, \ldots, r$, if there is a maximum empty rectangle with p_i as the bottom-left corner and with q_j as the top-right corner, append the symbol q_j to w. The length of the string w thus constructed is $l + h$; recall that h is the number of

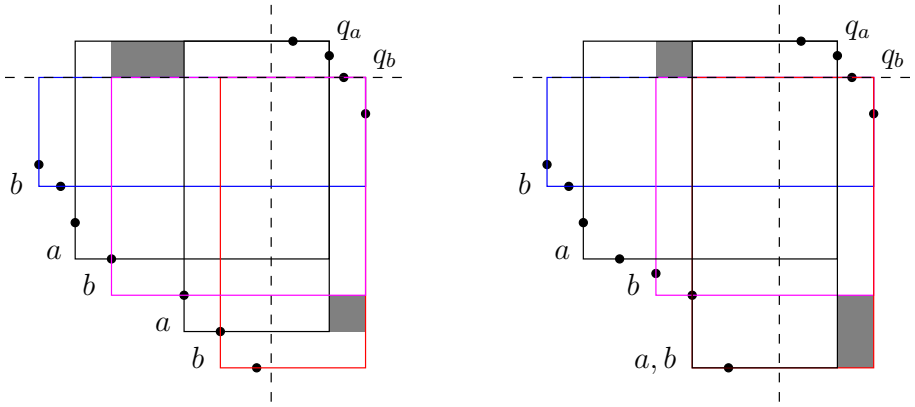


Figure 3 Extending the union of two maximum empty rectangles selected when reading a subsequence of the form *babab* with $a < b$. The figure shows two examples.

maximum empty rectangles under consideration. No two consecutive symbols in w are the same, because each symbol p_i occurs exactly once in w, and each symbol q_j occurs at most once after p_i and before p_{i+1}. We claim that w has no alternating subsequence of length 6, i.e., no subsequence of the form $\alpha\beta\alpha\beta\alpha\beta$; here α, β are elements of $\{q_1, \ldots, q_r\}$, since each p_i occurs exactly once. Equivalently, we claim that if $1 \le a < b \le r$, then w has no subsequence of the forms $q_aq_bq_aq_bq_aq_b$ or $q_bq_aq_bq_aq_bq_a$. If such a subsequence exists, we extract out of it a subsequence of length five of the form $q_bq_aq_bq_aq_b$, where $a < b$.

Suppose for contradiction that w has a subsequence $q_bq_aq_bq_aq_b$, where $a < b$, so $y(q_a) \ge y(q_b)$. By our construction of w, the corner q_a cannot have both coordinates greater than those of q_b. Suppose the contrary, that $x(q_a) > x(q_b)$ and $y(q_a) > y(q_b)$. Then the maximum empty rectangle corresponding to the first occurrence of the symbol q_a in $q_aq_bq_aq_b$ would contain the corner q_b and hence a point of R with the same y-coordinate as q_b, contradicting its emptiness assumption.

Label each corner p_i that is the bottom-left corner of some maximum empty rectangle with top-right corner at q_a (respectively, q_b) with the index a (respectively b). Then we have a sequence of bottom-left corners p_i on the left side of the y-axis and below the horizontal line through q_b, with non-decreasing y-coordinates and non-increasing x-coordinates, as illustrated in Fig. 3, where each corner is labeled with either a or b or ab (note that two maximum empty rectangles with different top-right corners may have the same point in L as the bottom support), and concatenation of all these labels is the index sequence *babab* corresponding to the point sequence $q_bq_aq_bq_aq_b$.

Indeed: (i) since when constructing w, i goes from 1 to l, the sequence of corners has non-increasing y-coordinates; (ii) suppose for contradiction that the sequence of corners does not have non-decreasing x-coordinates, i.e., for some $i < i'$ we have $p_i, p_{i'}$ as lower-left corners corresponding to a subsequence q_bq_a in w, with $x(p_{i'}) < x(p_i)$ and $y(p_{i'}) < y(p_i)$; then, as in a previous argument, the maximum empty rectangle corresponding to the occurrence of q_a in q_bq_a would contain the corner q_b and hence a point of L with the same y-coordinate as q_b, contradicting its emptiness assumption; consequently, the sequence of corners has non-decreasing x-coordinates.

Note that the shaded area above the horizontal line through q_b is contained in the maximum empty rectangle with bottom-left corner labeled with the first a, and that the shaded area on the right side of the y-axis is contained in the maximum empty rectangle

with bottom-left corner labeled with the third b. Thus they are both empty. The reader may reflect at this point on the significance of the occurrence of a subsequence $babab$, with $a < b$. The first b in $babab$ was used only to ensure that the a after it has smaller y-coordinate than q_b. The last b in $babab$ was used to ensure that the right shaded area is empty.

We are now in position to finalize the argument. Consider the two maximum empty rectangles with bottom-left corners labeled with the second b and the second a, respectively. By adding the two shaded areas, the union of the two maximum empty rectangles can be extended into the union of two other empty rectangles with the same bottom-left corners and with the top-right corners swapped. Since the two pairs of rectangles have the same intersection, it follows that the total area of the two extended empty rectangles is larger than the total area of the two maximum empty rectangles, a contradiction.

Since w is an $(m, 4)$-Davenport-Schinzel sequence, its length at most $\lambda_4(m)$. It then follows that $h \leq \lambda_4(m) = O(\lambda_4(n))$, concluding the proof of Lemma 5. ◀

Let $Y(n)$ denote the total number of rectangles in Y. By Lemmas 3, 4, 5, we have $Y(n) = O(\lambda_4(n))$. By the Divide & Conquer approach we have

$$X(n) \leq X\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + X\left(\left\lceil \frac{n}{2} \right\rceil\right) + Y(n),$$

thus $X(n)$ satisfies the following recurrence:

$$X(n) \leq X\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + X\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(\lambda_4(n)). \tag{3}$$

The solution of this recurrence is $X(n) = O(\lambda_4(n) \log n) = O(n \log n\, 2^{\alpha(n)})$, and this completes the proof of the planar upper bound in Theorem 2.

A more refined upper bound on $X_2(n)$. Let A be the maximum area of an empty rectangle in U. We next show that the number of maximum empty boxes is $O(\lambda_4(n) \log(1/A))$. In particular, $A = \Omega(1/n)$ hence $\log(1/A) = O(\log n)$. Also, when A is a constant, the bound is almost tight: the maximum number of maximum empty boxes is $\Omega(n)$ and $O(n2^{\alpha(n)})$ in this case.

We can get this slightly stronger formulation of our current result because every maximum empty rectangle has area A and hence has width at least A. Thus we can group all maximum empty rectangles by their widths into $\log(1/A)$ groups, such that the widths of all maximum empty rectangles in each group differ by a factor of at most 2, and then argue that the number of maximum empty rectangles in each group is at most $O(\lambda_4(n))$.

For each group, let the width of the maximum empty rectangles be between w and $2w$. Then we can use $1/w$ evenly-spaced vertical lines to stab all maximum empty boxes at least once and at most twice. Using Lemmas 4 and 5 with each vertical line as the y-axis yields the $O(\lambda_4(n))$ bound for each group.

4 The number of maximum empty boxes in $\mathbb{R}^d$

In this section we prove the following.

▶ **Theorem 6.** *For any fixed $d \geq 3$, we have $X_d(n) = \Omega(n^{\lfloor d/2 \rfloor})$ and $X_d(n) = O(n^d)$.*

4.1 Proof of Theorem 6

Upper bound. The upper bound $X_d(n) = O(n^d)$ immediately follows from the same upper bound on the on the number of *maximal empty boxes*, as established (indirectly) in [28]. We next recall the main lines of reasoning in obtaining this result.

Motivated by a related problem, Coloured Orthogonal Range Counting, Kaplan, Rubin, Sharir and Verbin [28] estimated from above the number of maximal empty boxes. In the context of this related problem, each input point in $\mathbb{R}^d$ can be transformed into a point in $\mathbb{R}^{2d}$ by splitting the coordinate x along the ith axis into two coordinates x and $-x$ in the $(2i-1)$th and $(2i)$th axes, respectively, such that each maximal empty box in $\mathbb{R}^d$ becomes a maximal empty orthant in $\mathbb{R}^{2d}$. Kaplan et al. then provided an upper bound of $O(n^{\lfloor d/2 \rfloor})$ on the maximum number of maximal empty orthants amidst n points in $\mathbb{R}^d$, which, by the transformation, implies an upper bound of $O(n^d)$ on the maximum number of maximal empty boxes amidst n points in $\mathbb{R}^d$. Moreover they also provided matching lower bounds of $\Omega(n^{\lfloor d/2 \rfloor})$ for maximal empty orthants and $\Omega(n^d)$ for maximal empty boxes. Thus the maximum number of maximal empty boxes amidst n points in $\mathbb{R}^d$ is $\Theta(n^d)$. We refer to [18] for a historical account of related work on these bounds.

Since maximum empty boxes are also maximal empty boxes, the known upper bound for maximal empty boxes [28] immediately implies the upper bound $X_d(n) = O(n^d)$ for maximum empty boxes. In the following, we derive the lower bound $X_d(n) = \Omega(n^{\lfloor d/2 \rfloor})$.

Lower bound. Let $d \geq 4$ be even. Pair up the d dimensions into $d/2$ pairs, where each pair determines a 2-dimensional plane. Let $n = k \cdot (d/2)$. Associate k distinct points with each of the $d/2$ planes. Let $1/2 < c < 1$. In each of the $d/2$ planes, set the two coordinates of the k points associated with this plane as in the first planar construction in the proof of the lower bound in Theorem 2, such that there are $k+1$ maximum empty rectangles of area $c^{2/d}$ amidst the k points in that plane. Set the other $d-2$ coordinates of these k points to c. Note that $c^{2/d} > c$ and thus all point coordinates are at least c and less than 1.

▶ **Claim 7.** *The maximum volume of an empty box is c. The number of maximum empty boxes is $(k+1)^{d/2} + d$.*

To verify the claim, we first show that there are $(k+1)^{d/2} + d$ maximal empty boxes of volume c. Since all coordinates of all points are at least c, a box that projects to the interval $(0, c)$ along any one of the d axes and to the whole interval $(0, 1)$ along the other $d-1$ axes is empty. This box is maximal because all points are interior and there are points of coordinate c in each dimension. There are d boxes of this type.

Take any one of the $k+1$ maximum empty rectangles from each of the $d/2$ planes, the product of these $d/2$ rectangles is a box. This box is empty of all n points because it is empty of the k distinct points associated with each of the $d/2$ planes. There are $(k+1)^{d/2}$ boxes of this type, each of volume $(c^{2/d})^{d/2} = c$. For fixed d, the number of boxes is $\Omega(n^{d/2})$.

We next show that the maximum volume of an empty box is c, and hence these $(k+1)^{d/2}+d$ boxes are all maximum empty boxes.

Let B be an arbitrary empty box of volume at least c. Then the projection of B to each axis is an interval of length at least c; moreover, if the projection of B to one axis is an interval of length exactly c, then its projection to all other axes must have length 1. Since there is at least one point of coordinate c along each axis, B either projects to the interval $(0, c)$ along one axis and hence has to project to the whole interval $(0, 1)$ along all other axes, or projects to an interval that contains c along every axis. There are clearly d empty boxes

of volume c of the first type. In the following, we assume that B belongs to the second type, that is, the projection of B to each axis contains c.

Let B_i and B_π denote the projection of B onto dimension i and plane π respectively. Consider one of the $d/2$ planes, say, the x_1x_2 plane π. Recall that the k points associated with this plane have coordinate c in the other $d-2$ dimensions, and so their projections onto any axis x_i, $i \geq 3$, are contained in the interior of the corresponding interval B_i. On the other hand, their projections onto π are not contained in the interior of B_π, since otherwise B would not be empty.

Thus B_π is an empty rectangle in the plane π and so its area is at most $c^{2/d}$, which is the area of a maximum empty rectangle in this plane. By the same argument, the projection of B to each of the $d/2$ planes must have area at most $c^{2/d}$. Hence the volume of B is at most $(c^{2/d})^{d/2} = c$. Since the volume of B is assumed to be at least c, its volume must be exactly c. Moreover, the projection of B to each of the $d/2$ planes must have area exactly $c^{2/d}$ and must be one of the $k+1$ maximum empty rectangles. The claim has been verified.

Let $d \geq 3$ be odd. Then $d-1 \geq 2$ is even. Take n points in U_{d-1} attaining the lower bound for even dimension, with maximum volume c, where $1/2 < c < 1$. Set the dth coordinate to c for all n points, now in U_d. Observe that the hyperplane $x_d = c$ divides $U_d = [0,1]^d$ into two maximal empty boxes of volumes c and $1-c < c$, respectively, and that all other maximal empty boxes in U_d project to maximal empty boxes in U_{d-1}. For this point set the maximum volume of an empty box is again c, and there is exactly one extra maximum empty box, namely $(0,1)^{d-1} \times (0,c)$, in addition to the maximum empty boxes lifted up from U_{d-1}. Thus $X_d(n) \geq X_{d-1}(n) + 1$.

Consequently, the lower bound $X_d(n) = \Omega(n^{\lfloor d/2 \rfloor})$ holds for any fixed $d \geq 3$, as required. The proof of Theorem 2 is now complete.

5 Algorithmic aspects

In this section we present an alternative approximation algorithm in $\mathbb{R}^3$ and a possible approach towards a faster exact algorithm, also in $\mathbb{R}^3$.

5.1 An alternative $(1-\varepsilon)$-approximation algorithm in $\mathbb{R}^3$

Given n points in $U_d = [0,1]^3$ let $B_0 = \Pi_{i=1}^3 [a_i, b_i]$ be an (unknown) maximum empty box, and let v_0 denote its volume. Since $v_0 \geq 1/(n+1)$, the maximum extent of B_0 is roughly at least $n^{-1/3}$. Guess (by repeating over the three axis-directions) that the maximum extent of B_0 is along the z-coordinate. Consider the subdivision of B_0 into horizontal slabs of height about $0.5\,\varepsilon\, n^{-1/3}$.

Repeat over the $O(\varepsilon^{-2} n^{2/3})$ pairs of horizontal planes as top and bottom planes: project all the points in the corresponding slab onto a horizontal plane and use the planar exact algorithm by Aggarwal and Suri [4] to find a maximum empty rectangle in the projection plane in $O(n \log^2 n)$ time. Return the maximum volume of a box found in this way, i.e., as the product of the area of a maximum empty rectangle in the projection plane and the slab-extent along the z-coordinate. The relative error in the volume returned is at most ε and the resulting overall time is $O(\varepsilon^{-2}\, n^{2/3}\, n \log^2 n) = O(\varepsilon^{-2}\, n^{5/3} \log^2 n)$.

In comparison, the running time of the $(1-\varepsilon)$-approximation algorithm from [17] is $O(\varepsilon^{-6}\, n \log^3 n)$. With an ε^{-4} factor reduction at the expense of an $n^{2/3} \log^{-1} n$ factor increase in the running time, our alternative approximation algorithm offers an attractive trade-off for input instances with small values of ε and moderate values of n.

5.2 Towards a faster exact algorithm in $\mathbb{R}^3$ (conditional result)

Augustine et al. [7] and Kaplan et al. [27] studied the problem of finding the largest-area empty rectangle containing a query point. Kaplan et al. [27] showed how to construct in nearly linear time a data structure that takes nearly linear space, so that given a query point q, the largest-area empty rectangle containing q can be computed in $O(\log^4 n)$ time.

We next show that a semi-dynamic version of this data structure allows solving the Maximum Empty Box problem in 3-space in subcubic time.

Given n points in $[0,1]^3$, sort them by their z-coordinate; and set $z_0 = 0$, $z_{n+1} = 1$; hence we have $0 = z_0 \le z_1 \le \ldots \le z_n \le 1 = z_{n+1}$, where $p_i = (x_i, y_i, z_i)$, $i = 1, \ldots, n$. Assume for simplicity that all z-coordinates are distinct (the number of phases done by the algorithm is essentially the number of distinct value of z_i). For $0 \le i < j \le n+1$, let $\Gamma(i,j)$ be the planar data structure from [27] for the set of $j-i-1$ points $P_{ij} = \{q_k = (x_k, y_k, 0), k = i+1, \ldots, j-1\}$. The points q_k are the projections of the points in the open slab bounded by $z = z_i$ and $z = z_j$ onto a horizontal plane. Note that for $j = i+1$, $P_{ij} = \emptyset$ (so these slabs are trivial to deal with).

The algorithm proceeds in n phases, numbered from 0 to n. Phase i finds the maximum volume of an empty box whose z-extent is $[z_i, z_j]$, over all $j = n+1, \ldots, i+1$. To do this, it first computes $\Gamma(i, n+1)$, and then by repeated updates obtains $\Gamma(i,n), \Gamma(i,n-1), \ldots, \Gamma(i,i+2), \Gamma(i,i+1)$.

The maximum volume of an empty box whose z-extent is $[z_i, z_j]$, where $i < j$, equals the area of a maximum empty rectangle for the planar set $P_{ij} \subset U$ times the height $z_j - z_i$. In each subsequent step of Phase i, the height of current slab is reduced vertically by removing its top point. The corresponding planar data structure $\Gamma(i,j)$ is updated with respect to the deletion of one point, say, in time at most $U(n)$. There are about $n - i$ updates (and point deletions) in Phase i. The data structure is queried for the maximum empty rectangle containing the planar (projection) point that was just deleted. The area of this rectangle is then multiplied with the height of the current new slab to obtain the volume of a maximum empty box in the new slab and then the process is repeated. The algorithm outputs the volume of a maximum empty box over all $\binom{n+2}{2}$ slabs.

The data structure is computed from scratch n times, once for each phase, and there are at most $O(n^2)$ deletions, each taking $U(n)$ time to process. Alternatively one can compute that data structure only once, i.e., $\Gamma(0, n+1)$, but work with two copies, one being updated by deleting one point for each phase, and one being updated at each point deletion within a phase.

Suppose now that the above data structure can be made semi-dynamic with respect to point deletions: given a sequence of $m \le n$ points in $[0,1]^2$, construct the data structure and delete the points one by one in the given order, while updating the data structure after each deletion. Assume that this can be done in $U(n) = O(n^\gamma)$ (amortized) time per point deletion, where $\gamma < 1$. Recall that the data structure can be built from scratch in $T(n) = n \log^{O(1)} n$ time. Consequently, if $U(n) = O(n^\gamma)$, the overall running time for computing an empty box of maximum volume amidst n points in $[0,1]^3$ becomes $T(n) = O(n^{2+\gamma})$, as desired.

Perhaps obtaining a dependence of the form $U(n) = O(\log^{O(1)} n)$ is too optimistic, due to the fact that computing a maximum empty box is *non-decomposable*; see [16]; a different variant of largest empty box is considered there, however, the conclusion remains the same.

6 Open problems

We conclude with some open problems.

▶ **Problem 8.** *Can a maximum empty box in $\mathbb{R}^d$ for some fixed $d \geq 3$ be computed in $O(n^{\gamma_d})$ time for some constant $\gamma_d < d$?*

A possible approach in $\mathbb{R}^3$ is suggested by the following.

▶ **Problem 9.** *Is there a dynamic or semi-dynamic version – with respect to the deletion of points – of the data structure of Kaplan et al.* [27] *for finding the largest-area empty rectangle containing a query point?*

▶ **Problem 10.** *Is there an efficient algorithm for enumeration (or counting) of all maximum empty boxes amist n points in U_d? Is there one for the planar version running in $O(n \log^{O(1)} n)$ time?*

While we showed that the maximum number of rectangles of maximum area is linear or nearly linear in the plane, a significantly larger gap, between $\Omega(n^{\lfloor d/2 \rfloor})$ and $O(n^d)$, remains in higher dimensions. Moreover, the rôle of the parity of d remains unclear. We suspect that the $\log n$ factor in the planar upper bound is only an artifact of our proof. These remarks suggests the following.

▶ **Problem 11.** *Can an upper bound $X_2(n) = O(n\, 2^{\alpha(n)})$ be deduced? Is $X_2(n)$ superlinear in n?*

▶ **Problem 12.** *For fixed $d \geq 2$, what is the order of growth of $X_d(n)$? In particular, is $X_3(n) = \omega(X_2(n))$? Is $X_d(n) = \omega(X_{d-1}(n))$ for odd d?*

References

1 P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds for the length of general Davenport-Schinzel sequences. *J. Combin. Theory, Ser. A*, 52:228–274, 1989.

2 A. Aggarwal, M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.

3 A. Aggarwal and M. Klawe. Applications of generalized matrix searching to geometric algorithms. *Discrete Appl. Math.*, 27:3–23, 1990.

4 A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In *Proc. 3rd Ann. Sympos. on Comput. Geom.*, pp. 278–290, 1987.

5 C. Aistleitner, A. Hinrichs, and D. Rudolf. On the size of the largest empty box amidst a point set. Preprint, `http://arxiv.org/abs/1507.02067v1`, 2015.

6 M. J. Atallah and G. N. Frederickson. A note on finding a maximum empty rectangle. *Discrete Appl. Math.*, 13(1):87–91, 1986.

7 J. Augustine, S. Das, A. Maheshwari, S. C. Nandy, S. Roy, and S. Sarvattomananda. Querying for the largest empty geometric object in a desired location. Preprint, `http://arxiv.org/abs/1004.0558v2`, 2010.

8 J. Backer and M. Keil. The bichromatic rectangle problem in high dimensions. In *Proc. 21st Canadian Conf. on Comput. Geom.*, pp. 157–160, 2009.

9 J. Backer and M. Keil. The mono- and bichromatic empty rectangle and square problems in all dimensions. In *Proc. 9th Latin American Sympos. on Theor. Informatics*, pp. 14–25, 2010.

10 P. Braß, G. Rote, and K. J. Swanepoel. Triangles of extremal area or perimeter in a finite planar point set. *Discrete Comput. Geom.*, 26:51–58, 2001.

11 B. Chazelle, R. Drysdale, and D. T. Lee. Computing the largest empty rectangle. *SIAM J. Comput.*, 15:300–315, 1986.

12 J. G. van der Corput. Verteilungsfunktionen I. *Proc. Nederl. Akad. Wetensch.*, 38:813–821, 1935.

13 J. G. van der Corput. Verteilungsfunktionen II. *Proc. Nederl. Akad. Wetensch.*, 38:1058–1066, 1935.

14 A. Datta and S. Soundaralakshmi. An efficient algorithm for computing the maximum empty rectangle in three dimensions. *Inform. Sci.*, 128:43–65, 2000.

15 H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *Amer. J. Math.*, 87:684–694, 1965.

16 D. Dobkin and S. Suri. Maintenance of geometric extrema. *J. of ACM*, 38(2):275–298, 1991.

17 A. Dumitrescu and M. Jiang. On the largest empty axis-parallel box amidst n points. *Algorithmica*, 66(2):225–248, 2013.

18 A. Dumitrescu and M. Jiang. Maximal empty boxes amidst random points. *Combin. Probab. Comput.*, 22:477–498, 2013.

19 A. Dumitrescu and M. Jiang. Computational Geometry Column 60. *SIGACT News Bulletin*, 45(4):76–82, 2014.

20 A. Dumitrescu and M. Jiang. Perfect vector sets, properly overlapping partitions, and largest empty box. Manuscript, 2016.

21 J. Edmonds, J. Gryz, D. Liang, and R. Miller. Mining for empty spaces in large data sets. *Theoret. Comp. Sci.*, 296(3):435–452, 2003.

22 P. Erdős and G. Purdy. Some extremal problems in geometry. *J. Combin. Theory, Ser. A*, 10:246–252, 1971.

23 P. Giannopoulos, C. Knauer, M. Wahlström, and D. Werner. Hardness of discrepancy computation and ε-net verification in high dimension. *J. Complexity*, 28:162–176, 2012.

24 J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.*, 2:84–90, 1960.

25 J. M. Hammersley. Monte Carlo methods for solving multivariable problems. *Ann. New York Acad. Sci.*, 86: 844–874, 1960.

26 S. Hart and M. Sharir. Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes. *Combinatorica*, 6:151–177, 1986.

27 H. Kaplan, S. Mozes, Y. Nussbaum, and M. Sharir. Submatrix maximum queries in Monge matrices and Monge partial matrices, and their applications. In *Proc. 23rd ACM-SIAM Sympos. on Discrete Algorithms*, pp. 338–355, 2012.

28 H. Kaplan, N. Rubin, M. Sharir, and E. Verbin. Efficient colored orthogonal range counting. *SIAM J. Comput.*, 38:982–1011, 2008.

29 J. Matoušek. *Geometric Discrepancy: An Illustrated Guide*. Springer, 1999.

30 J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

31 A. Namaad, D. T. Lee, and W.-L. Hsu. On the maximum empty rectangle problem. *Discrete Appl. Math.*, 8:267–277, 1984.

32 S. Nandy and B. Bhattacharya. Maximal empty cuboids among points and blocks. *Computers Math. Applic.*, 36(3):11–20, 1998.

33 F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer, New York, 1985.

34 G. Rote and R. F. Tichy. Quasi-Monte-Carlo methods and the dispersion of point sequences. *Math. Comput. Modelling*, 23:9–23, 1996.

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

Strongly Monotone Drawings of Planar Graphs*

Stefan Felsner[1], Alexander Igamberdiev[2], Philipp Kindermann[†3], Boris Klemz[4], Tamara Mchedlidze[5], and Manfred Scheucher[‡6]

1 **Institut für Mathematik, Technische Universität Berlin, Berlin, Germany**
2 **LG Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany**
3 **LG Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany**
4 **Institute of Computer Science, Freie Universität Berlin, Berlin, Germany**
5 **Institute of Theoretical Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany**
6 **Institute of Software Technology, Graz University of Technology, Graz, Austria**

Abstract

A straight-line drawing of a graph is a *monotone drawing* if for each pair of vertices there is a path which is monotonically increasing in some direction, and it is called a *strongly monotone drawing* if the direction of monotonicity is given by the direction of the line segment connecting the two vertices.

We present algorithms to compute crossing-free strongly monotone drawings for some classes of planar graphs; namely, 3-connected planar graphs, outerplanar graphs, and 2-trees. The drawings of 3-connected planar graphs are based on primal-dual circle packings. Our drawings of outerplanar graphs depend on a new algorithm that constructs strongly monotone drawings of trees which are also convex. For irreducible trees, these drawings are strictly convex.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, G.2.2 Graph Theory

Keywords and phrases graph drawing, planar graphs, strongly monotone, strictly convex, primal-dual circle packing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.37

1 Introduction

Finding a path between a source vertex and a target vertex is one of the most important tasks when data are given by a graph, c.f. Lee et al. [16]. This task may serve as criterion for rating the quality of a drawing of a graph. Consequently researchers addressed the question of how to visualize a graph such that finding a path between any pair of nodes is easy. A user study of Huang et al. [13] showed that, in performing path-finding tasks, the eyes follow edges that go in the direction of the target vertex. This empirical study triggered the research topic of finding drawings with presence of some kind of geodesic paths. Several formalizations for the notion of geodesic paths have been proposed, most notably the notion of strongly monotone paths. Related drawing requirements are studied under the titles of self-approaching drawings and greedy drawings [1, 20].

* This research was initiated during the Geometric Graphs Workshop Week (GGWeek'15) at the FU Berlin in September 2015.
† Work by P. Kindermann was supported by DFG grant SC2458/4-1.
‡ Work by M. Scheucher was partially supported by the ESF EUROCORES programme EuroGIGA – CRP ComPoSe, Austrian Science Fund (FWF): I648-N18 and FWF project P23629-N18 'Combinatorial Problems on Geometric Graphs'.

© Stefan Felsner, Alexander Igamberdiev, Philipp Kindermann, Boris Klemz, Tamara Mchedlidze, and Manfred Scheucher;
licensed under Creative Commons License CC-BY

Let $G = (V, E)$ be a graph. We say that a path P is *monotone with respect to* a direction (or vector) d if the orthogonal projections of the vertices of P on a line with direction d appear in the same order as in P. A straight-line drawing of G is called *monotone* if for each pair of vertices $u, v \in V$ there is a connecting path that is monotone with respect to some direction. To support the path-finding tasks it is useful to restrict the monotone direction for each path to the direction of the line segment connecting the source and the target vertex: a path $v_1 v_2 \dots v_k$ is called *strongly monotone* if it is monotone with respect to the vector $\overrightarrow{v_1 v_k}$. A straight-line drawing of G is called *strongly monotone* if each pair of vertices $u, v \in V$ is connected by a strongly monotone path.

If crossings are allowed, then any strongly monotone drawing of a spanning tree of G yields a strongly monotone drawing of G, this has been observed by Angelini et al. [2]. For this reason, strongly monotone drawings referred to in this paper are always crossing-free.

Related Work

In addition to (strongly) monotone drawings, there are several other drawing styles that support the path-finding task. The earliest studied is the concept of *greedy drawings*, introduced by Rao et al. [20]. In a greedy drawing, one can find a source–target path by iteratively selecting a neighbor that is closer to the target. Triangulations admit crossing free greedy drawings [7], and more generally 3-connected planar graphs have greedy drawings [17]. Trees with a vertex of degree at least 6 have no greedy drawing. Nöllenburg and Prutkin [18] gave a complete characterization of trees that admit a greedy drawing.

Greedy drawings can have some undesirable properties, e.g., a greedy path can look like a spiral around the target vertex. To get rid of this effect, Alamdari et al. [1] introduced a subclass of greedy drawings, so-called *self-approaching drawings* which require the existence of a source–target path such that for any point p on the path the distance to another point q is decreasing along the path. In greedy drawings this is only required for q being the target-vertex. These drawings are related to the concept of self-approaching curves [14]. Alamdari et al. provide a complete characterization of trees that admit a self-approaching drawing.

Even more restricted are *increasing-chord drawings*, which require that there always is a source–target path which is self-approaching in both directions. Nöllenburg et al. [19] proved that every triangulation has a (not necessarily planar) increasing-chord drawing and every planar 3-tree admits a planar increasing-chord drawing. Dehkordi et al. [6] studied the problem of connecting a given point set in the plane with an increasing-chord graph.

Monotone drawings were introduced by Angelini et al. [2] They showed that any n-vertex tree admits a monotone drawing on a grid of size $O(n^{1.6}) \times O(n^{1.6})$ or $O(n) \times O(n^2)$. They also showed that any 2-connected planar graph has a monotone drawing having exponential area. Kindermann et al. [15] improved the area bound for trees to $O(n^{1.5}) \times O(n^{1.5})$ even with the property that the drawings are convex. The area bound was further lowered to $O(n^{1.205}) \times O(n^{1.205})$ by He and He [10]. Recently, the same authors [9] further reduced the area bound to $O(n \log n) \times O(n \log n)$.

Hossain and Rahman [12] showed that every connected planar graph admits a monotone drawing on a grid of size $O(n) \times O(n^2)$. For 3-connected planar graphs, He and He [11] proved that the convex drawings on a grid of size $O(n) \times O(n)$, produced by the algorithm of Felsner [8], are monotone. For the fixed embedding setting, Angelini et al. [3] showed that every plane graph admits a monotone drawing with at most two bends per edge, and all 2-connected plane graphs and all outerplane graphs admit a straight-line monotone drawing.

Angelini et al. [2] also introduced the concept of *strong monotonicity* and gave an example of a drawing of a planar triangulation that is not strongly monotone. Kindermann et al. [15]

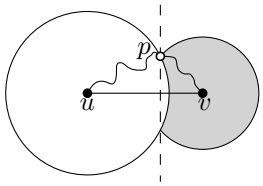


Figure 1 Any increasing-chord path is also strongly monotone.

showed that every tree admits a strongly monotone drawing. However, their drawing is not necessarily strictly convex and requires more than exponential area. Further, they presented an infinite class of 1-connected graphs that do not admit strongly monotone drawings. Nöllenburg et al. [19] have recently shown that exponential area is required for strongly monotone drawings of trees and binary cacti.

There are some relations among the aforementioned drawing styles. Plane increasing-chord drawings are self-approaching by definition, but they are also strongly monotone: consider an increasing-chord path from u to v and any point p on this path. By definition, by walking along this path, the distance to u increases and the distance to v decreases; hence, any point q on the walk from p to v lies outside the circle around u through p but inside the circle around v through p. Since these points all lie on the same side of the line through p with direction $\overrightarrow{uv}^{\perp}$ as v, the path is strongly monotone; see Figure 1. Self-approaching drawings are greedy by definition. On the other hand, (plane) self-approaching drawings are not necessarily monotone, and vice-versa.

Our Contribution

After giving some basic definitions used throughout the paper in Section 2, we present four results. First, we show that any 3-connected planar graph admits a strongly monotone drawing induced by primal-dual circle packings (Section 3). Then, we answer in the affirmative the open question of Kindermann et al. [15] on whether every tree has a strongly monotone drawing which is strictly convex. We use this result to show that every outerplanar graph admits a strongly monotone drawing (Section 4). Finally, we prove that 2-trees can be drawn strongly monotone (Section 5). All our proofs are constructive and admit efficient drawing algorithms. Our main open question is whether every planar 2-connected graph admits a strongly monotone drawing (Section 6). It would also be interesting to understand which graphs admit strongly monotone drawings on a grid of polynomial size.

2 Definitions

Let $G = (V, E)$ be a graph. A *drawing* Γ of G maps the vertices of G to distinct points in the plane and the edges of G to simple Jordan curves between their end-points. In a *straight-line* drawing, each edge is mapped to a straight line segment. A planar drawing induces a *combinatorial embedding* which is the class of topologically equivalent drawings. In particular, an embedding specifies the connected regions of the plane, called *faces*, whose boundary consists of a cyclic sequence of edges. The unbounded face is called the *outer face*, the other faces are called *internal faces*. For connected graphs, an embedding can also be defined by a *rotation system*, that is, the circular order of the incident edges around each vertex. Note that both definitions are equivalent for planar graphs.

A (straight-line) drawing of a planar graph is a *convex drawing* if it is crossing free and internal faces are realized as convex non-overlapping (polygonal) regions. The *augmentation*

of a straight-line drawn tree is obtained by substituting each edge incident to a leaf by a ray which begins with the edge and extends across the leaf. A drawing of a tree is a *(strictly) convex drawing* if the augmented drawing is crossing free and has (strictly) convex faces, i.e., all the angles of the unbounded polygonal regions are less or equal to (strictly less than) π. Note that strict convexity forbids vertices of degree 2. We call a tree *irreducible* if it contains no vertices of degree 2. It has been observed before that a convex drawing of a tree is also monotone but a monotone drawing is not necessarily convex, see [2, 4].

A *k-tree* is a graph which can be produced from a complete graph K_{k+1} and then repeatedly adding vertices in such a way that the neighbors of the added vertex form a k-clique. We say that the new vertex is *stacked* on the clique. By construction k-trees are chordal graphs. They can also be characterized as maximal graphs with treewidth k, that is, no edges can be added without increasing the treewidth. Note that 1-trees are equivalent to trees and 2-trees are equivalent to maximal series-parallel graphs.

We denote an undirected edge between two vertices $a, b \in V$ by (a, b). In a drawing of G, we may identify each vertex with the point in the plane it is mapped to. For two vectors x and y, we define the angle $\measuredangle(x, y)$ as the smallest angle between the two vectors, that is, $\measuredangle(x, y) = \arccos\left(\frac{\langle x,y\rangle}{|x||y|}\right)$, and for three points p, q, r, we define $\measuredangle pqr = \measuredangle(\overrightarrow{qp}, \overrightarrow{qr})$. We say that a vector x is monotone with respect to y if $\measuredangle(x, y) < \pi/2$. This yields an alternative definition of a strongly monotone path: A path $v_1 v_2 \dots v_k$ is strongly monotone if $\measuredangle(\overrightarrow{v_i v_{i+1}}, \overrightarrow{v_1 v_k}) < \pi/2$, for $1 \le i \le k-1$. Note that we interpret monotonicity as strict monotonicity, i.e., we do not allow edges on the path that are orthogonal to the segment between the endpoints.

3 3-Connected Planar Graphs

In this section, we prove the following theorem.

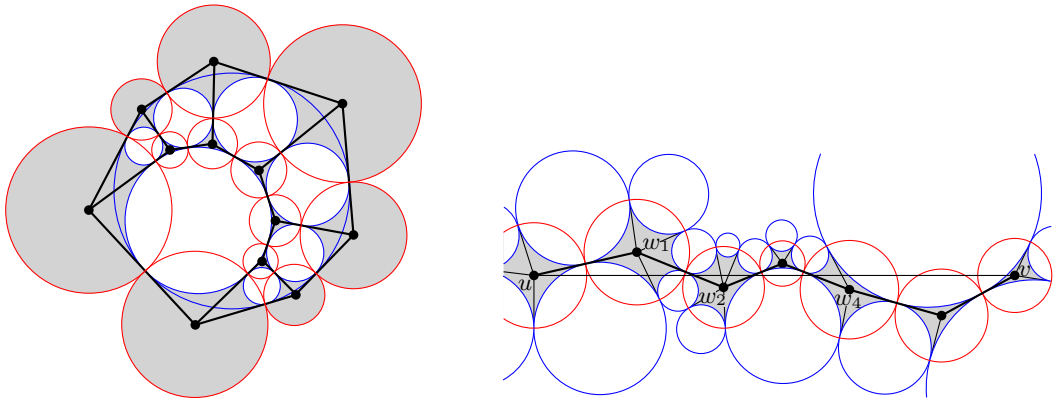
▶ **Theorem 1.** *Every 3-connected planar graph has a strongly monotone drawing.*

Proof. We show that the straight-line drawing corresponding to a primal-dual circle packing of a graph G is already strongly monotone. The theorem then follows from the fact that any 3-connected planar graph $G = (V, E)$ admits a primal-dual circle packing. This was shown by Brightwell and Scheinerman [5]; for a comprehensive treatment of circle packings we refer to Stephenson's book [21].

A *primal-dual circle packing* of a plane graph G consists of two families $\mathcal{C}_V$ and $\mathcal{C}_F$ of circles such that, there is a bijection $v \leftrightarrow C_v$ between the set V of vertices of G and circles of $\mathcal{C}_V$ and a bijection $f \leftrightarrow C_f$ between the set F of faces of G and circles of $\mathcal{C}_F$. Moreover, the following properties hold:

1. The circles in the family $\mathcal{C}_V$ are interiorly disjoint and their contact graph is G, i.e., $C_u \cap C_v \neq \emptyset$ if and only if $(u, v) \in E(G)$.
2. If $C_o \in \mathcal{C}_F$ is the circle of the outer face o, then the circles of $\mathcal{C}_F \setminus \{C_o\}$ are interiorly disjoint while C_o contains all of them. The contact graph of $\mathcal{C}_F$ is the dual G^* of G, i.e., $C_f \cap C_g \neq \emptyset$ if and only if $(f, g) \in E(G^*)$.
3. The circle packings C_V and C_F are orthogonal, i.e., if $e = (u, v)$ and the dual of e is $e^* = (f, g)$, then there is a point $p_e = C_u \cap C_v = C_f \cap C_g$; moreover, the common tangents t_{e^*} of C_u, C_v and t_e of C_f, C_g cross perpendicularly in p_e.

Let a primal-dual circle packing of a graph G be given. For each vertex v, let p_v be the center of the corresponding circle C_v. By placing each vertex v at p_v, we obtain a planar straight-line drawing Γ of G. In this drawing, the edge $e = (u, v)$ is represented by the



(a) (b)

Figure 2 (a) Drawing Γ of 3-connected graph $G = (V, E)$. Red circles are vertex circles $\mathcal{C}_V$, Blue circles are face circles $\mathcal{C}_F$. Regions of faces in white, regions of vertices in gray. (b) A strongly monotone path (thick edges) from u to v.

segment with end-points p_u and p_v on t_e. The face circles are inscribed circles of the faces of Γ; moreover, C_f is touching each boundary edge of the face f; see Figure 2a.

A straight-line drawing Γ^* of the dual G^* of G with the dual vertex of the outer face o at infinity can be obtained similarly by placing the dual vertex of each bounded face f at the center of the corresponding circle C_f. In this drawing, a dual edge $e^* = (f, o)$ is represented by the ray supported by t_{e^*} that starts at p_f and contains p_e.

In the following, we will make use of a specific partition Π of the plane. The regions of Π correspond to the vertices and the faces of G. For a vertex or face x, let D_x be the interior disk of C_x.

- The region R_f of a bounded face f is D_f.
- The region R_v of a vertex v is obtained from the disk D_v by removing the intersections with the disks of bounded faces, i.e., $R_v = D_v \setminus \bigcup_{f \neq o} R_f = D_v \setminus \bigcup_{f \neq o} D_f$; see Figure 2a.

To get a partition of the whole plane, we assign the complement of the already defined regions to the outer face, i.e, $R_o = \mathbb{R}^2 \setminus (\bigcup_{f \neq o} R_f \cup \bigcup_v R_v) = \mathbb{R}^2 \setminus (\bigcup_{f \neq o} D_f \cup \bigcup_v D_v)$.

Note that the edge-points p_e are part of the boundary of four regions of Π and if two regions of Π share more than one point on the boundary, then one of them is a vertex region R_v, the other is a face-region D_f, and (v, f) is an incident pair of G.

We are now prepared to prove the strong monotonicity of Γ. Consider two vertices u and v and let ℓ be the line spanned by p_u and p_v. W.l.o.g., assume that ℓ is horizontal and p_u lies left of p_v. Let ℓ_s be the directed segment from p_u to p_v. Since $p_u \in R_u$ and $p_v \in R_v$, the segment ℓ_s starts and ends in these regions. In between, the segment will traverse some other regions of Π. This is true unless (u, v) is an edge of G whence the strong monotonicity for the pair is trivial. We assume non-degeneracy in the following sense.
Non-degeneracy: The interior of the segment ℓ_s contains no vertex-point p_w, edge-point p_e, or face-point p_f.

Möbius transformations of the plane map circle packings to circle packings. In fact, the primal-dual circle packing of G is unique up to Möbius transformation; see Stephenson [21]. Now, any degenerate primal-dual circle packing of G can be mapped to a non-degenerate one by a Möbius transformation. This justifies the non-degeneracy assumption. Later we will give a more direct handling of degenerate situations.

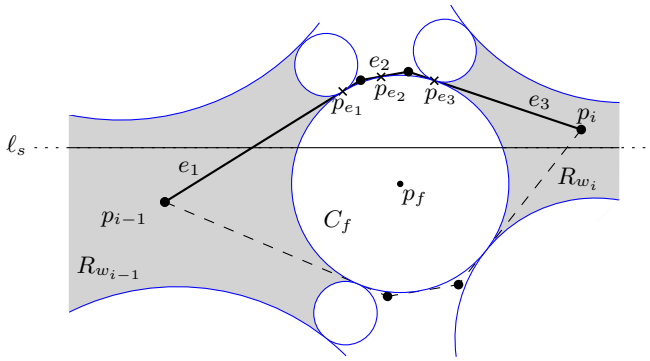


Figure 3 The path P_i connecting p_{i-1} and p_i.

Let $u = w_0, w_1, \ldots, w_k = v$ be the sequence of vertices whose region is intersected by ℓ_s, in the order of intersection from left to right, and let $p_i = p_{w_i}$; see Figure 2b. We will construct a strongly monotone path P from p_u to p_v in Γ that contains $p_u = p_0, p_1, \ldots, p_k = p_v$ in this order. Let P_i be the subpath of P from p_{i-1} to p_i. Since ℓ_s may revisit a vertex-region, it is possible that $p_{i-1} = p_i$; in this case we set $P_i = p_i$. Now suppose that $p_{i-1} \neq p_i$. Non-degeneracy implies that the segment ℓ_s alternates between vertex-regions and face-regions; hence, a unique disk D_f is intersected by ℓ_s between the regions of w_{i-1} and w_i. It follows that w_{i-1} and w_i are vertices on the boundary of f. The boundary of f contains two paths from w_{i-1} to w_i. In Γ, one of these two paths from p_{i-1} to p_i lies above D_f, we call it the *upper path*; the other one lies below D_f, we call it the *lower path*. If the center p_f of D_f lies below ℓ, then we choose the upper path from p_{i-1} to p_i as P_i; otherwise, we choose the lower path.

Suppose that this rule led to the choice of the upper path; see Figure 3. The case that the lower path was chosen works analogously. We have to show that P_i is monotone with respect to ℓ, i.e., to the x-axis. Let $e_1, \ldots, e_r$ be the edges of this path and let $e_j = (q_{j-1}, q_j)$; in particular, $q_0 = p_{i-1}$ and $q_r = p_i$. Since $R_{w_{i-1}}$ is star-shaped with center p_{i-1}, the segment connecting p_{i-1} with the first intersection point of ℓ with C_f belongs to $R_{w_{i-1}}$. Therefore, the point p_{e_1} of tangency of edge e_1 at C_f lies above ℓ. Similarly, p_{e_r} and, hence, all the points p_{e_j} lie above ℓ. Since the points $p_{e_1}, \ldots, p_{e_r}$ appear in this order on C_f and the center of C_f lies below ℓ, we obtain that their x-coordinates are increasing in this order. This sequence is interleaved with the x-coordinates of $q_0, q_1, \ldots, q_r$; hence, this is also monotone. This proves that the chosen path P_i is monotone with respect to ℓ. Monotonicity also holds for the concatenation $P = P_1 + P_2 + \ldots + P_k$; see Figure 2b.

We have shown strong monotonicity under the non-degeneracy assumption. Next, we consider degenerate cases and show how to find strongly monotone paths in these cases.

If ℓ_s contains a vertex-point p_w with $w \neq u, v$, the path P between u and v is just the concatenation of monotone paths between the pairs u, w and w, v; hence, it is strongly monotone. Next suppose that ℓ_s contains an edge-point p_e. If the edge e in Γ is horizontal, then we also have two vertex-points on ℓ_s and are in the case described above; otherwise, we consider the region which is touching ℓ from above as intersecting and the region which is touching ℓ from below as non-intersecting. This recovers the property that there is an alternation between vertex-regions and face-regions intersected by ℓ_s. Hence, the definition of the path for u and v gives a strongly monotone path unless it contains a vertical edge. The use of a vertical edge can be excluded by properly adjusting degeneracies of the form $p_f \in \ell$. For faces f with $p_f \in \ell$, we use the upper path, i.e., we consider p_f to be below ℓ. Thus,

even in degenerate situations the drawing corresponding to a primal-dual circle packing is strongly monotone. This concludes the proof. ◀

4 Trees and Outerplanar Graphs

Kindermann et al. [15] have shown that any tree has a strongly monotone drawing and that any irreducible binary tree has a strictly convex strongly monotone drawing. They left as an open question whether every tree admits a convex strongly monotone drawing; noticing that, in the positive case, this would imply that every Halin graph has a convex strongly monotone drawing.

In this section, we show that every tree has a convex strongly monotone drawing. Moreover, if the tree is irreducible, then the drawing is strictly convex. We use the result on trees to prove that every outerplanar graphs admits a strongly monotone drawing.

▶ **Theorem 2.** *Every tree has a convex strongly monotone drawing. If the tree is irreducible, then the drawing is strictly convex.*

Proof. We actually prove something stronger, namely, that any tree T has a drawing Γ with the following properties:

I1. Every leaf of T is placed on a corner of the convex hull of the vertices in Γ.

I2. If $a_1, \ldots, a_\ell$ is the counterclockwise order of the leaves on the convex hull, then for $i = 1, \ldots, \ell$ the vectors $(\overrightarrow{a_i a_{i-1}})^\perp$, $\overrightarrow{p_i a_i}$, $(\overrightarrow{a_{i+1} a_i})^\perp$ appear in counterclockwise radial order, where p_i denotes the unique vertex adjacent to a_i (see Figure 4a).

I3. The angle between two consecutive edges incident to a vertex $v \in V(T)$ is at most π and is equal to π only when v has degree two.

I4. Γ is strongly monotone.

Let T be a tree on at least 3 vertices, rooted at some vertex v_0 with degree at least 2. We inductively produce a drawing of T. We begin with placing the root v_0 at any point in the plane and the children $u_1, \ldots, u_k$ of v_0 at the corners of a regular k-gon with center v_0. The resulting drawing clearly fulfills the four desired properties.

Let T^- be a subtree of T such that T^- has at least one leaf a_i that is not a leaf in T. Let Γ^- be a drawing of T^- that fulfills the properties I1–I4. Let $u_1, \ldots, u_k$ be the children of a_i in T. Let T^+ denote the subtree of T induced by $V(T^-) \cup \{u_1, \ldots, u_k\}$. In the inductive step, we explain how to extend the drawing Γ^- of T^- to a drawing Γ^+ of T^+ such that it fulfills the properties I1–I4.

We first define a region R which is appropriate for the placement of $u_1, \ldots, u_k$; see Figure 4a for an illustration. Let $C_i(a_i)$ be the open cone containing all points x such that the vectors $(\overrightarrow{a_i a_{i-1}})^\perp$, $\overrightarrow{a_i x}$, and $(\overrightarrow{a_{i+1} a_i})^\perp$ are ordered counterclockwise. From property I2, it follows that $C_i(a_i)$ contains the *prolongation* h_{p_i,a_i} of $\overrightarrow{p_i a_i}$, i.e., the ray that starts with $\overrightarrow{p_i a_i}$ and extends across a_i. For every vertex $y \neq a_i$ of T^-, let $C_i(y)$ be the open cone consisting of all points p such that the path from y to a_i in T^- is monotone with respect to $\overrightarrow{yp}$. Since the drawing Γ^- is strongly monotone in a strict sense, $C_i(y)$ contains an open disk centered at a_i. Therefore, the intersection $\cap_{y \in V(T^-) \setminus \{a_i\}} C_i(y)$ of the all these cones contains an open disk D centered at a_i. We define the region R to be the intersection of all cones $C_i(y)$ and the cone $C_i(a_i)$, i.e., $R = \cap_{v \in V(T^-)} C_i(v)$. The intersection of disk D with $C_i(a_i)$ yields an open 'pizza slice' contained in R. In particular, R is non-empty.

Since R is an open convex set, we can construct a circular arc A in R with center a_i that contains points on both sides of the prolongation h_{p_i,a_i} of $\overrightarrow{p_i a_i}$; see Figure 4b. We place the vertices $u_1, \ldots, u_k$ on the arc A between the intersection of the tangent to A

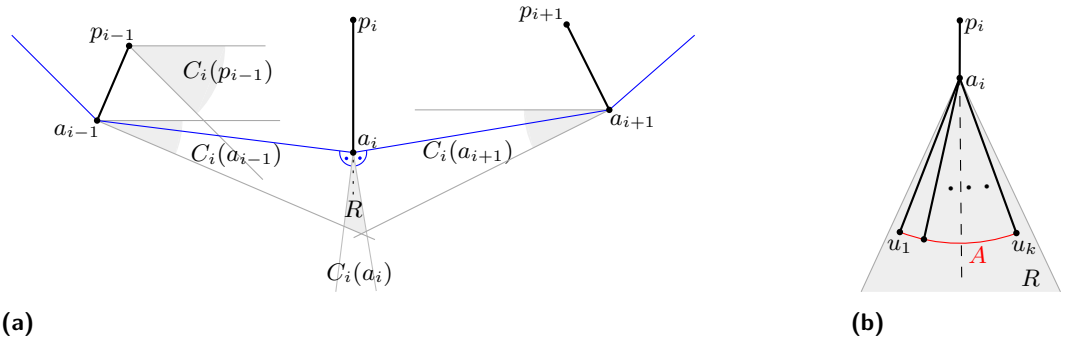


Figure 4 (a) The region R which is used for placing all the children of vertex a_i. The boundary of the convex hull is drawn blue. (b) Placement of the children $u_1, \ldots, u_k$ on the arc $A \subset R$. The prolongation h_{p_i,a_i} is drawn dashed, the arc A is drawn red.

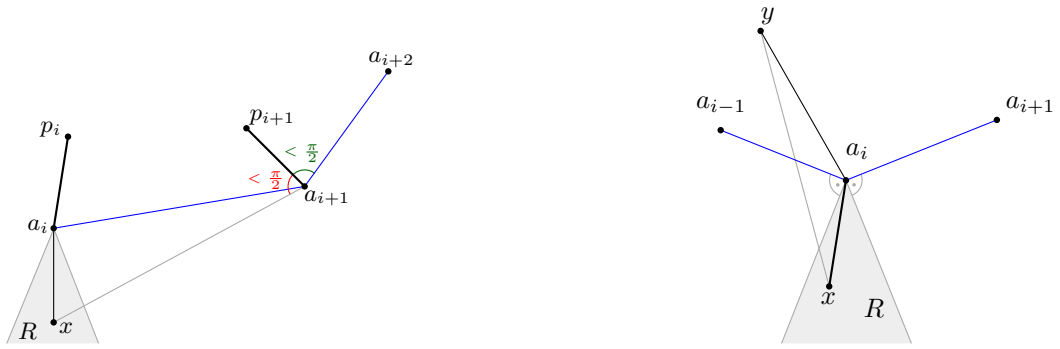


Figure 5 (a) An illustration for the proof of property I1 and property I2. (b) An illustration of the case where $y \in V(T^-)$ and $x \in \{u_1, \ldots, u_k\}$.

through a_{i-1} (if it exists) and the intersection of the tangent to A through a_{i+1} (if it exists) such that $\measuredangle p_i a_i u_1 = \measuredangle u_k a_i p_i$. This placement implies that in case a_i has degree 2 (that is, $k = 1$), $\measuredangle p_i a_i u_1 = \measuredangle u_1 a_i p_i = \pi$, and otherwise all the angles $\measuredangle p_i a_i u_1$, $\measuredangle u_k a_i p_i$, $\measuredangle u_j a_i u_{j+1}$, for $j = 1, \ldots, k-1$, are all less than π. This ensures property I3.

Next, we prove that the drawing Γ^+ of T^+ fulfills property I1. We first show that a_{i-1} and a_{i+1} lie on the convex hull of Γ^+; see Figure 5a. Consider the path from a_{i+1} to a_i in T^-, and let x be a point in R. By definition of R, this path is monotone (in a strict sense) with respect to $\overrightarrow{a_i x}$; therefore, $\measuredangle p_{i+1} a_{i+1} x < \pi/2$. Considering the strictly monotone path from a_{i+2} to a_{i+1} in T^- we obtain that $\measuredangle a_{i+2} a_{i+1} p_{i+1} < \pi/2$. The two inequalities above sum up to $\measuredangle x a_{i+1} a_{i+2} < \pi$ which means that a_{i+1} lies on the convex hull of Γ^+. Analogously, we obtain that a_{i-1} lies on the convex hull of Γ^+.

Notice that at least one of $u_1, \ldots, u_k$ lies on the convex hull of Γ^+ since they are placed outside of the convex hull of Γ^-. On the other hand, the construction of the circular arc A and the placement between the intersection points of the tangents through a_{i-1} and a_{i+1} ensures that $\measuredangle u_2 u_1 a_{i-1} < \pi$ and $\measuredangle a_{i+1} u_k u_{k-1} < \pi$. Hence, all of them lie on the convex hull of Γ^+. This ensures property I1.

For property I2, observe that $\measuredangle x a_i a_{i+1} > \pi/2$ holds for every $x \in C_i(a_i)$ (see Figure 5a), and therefore $\measuredangle a_{i+1} x a_i < \pi/2$, as these two angles lie in the triangle $\triangle x a_i a_{i+1}$. The last inequality implies property I2 for Γ^+.

Finally, we show that property I4 holds, i.e., that Γ^+ is a strongly monotone drawing. Consider $x, y \in V(T^+)$, let P_{xy} denote the path between x and y in T^+. We distinguish the following three cases:

1. If $x, y \in V(T^-)$, then the path P_{xy} is contained in T^-. Since Γ^- is a strongly monotone drawing by induction hypothesis, P_{xy} is strongly monotone.
2. If $y \in V(T^-)$ and $x \in \{u_1, \ldots, u_k\}$, then $P_{yx} = P_{ya_i} + (a_i, x)$; refer to Figure 5b. The path P_{ya_i} is monotone with respect to $\overrightarrow{yx}$ by construction because $x \in A \subset R \subset C_i(x)$. The definition of R also implies that $\measuredangle a_{i-1}a_ix$ and $\measuredangle a_{i+1}a_ix$ are greater than $\pi/2$. Since y lies inside the convex hull of Γ^-, the smallest angle $\measuredangle ya_ix$ is also greater than $\pi/2$. Thus, $\measuredangle a_ixy < \pi/2$ which implies that the vector $\overrightarrow{xa_i}$ is monotone with respect to $\overrightarrow{xy}$. We conclude that P_{xy} is strongly monotone.
3. If $x, y \in \{u_1, \ldots, u_k\}$, then the path $P_{xy} = (x, a_i) + (a_i, y)$ is strongly monotone since x and y are placed on the circular arc A centered at a_i.

We have proven that each tree has a drawing that fulfills the four properties I1–I4. Property I2 implies that the prolongations of the edges incident to the leaves do not intersect. This, together with property I3, implies the convexity of the drawing and strict convexity in case of an irreducible tree. This concludes the proof of the theorem. ◀

▶ **Theorem 3.** *Every outerplanar graph has a convex strongly monotone drawing.*

Proof. Let G be an outerplanar graph with at least 2 vertices. For every vertex $v \in V$, we add two dummy vertices v', v'' and edges (v, v'), (v, v''). By construction, the resulting graph H is outerplanar and does not contain vertices of degree 2. Let Γ_H be an outerplanar drawing of H. We will construct a convex strongly monotone drawing Γ'_H of H with the same combinatorial embedding as Γ_H.

Let T be an arbitrary spanning tree of H. By construction, no vertex in T has degree 2. Thus, according to Theorem 2, T admits a strongly monotone drawing Γ_T which is strictly convex and which also preserves the order of the children for every vertex, i.e., the rotation system coincides with the one in Γ_H.

Now, we insert all the missing edges. Recall that, by removing an edge from a planar drawing, the two adjacent faces are merged. Since the drawing Γ_T of T is strictly convex and since Γ_T preserves the rotation system of Γ_H, by inserting an edge $e = (u, v)$ of the graph H into Γ_T one strictly convex face is partitioned into two strictly convex faces. Note that vertices u, v have to be incident to the same strictly convex face. Otherwise let C be the cycle contained in $E(T) \cup \{e\}$. If u and v are not incident to the same strictly convex face of Γ_T, there exists a leaf of T that is contained in the interior of C in Γ_H contradicting to the outerplanarity of H. Furthermore, the insertion of edge e does not destroy strong monotonicity. We re-insert all edges of H iteratively. The resulting drawing Γ'_H of H is a strictly convex and strongly monotone.

Finally, we remove all the dummy vertices and obtain a strongly monotone drawing of G. Since Γ'_H has the same combinatorial embedding as Γ_H, every dummy vertex lies in the outer face. Hence, no internal face is affected by the removal of dummy vertices, and thus all internal faces remain strictly convex. ◀

5 2-Trees

In this section, we show how to construct a strongly monotone drawing for any 2-tree. We begin by introducing some notation. A *drawing with bubbles* of a graph $G = (V, E)$ is a straight-line drawing of G in the plane such that, for some $E' \subseteq E$, every edge $e \in E'$ is

(a) (b)

Figure 6 (a) A drawing of a 2-tree with bubbles (orange) and (b) an extension of the drawing.

associated with a circular region in the plane, called a *bubble* B_e; see Figure 6a. An *extension* of a drawing with bubbles is a straight-line drawing that is obtained by taking some subset of edges with bubbles $E'' \subseteq E'$ and stacking one vertex on top of each edge $e \in E''$ into the corresponding bubble B_e; see Figure 6b. (Since every bubble is associated with a unique edge we often simply say that a vertex is stacked into a bubble without mentioning the corresponding edge.) We call a drawing with bubbles Γ strongly monotone if *every* extension of Γ is strongly monotone. Note that this implies that if a vertex w is stacked on top of edge e into bubble B_e, then there exists a strongly monotone path from w to any other vertex in the drawing and, furthermore, there exists a strongly monotone path from w to any of the current bubbles, i.e., to any vertex that might be stacked into another bubble.

Every 2-tree $T = (V, E)$ can be constructed through the following iterative procedure:

1. We start with one edge and tag it as *active*. During the entire procedure, every present edge is tagged either as active or *inactive*.
2. As an iterative step we pick one active edge e and stack vertices $w_1, \ldots, w_k$ on top of this edge for some $k \geq 0$ (we note that k might equal 0). Edge e is then tagged as inactive and all new edges incident to the stacked vertices $w_1, \ldots, w_k$ are tagged as active.
3. If there are active edges remaining, repeat Step 2.

Observe that Step 2 is performed exactly once per edge and that an according decomposition for T can always be found by the definition of 2-trees.

We construct a strongly monotone drawing of T by geometrically implementing the iterative procedure described above, so that after every step of the algorithm the present part of the graph is realized as a drawing with bubbles. We use the following additional geometrical condition:

(C) After each step of the algorithm every active edge comes with a bubble and the drawing with bubbles is strongly monotone. Additionaly, for an edge $e = (u, v)$ with bubble B_e for each point $w \in B_e$, the angle $\measuredangle(\overrightarrow{uw}, \overrightarrow{wv})$ is obtuse.

In Step 1, we arbitrarily draw the edge e_0 in the plane. Clearly, it is possible to define a bubble for e_0 that only allows obtuse angles. In Step 2, we place the vertices $w_1, \ldots, w_k$ over an edge $e = (u, v)$ as follows. The fact that stacking a vertex into B_e gives an obtuse angle allows us to place the to-be stacked vertices $w_1, \ldots w_k$ in B_e on a circular arc around u such that, for any $1 \leq i, j \leq k$, there exists a strongly monotone path between w_i and w_j; see Figure 7a. Due to condition 3, there also exists a strongly monotone path between any of the newly stacked vertices and any vertex of an extension of the previous drawing with bubbles. Hence, after removing the bubble B_e, the resulting drawing is a strongly monotone drawing with bubbles.

In order to maintain condition 3, it remains to describe how to define the bubbles for the new active edges incident to the stacked vertices. For this purpose, we state the following Lemma 4, which enables us to define the two bubbles for the edges incident to any degree-2 vertex with an obtuse angle. The Lemma is then iteratively applied to the vertices $w_1, \ldots, w_k$ and after every usage of the Lemma the produced drawing with bubbles is strongly monotone.

(a) Stacking vertices into a bubble **(b)** The empty neighbourhood $\mathcal{N}$ (dotted)

Figure 7 Illustrations for the drawing approach for strongly monotone 2-trees.

This iterative approach is used to ensure that, when defining bubbles for some vertex w_i, the previously added bubbles for $w_1, \ldots, w_{i-1}$ are taken into account.

▶ **Lemma 4.** *Let Γ be a strongly monotone drawing with bubbles and let w be a vertex of degree 2 with an obtuse angle such that the two incident edges $e_1 = (u, w)$ and $e_2 = (v, w)$ have no bubbles. Then, there exist bubbles B_{e_1} and B_{e_2} for edges e_1 and e_2 respectively that only allow obtuse angles such that Γ remains strongly monotone with bubbles if we add B_{e_1} and B_{e_2}.*

Proof. We begin by describing how we determine the size and location of the new bubbles. Since Γ is planar, there exists a neighborhood $\mathcal{N}$ of w, e_1 and e_2 that does not contain elements of any extension of Γ; see Figure 7b.

Furthermore, consider any extension Γ' of Γ. Since we consider monotonicity in a strict fashion, there exists a constant $\alpha > 0$ such that, for any pair of vertices s_0, s_t of Γ', there exists a strongly monotone path $P = (s_0, \ldots, s_t)$ with $\measuredangle(\overrightarrow{s_0 s_t}, \overrightarrow{s_i s_{i+1}}) \leq \pi/2 - \alpha$ for $i = 0, \ldots, t-1$. We refer to this property of P as being *α-safe* with respect to $\overrightarrow{s_o s_t}$. A simple compactness argument shows that this safety parameter can be chosen simultaneously for all the extensions of Γ: there exists $\alpha(\Gamma) > 0$ such that for every extension Γ' of Γ for every two vertices s_0, s_t of Γ' there exists a strongly monotone path connecting these vertices that is $\alpha(\Gamma)$-safe with respect to $\overrightarrow{s_o s_t}$. More precisely, for a fixed extension Γ', let $\alpha(\Gamma')$ denote the *maximal* α such that for any pair of vertices s_0, s_t of Γ' there exists a strongly monotone path connecting them which is α-safe with respect to $\overrightarrow{s_o s_t}$. Then we can chose the global safety parameter as $\alpha(\Gamma) := \inf_{\Gamma'} \alpha(\Gamma')$, where the infimum is taken over all the extensions Γ' of Γ. This infimum is strictly positive since the set of extensions is compact and the function $\alpha(\Gamma')$ is continuous and strictly positive.

For the edge e_1, we define the bubble B_{e_1} as the circle of radius r with center at the extension of the edge e_2 over w with distance ε to w as depicted in Figure 8a. In order to ensure the strong monotonicity, we choose r and ε such that the following properties hold (these properties clearly hold as soon as r, ε and r/ε are small enough):

- **(i)** Bubble B_{e_1} is located inside the empty neighborhood $\mathcal{N}$. Moreover, to preserve obtusity, B_{e_1} needs to lie inside the semicircle with edge e_1 as diameter, as depicted in Figure 8a.
- **(ii)** Consider angles β_1 and β_2 as illustrated in Figure 8b. We require that both angles are smaller than $\alpha(\Gamma)/4$.
- **(iii)** For any vertex y of any extension of Γ, consider the angle β_y as illustrated in Figure 8c. We require that this angle is smaller than $\alpha(\Gamma)/4$. That guarantees that for any point $x \in B_{e_1}$ it holds that $\measuredangle(\overrightarrow{yw}, \overrightarrow{yx}) < \alpha(\Gamma)/4$.

We define the bubble B_{e_2} for the edge e_2 analogously with B_{e_1}. Moreover, we can use the same pair of parameters r and ε for B_{e_1} and B_{e_2}.

For the strong monotonicity of the drawing Γ with two new bubbles B_{e_1} and B_{e_2} we have to show two conditions: (1) that from any vertex stacked into one of the new bubbles there

exists a strongly monotone path to any vertex y of any extension of Γ and (2) that there exists a strongly monotone path between any vertex stacked into B_{e_1} and any vertex stacked into B_{e_2}.

Since we use the same pair of r and ε for defining B_{e_1} and B_{e_2}, the condition (2) clearly holds as soon as r/ε is small enough. Thus we are left with ensuring that the condition (1) holds.

Consider the new bubble B_{e_1}, a point $x \in B_{e_1}$ and any vertex y of any extension Γ' of Γ. Since the drawing Γ' is strongly monotone, there exists a strongly monotone path P_{yw} in Γ' between y and w, furthermore by definition of α-safety we can choose P_{yw} being $\alpha(\Gamma)$-safe. Since w has only two incident edges in Γ', the last edge of the path P_{yw} is either $e_1 = (u, w)$ or $e_2 = (v, w)$. We distinguish between these two cases: in the first case we construct a path P_{yx} from y to x by re-routing the last edge of P_{yw} from (u, w) to (u, x) as illustrated in Figure 9a; in the second case we construct a path P_{yx} by appending the edge (w, x) to the end of P_{yw} as illustrated in Figure 9b;

It remains to show that P_{yx} is strongly monotone. First, observe that P_{yw} is strongly monotone and $\alpha(\Gamma)$-safe. By property 2, the final edges e_w of P_{yw} and e_x of P_{yx} satisfy $\measuredangle(\overrightarrow{e_w}, \overrightarrow{e_x}) < \alpha(\Gamma)/4$ and all other edges of these paths are identical. Thus, P_{yx} is $(3\alpha(\Gamma)/4)$-safe with respect to $\overrightarrow{yw}$. By property 3 $\measuredangle(\overrightarrow{yw}, \overrightarrow{yx}) < \alpha(\Gamma)/4$ and, therefore, P_{yx} is $(\alpha(\Gamma)/2)$-safe with respect to $\overrightarrow{yx}$ and thus in particular it is strongly monotone.

The arguments for a vertex stacked on e_2 into B_{e_2} are identical. ◀

Thus, we obtain the main result of this section:

▶ **Theorem 5.** *Every 2-tree admits a strongly monotone drawing.*

6 Conclusion

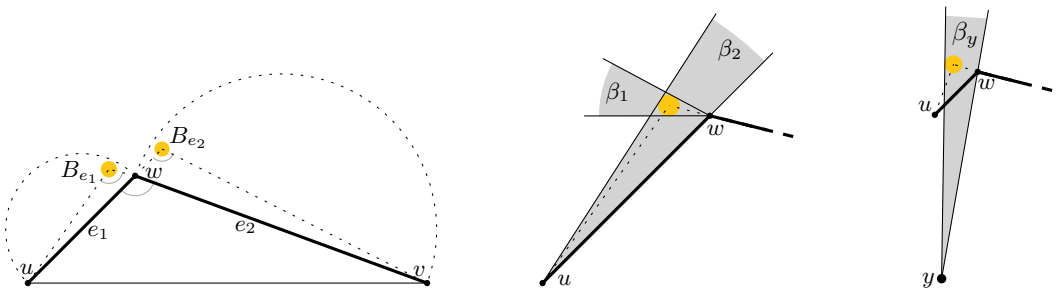
We have shown that any 3-connected planar graph, tree, outerplanar graph, and 2-tree admits a strongly monotone drawing. All our drawings require exponential area. For trees, this area bound has been proven to be required; however, it remains open whether the other graph classes can be drawn in polynomial area. Further, the question whether any 2-connected planar graph admits a strongly monotone drawing remains open. Last but not least, we could observe (using a computer-assisted search) that 2-connected graphs with at most 9 vertices admit a strongly monotone drawing, while there is exactly one connected graph with 7 vertices that is the smallest graph not admitting a strongly monotone drawing; see Figure 10.

References

1 Soroush Alamdari, Timothy M. Chan, Elyot Grant, Anna Lubiw, and Vinayak Pathak. Self-approaching graphs. In Walter Didimo and Maurizio Patrignani, editors, *Proc. 20th Int. Symp. Graph Drawing (GD'12)*, volume 7704 of *Lecture Notes Comput. Sci.*, pages 260–271. Springer, 2013. doi:10.1007/978-3-642-36763-2_23.

2 Patrizio Angelini, Enrico Colasante, Giuseppe Di Battista, Fabrizio Frati, and Maurizio Patrignani. Monotone drawings of graphs. *J. Graph Algorithms Appl.*, 16(1):5–35, 2012. doi:10.7155/jgaa.00249.

3 Patrizio Angelini, Walter Didimo, Stephen Kobourov, Tamara Mchedlidze, Vincenzo Roselli, Antonios Symvonis, and Stephen Wismath. Monotone drawings of graphs with fixed embedding. *Algorithmica*, 71:1–25, 2013. doi:10.1007/s00453-013-9790-3.

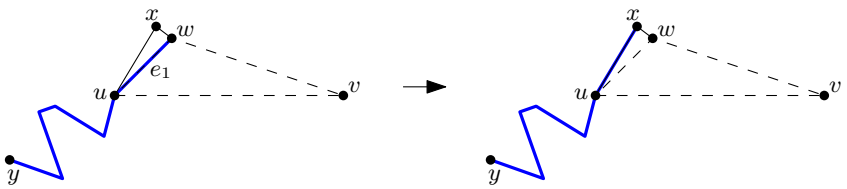


(a) Semi circles around e_1 and e_2

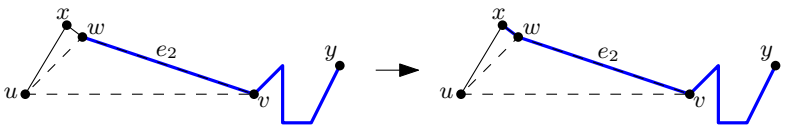
(b) Cones centered at u and w

(c) Cone centered at y

Figure 8 Illustrations for the placement of the new bubbles B_{e_1} and B_{e_2}.



(a) Rerouting in case the last edge of P_{yw} is e_1.



(b) Rerouting in case the last edge of P_{yw} is e_2.

Figure 9 A strongly monotone path P_{yw} from y to w is re-routed to x. Two possible cases are distinguished: The last edge of P_{yw} is either e_1 or e_2.

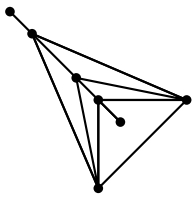


Figure 10 The unique connected 7-vertex graph without a strongly monotone drawing.

4 Esther M. Arkin, Robert Connelly, and Joseph S. B. Mitchell. On monotone paths among obstacles with applications to planning assemblies. In *Proc. 5th Ann. ACM Symp. Comput. Geom. (SoCG'89)*, pages 334–343. ACM, 1989. doi:10.1145/73833.73870.

5 Graham R. Brightwell and Edward R. Scheinerman. Representations of planar graphs. *SIAM J. Discrete Math.*, 6(2):214–229, 1993. doi:10.1137/0406017.

6 Hooman R. Dehkordi, Fabrizio Frati, and Joachim Gudmundsson. Increasing-chord graphs on point sets. In Christian Duncan and Antonios Symvonis, editors, *Proc. 22nd Int. Symp. Graph Drawing (GD'14)*, volume 8871 of *Lecture Notes Comput. Sci.*, pages 464–475. Springer, 2014. doi:10.1007/978-3-662-45803-7_39.

7 Raghavan Dhandapani. Greedy drawings of triangulations. *Discrete Comput. Geom.*, 43(2):375–392, 2010. doi:10.1007/s00454-009-9235-6.

8 Stefan Felsner. Convex drawings of planar graphs and the order dimension of 3-polytopes. *Order*, 18(1):19–37, 2001. doi:10.1023/A:1010604726900.

9 Dayu He and Xin He. Nearly optimal monotone drawing of trees. *Theoretical Computer Science*, 2016. To appear. doi:10.1016/j.tcs.2016.01.009.

10 Xin He and Dayu He. Compact monotone drawing of trees. In Dachuan Xu, Donglei Du, and Dingzhu Du, editors, *Proc. 21st Int. Conf. Comput. Combin. (COCOON'15)*, volume 9198 of *Lecture Notes Comput. Sci.*, pages 457–468. Springer, 2015. doi:10.1007/978-3-319-21398-9_36.

11 Xin He and Dayu He. Monotone drawings of 3-connected plane graphs. In Nikhil Bansal and Irene Finocchi, editors, *Proc. 23rd Ann. Europ. Symp. Algorithms (ESA'15)*, volume 9294 of *Lecture Notes Comput. Sci.*, pages 729–741. Springer, 2015. doi:10.1007/978-3-662-48350-3_61.

12 Md. Iqbal Hossain and Md. Saidur Rahman. Monotone grid drawings of planar graphs. In Jianer Chen, John E. Hopcroft, and Jianxin Wang, editors, *Proc. 8th Int. Workshop Front. Algorithmics (FAW'14)*, volume 8497 of *Lecture Notes Comput. Sci.*, pages 105–116. Springer, 2014. doi:10.1007/978-3-319-08016-1_10.

13 Weidong Huang, Peter Eades, and Seok-Hee Hong. A graph reading behavior: Geodesic-path tendency. In Peter Eades, Thomas Ertl, and Han-Wei Shen, editors, *Proc. 2nd IEEE Pacific Visualization Symposium (PacificVis'09)*, pages 137–144. IEEE Computer Society, 2009. doi:10.1109/PACIFICVIS.2009.4906848.

14 Christian Icking, Rolf Klein, and Elmar Langetepe. Self-approaching curves. *Math. Proc. Camb. Philos. Soc.*, 125:441–453, 1995. doi:10.1017/S0305004198003016.

15 Philipp Kindermann, André Schulz, Joachim Spoerhase, and Alexander Wolff. On monotone drawings of trees. In Christian Duncan and Antonis Symvonis, editors, *Proc. 22nd Int. Symp. Graph Drawing (GD'14)*, volume 8871 of *Lecture Notes Comput. Sci.*, pages 488–500. Springer, 2014. doi:10.1007/978-3-662-45803-7_41.

16 Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In Enrico Bertini, Catherine Plaisant, and Giuseppe Santucci, editors, *Proc. AVI Workshop Beyond Time Errors: Novel Eval. Methods Inform. Vis. (BELIC'06)*, pages 1–5. ACM, 2006. doi:10.1145/1168149.1168168.

17 Tom Leighton and Ankur Moitra. Some results on greedy embeddings in metric spaces. *Discrete Comput. Geom.*, 44(3):686–705, 2010. doi:10.1007/s00454-009-9227-6.

18 Martin Nöllenburg and Roman Prutkin. Euclidean greedy drawings of trees. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Proc. 21st Europ. Symp. Algorithms (ESA'13)*, volume 8125 of *Lecture Notes Comput. Sci.*, pages 767–778. Springer, 2013. doi:10.1007/978-3-642-40450-4_65.

19 Martin Nöllenburg, Roman Prutkin, and Ignaz Rutter. On self-approaching and increasing-chord drawings of 3-connected planar graphs. *J. Comput. Geom.*, 7(1):47–69, 2016. URL: http://jocg.org/v7n1p3.

20 Ananth Rao, Sylvia Ratnasamy, Christos H. Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In David B. Johnson, Anthony D. Joseph, and Nitin H. Vaidya, editors, *Proc. 9th Ann. Int. Conf. Mob. Comput. Netw. (MOBICOM'03)*, pages 96–108. ACM, 2003. doi:10.1145/938985.938996.

21 Kenneth Stephenson. *Introduction to circle packing: the theory of discrete analytic functions*. Cambridge Univ. Press, 2005. URL: http://www.cambridge.org/9780521823562.

Hyperplane Separability and Convexity of Probabilistic Point Sets

Martin Fink*[1], John Hershberger[2], Nirman Kumar†[3], and Subhash Suri‡[4]

1 **University of California, Santa Barbara, CA, USA**
2 **Mentor Graphics Corp., Wilsonville, OR, USA**
3 **University of California, Santa Barbara, CA, USA**
4 **University of California, Santa Barbara, CA, USA**

Abstract

We describe an $O(n^d)$ time algorithm for computing the exact probability that two d-dimensional probabilistic point sets are linearly separable, for any fixed $d \geq 2$. A probabilistic point in d-space is the usual point, but with an associated (independent) probability of existence. We also show that the d-dimensional separability problem is equivalent to a $(d+1)$-dimensional convex hull *membership* problem, which asks for the probability that a query point lies inside the convex hull of n probabilistic points. Using this reduction, we improve the current best bound for the convex hull membership by a factor of n [6]. In addition, our algorithms can handle "input degeneracies" in which more than $k+1$ points may lie on a k-dimensional subspace, thus resolving an open problem in [6]. Finally, we prove lower bounds for the separability problem via a reduction from the k-SUM problem, which shows in particular that our $O(n^2)$ algorithms for 2-dimensional separability and 3-dimensional convex hull membership are nearly optimal.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, F.2.2 Nonnumerical Algorithms and Problems, G.3 Probability and Statistics

Keywords and phrases probabilistic separability, uncertain data, 3-SUM hardness, topological sweep, hyperplane separation, multi-dimensional data

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.38

1 Introduction

Multi-dimensional point sets are a commonly used abstraction for modeling and analyzing data in many domains. The ability to leverage familiar geometric concepts, such as nearest neighbors, convex hulls, hyperplanes, or partitioning of the space, is both a powerful intuition-builder and an important analysis tool. As a result, the design of useful data structures and algorithms for representing, manipulating, and querying these kinds of data has been a major research topic not only in computational geometry and theoretical computer science but also many applied fields including databases, robotics, graphics and vision, data mining, and machine learning.

Many newly emerging forms of multi-dimensional data, however, are "stochastic": the input set is not fixed, but instead is a *probability distribution* over a finite population. A

* Supported by NSF under grants CCF-1161495 and CCF-1525817. Fink was partially supported by a postdoc fellowship of the German Academic Exchange Service (DAAD).
† Supported by NSF under grants CCF-1161495 and CCF-1525817.
‡ Supported by NSF under grants CCF-1161495 and CCF-1525817.

© Martin Fink, John Hershberger, Nirman Kumar, and Subhash Suri;
licensed under Creative Commons License CC-BY

leading source of these forms of data is the area of machine learning, used to construct data-driven models of complex phenomena in application domains ranging from medical diagnosis and image analysis to financial forecasting, spam filtering, fraud detection, and recommendation systems. These machine-learned models often take the form of a probability distribution over some underlying population: for instance, the model may characterize users based on multiple observable attributes and attempt to predict the likelihood that a user will buy a new product, enjoy a movie, develop a disease, or respond to a new drug.

In these scenarios, the model can often be viewed as a multi-dimensional probabilistic point set in which each point (user) has an associated probability of being included in the sample. More formally, a *probabilistic point* is a tuple (p, π), consisting of a (geometric) point $p \in \mathbb{R}^d$ and its associated probability π, with $0 < \pi \leq 1$. (We assume the point probabilities are independent, but otherwise put no restrictions on either the values of these probabilities or the positions of the points.) We are interested in computing geometric primitives over probabilistic data models of this kind. For instance, how likely is a particular point to be a vertex of the convex hull of the probabilistic input set? Or, how likely are two probabilistic data sets to be linearly separable, namely, lie on opposite sides of some hyperplane? The main computational difficulty here is that the answer seems to require consideration of an exponential number of subsets: by the independence of point probabilities, the sample space includes all possible subsets of the input. For instance, the probability that a point z lies on the convex hull is a weighted sum over exponentially many possible subsets for which z lies outside the subset's convex hull. These "counting type problems" are typically $\#P$-hard [31]. Indeed, many natural graph problems that are easily solved for deterministic graphs, such as connectivity, reachability, minimum spanning tree, etc., become intractable in probabilistic graphs [27], and in fact they remain intractable even for planar graphs [30] or geometric graphs induced by points in the plane [20]. Our work explores to what extent the underlying (low-dimensional) *geometry* can be leveraged to avoid this intractability.

Our contributions

The *hyperplane separability* problem for probabilistic point sets is the following. Given two probabilistic points sets $\mathcal{A}$ and $\mathcal{B}$ in $\mathbb{R}^d$ with a total of n points, compute the probability that a random sample of $\mathcal{A}$ can be separated from a random sample of $\mathcal{B}$ by a hyperplane. One can interpret this quantity as the *expectation* of $\mathcal{A}$ and $\mathcal{B}$'s linear separability. (Throughout the paper, we use hyperplane separability interchangeably with linear separability.) Because separability by any *fixed degree polynomial* is reducible to hyperplane separability, using well-known linearization techniques, our approach can be used to determine separability by non-linear functions such as balls or ellipsoids as well.

The *convex hull membership* problem asks for the probability that a query point p lies inside the convex hull of a random sample of a probabilistic point set $\mathcal{A}$. This is the complement of the probability that p is an extreme point (convex hull vertex) of $\mathcal{A} \cup \{p\}$. Finally, the *halfspace intersection* problem asks for the probability that a set of d-dimensional halfspaces, each appearing with an independent probability, has a non-empty common intersection.

Throughout, we focus on problems in dimensions $d \geq 2$; their 1-dimensional counterparts are easily solved in $O(n \log n)$ time. Our main results can be summarized as follows.

1. We present an $O(n^d)$ time and $O(n)$ space algorithm for computing the hyperplane separability of two d-dimensional probabilistic point sets with a total of n points. The same bound also holds for the *oriented* version of separability, in which the halfspace containing one of the sets, say $\mathcal{A}$, is prespecified.

2. We prove that the d-dimensional separability problem is at least as hard as the $(d+1)$-SUM problem [9, 16, 17, 18], which implies that our $O(n^2)$ bound for $d = 2$ is nearly tight. (The 3-SUM problem is conjectured to require $\Omega(n^{2-o(1)})$ time [22].) When the dimension d is non-constant, we show that the problem is $\#P$-hard.
3. We show that the convex hull membership problem in d-space has a linear-time reduction to a hyperplane separability problem in dimension $(d-1)$, and therefore can be solved in time $O(n^{d-1})$, for $d \geq 3$, improving the previous best bound of Agarwal et al. [6] by a factor of n. Our lower bound for separability implies that this bound is nearly tight for $d = 3$.
4. We show that the non-empty intersection problem for n probabilistic halfspaces in d dimensions can be solved in time $O(n^d)$. Equivalently, we compute the exact probability that a random sample from a set of n probabilistic linear constraints with d variables has a feasible solution.
5. Finally, our algorithms can cope with input degeneracies. Thus, for the convex hull membership problem, our result simultaneously improves the previous best running time [6] as well as eliminates the assumption of general position.

Related work

The topic of algorithms for probabilistic (uncertain) data is a subject of extensive and ongoing research in many areas of computer science including databases, data mining, machine learning, combinatorial optimization, theory, and computational geometry [7, 8, 19]. We will only briefly survey the results that are directly related to our work and deal with multi-dimensional point data. Within computational geometry and databases, a number of papers address nearest neighbor searching, indexing and skyline queries under the *locational uncertainty* model in which the position of each data point is given as a probability distribution [1, 2, 3, 4, 5, 10, 23, 26], as well as separability by a line in the plane [13].

The uncertainty model we consider, in which each point's position is known but its existence is probabilistic, has also been studied in a number of papers recently. The problem of computing the *expected* length of the Euclidean minimum spanning tree (MST) of n probabilistic points is considered in [20], and shown to be $\#P$-hard even in two dimensions. The closest pair problem and nearest neighbor searching for probabilistic points are considered in [21]. Suri, Verbeek, and Yıldız [29] consider the problem of computing the *most likely convex hull* of a probabilistic point set, give a polynomial-time algorithm for dimension $d = 2$, but show *NP*-hardness for $d \geq 3$. The complexity of the most likely Voronoi diagram of probabilistic points has been explored by Suri and Verbeek [28], and also by Li et al. [24]. In the work most closely related to ours, Agarwal et al. [6] consider a number of problems related to probabilistic convex hulls, including the convex hull membership probability. Their main result is an $O(n^d)$-time algorithm for computing the probability of convex hull membership, but it only works for points satisfying the following non-degeneracy condition: the projection of no $k+1$ points on a subspace spanned by any k coordinates may lie on a $(k-1)$-dimensional hyperplane, for any $2 \leq k \leq d$. Our new algorithm improves the running time by a factor of n as well as eliminates the need for non-degeneracy assumptions.

2 Separability of Probabilistic Point Sets

2.1 Preliminaries

A *probabilistic point* is a tuple (p, π), consisting of a (geometric) point $p \in \mathbb{R}^d$ and its associated probability π, with $0 < \pi \leq 1$. For notational convenience, we denote a set of

probabilistic points as $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ with an implicit understanding that $\pi(p_i)$ is the probability associated with p_i. We assume that the point probabilities are independent but otherwise place no restrictions on either the values of these probabilities or the positions of the points. We are interested in computing how often certain geometric properties occur for sets of probabilistic points. This requires reasoning about random samples in which each point p is drawn according to its probability $\pi(p)$. In particular, a fixed subset $A \subseteq \mathcal{P}$ occurs as a *random* sample with probability

$$\mathbf{Pr}\big[A\big] = \prod_{p \in A} \pi(p) \cdot \prod_{p \notin A} (1 - \pi(p)).$$

The central problem of our paper is to compute the probability that two probabilistic point sets $\mathcal{A}$ and $\mathcal{B}$ are linearly separable. We say that two sample sets $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$ are linearly separable if there exists a hyperplane H for which A and B lie in different (open) halfspaces of H. The *open* halfspace separation means that no point of $A \cup B$ lies on H, thus enforcing a strict separation. When there is no loss of generality, we assume that A lies *above* H, namely in the positive halfspace, and B lies *below* H. For ease of reference, we define an indicator function $\sigma(\mathcal{A}, \mathcal{B})$ for linear separability:

$$\sigma(A, B) = \begin{cases} 1 & \text{if } A, B \text{ are linearly separable} \\ 0 & \text{otherwise.} \end{cases}$$

We assume $\sigma(\emptyset, \emptyset) = 1$ to handle the trivial case. Given two probabilistic point sets $\mathcal{A}$ and $\mathcal{B}$, their *separation probability* is the joint sum over all samples:

$$\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big] = \sum_{A \subseteq \mathcal{A}, B \subseteq \mathcal{B}} \mathbf{Pr}\big[A\big] \cdot \mathbf{Pr}\big[B\big] \cdot \sigma(A, B)$$

This is also the *expectation* of the random variable $\sigma(A, B)$. Because each sample pair is deterministic, we can decide its linear separability in $O(n)$ time using fixed-dimensional linear programming algorithms of Megiddo or Clarkson [11, 25]. We can, therefore, *estimate* $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$ in polynomial time by drawing many samples A, B and returning the fraction of separable samples, but we are interested in the complexity of computing this quantity *exactly*. We begin our discussion by describing a reduction to a special kind of separability.

2.2 Reduction to Anchored Separability

A natural idea is to compute the sum $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$ by considering the $O(n^d)$ combinatorially distinct separating hyperplanes induced by the points of $\mathcal{A} \cup \mathcal{B}$. However, two point sets may be separable by many different hyperplanes, so we need to ensure that the probability is assigned to a unique *canonical* hyperplane.[1] Our main insight is the following: if we introduce an extra point z into the input, then the canonical hyperplane can be defined uniquely (and computed efficiently) with respect to z: in particular, we prove that the separating hyperplane at *maximum distance* from z is a canonical one. We call this artificially added point z the *anchor point*.

How does the true separation probability, namely $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$, relate to this *anchored separability* that includes an artificially added point *anchor*? It turns out the former can be calculated from two instances of the latter and one *lower dimensional* instance of the former.

[1] Dualizing the points to hyperplanes can simplify the enumeration of separating planes for the summation but does not address the over-counting problem.

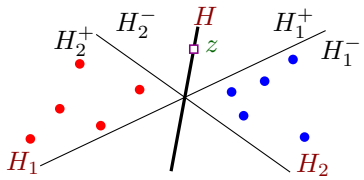


Figure 1 Proof of Lemma 1.

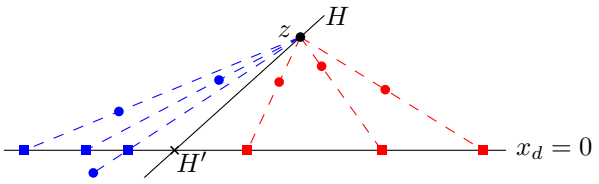


Figure 2 Proof of Lemma 3.

We initially assume that the input points are in general position, namely, no $k+1$ points of $\mathcal{A} \cup \mathcal{B}$ are affinely dependent for $k \leq d$, but revisit the degeneracy problem in Section 4. Without loss of generality, we also assume that all points have positive dth coordinate, and therefore lie above the hyperplane $x_d = 0$. Let the anchor point z be a point that lies *above* all the points of $\mathcal{A} \cup \mathcal{B}$ and is in general position with them. The probability of z is $\pi(z) = 1$, so it is always included in the sample.

If $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$ are two random samples and H a hyperplane separating them, then clearly z lies either (i) on the same side as A, (ii) on the same side as B, or (iii) on the hyperplane H. The cases (i) and (ii) are symmetric, and can be handled by including the anchor point once in A and once in B, but unfortunately they are *not disjoint:* A and B may admit separating hyperplanes with z lying on either side. Fortunately, the following lemma shows that this duplication of probability is precisely accounted for by case (iii). Finally, Lemma 3 shows that case (iii) itself is an instance of hyperplane separability in a lower dimension.

▶ **Lemma 1.** *Let z be the anchor point. Then there exist separating hyperplanes H_1, H_2 with z lying on the same side of H_1 as A but on the same side of H_2 as B if and only if there is another hyperplane H that passes through z and separates A from B.*

Proof. In the forward direction, if either H_1 or H_2 passes through z, we are done, so let us assume that neither contains z. Without loss of generality, assume that both hyperplanes contain A on their positive side, and B on their negative side. Thus, we have $A \subset H_1^+ \cap H_2^+$ and $B \subset H_1^- \cap H_2^-$. It follows that there are no points of $A \cup B$ in the region of the space $\Phi = \mathbb{R}^d \setminus \big((H_1^+ \cap H_2^+) \cup (H_1^- \cap H_2^-)\big)$. On the other hand, the anchor point z must lie in Φ because it lies on different sides of H_1 and H_2. See Figure 1 for illustration.

If H_1 and H_2 are parallel, then a hyperplane passing through z and parallel to H_1 is a separator, and we are done. On the other hand, if H_1 and H_2 intersect in a $(d-2)$-dimensional subspace, then we choose H as the hyperplane through z containing this subspace. This hyperplane lies in Φ, contains $H_1^+ \cap H_2^+$ and $H_1^- \cap H_2^-$ on opposite sides, and thus is a separating hyperplane for A and B.

To prove the reverse direction of the lemma statement, given a separating hyperplane H passing through z, we simply move H parallel to itself slightly, once toward A and once toward B. This completes the proof. ◀

Thus, event (iii) is precisely the intersection of events (i) and (ii). In the remainder of the paper, for notational convenience, we use $\mathcal{P} + z$ for the probabilistic point set $\mathcal{P} \cup \{(z, 1)\}$, where z is the anchor point with associated probability $\pi(z) = 1$. Let $\mathbf{Pr}\big[\sigma(z, \mathcal{A}, \mathcal{B})\big]$ denote the probability that sets $\mathcal{A}$ and $\mathcal{B}$ are linearly separable by a hyperplane passing through the anchor point z. Then, the preceding lemma gives the following result.

▶ **Lemma 2.** *Given two probabilistic point sets $\mathcal{A}$ and $\mathcal{B}$, we have the following equality:*

$$\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big] = \mathbf{Pr}\big[\sigma(\mathcal{A} + z, \mathcal{B})\big] + \mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B} + z)\big] - \mathbf{Pr}\big[\sigma(z, \mathcal{A}, \mathcal{B})\big].$$

Computing the probabilities $\mathbf{Pr}\big[\sigma(\mathcal{A}+z,\mathcal{B})\big]$ and $\mathbf{Pr}\big[\sigma(\mathcal{A},\mathcal{B}+z)\big]$ requires solving two instances of *anchored separability*, once with z included in $\mathcal{A}$ and once in $\mathcal{B}$. This leaves the last term $\mathbf{Pr}\big[\sigma(z,\mathcal{A},\mathcal{B})\big]$, which as the following lemma shows can be reduced to an instance of separability in dimension $d-1$.

Consider any sample $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$. We centrally project all these points onto the hyperplane $x_d = 0$ from the anchor point z: that is, the image of a point $p \in \mathbb{R}^d$ is the point $p' \in \mathbb{R}^{d-1}$ at which the line connecting z to p intersects the hyperplane $x_d = 0$. Observe that all points of $\mathcal{A} \cup \mathcal{B}$ have a well-defined projection because z lies above all of them.

▶ **Lemma 3.** *Let $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$ be two sample sets, and let A', B' be their projections onto $x_d = 0$ with respect to z. Then A and B are separable by a hyperplane passing through z if and only if A' and B' are linearly separable in $x_d = 0$.*

Proof. First, suppose there is a hyperplane H passing through z that separates A and B. We may assume that H is not parallel to $x_d = 0$; otherwise, rotate the input slightly. The intersection of H with $x_d = 0$ is a hyperplane H' in the $(d-1)$-dimensional subspace $x_d = 0$. See Figure 2. Clearly, the projection of each point p lies on the same side of H as does p. Since H separates A from B, it follows that H' separates A' from B'.

Conversely, suppose H' separates A' from B' in $x_d = 0$. The hyperplane H spanned by H' and z clearly separates A' from B' in $\mathbb{R}^d$. Since each point p lies on the same side of H as its projection, the point sets A and B are also separated by H. ◀

3 Computing Anchored Separability

We now describe our main technical result: efficiently computing the separation probability of two probabilistic sets when one of the sets contains the anchor point z. Without loss of generality, we explain how to compute $\mathbf{Pr}\big[\sigma(\mathcal{A}+z,\mathcal{B})\big]$. We can restrict our search to the $O(n^d)$ "combinatorially distinct" hyperplanes induced by the set of points $\mathcal{A} \cup \mathcal{B}$. Indeed, any free hyperplane can be translated and rotated until it passes through d distinct points of the input, without changing the *closed* halfspace membership of any point. Conversely, any hyperplane that contains A and B on opposite *closed* halfspaces and passes through at most d points can be translated and rotated until the same separation is realized by open halfspaces. (Recall that the input set of points, including the anchor z, is assumed to be in general position. We discuss how to handle degeneracies in Section 4.)

Given a hyperplane H, we can easily compute the probability that $\mathcal{A}+z$ lies in H^+ and $\mathcal{B}$ lies in H^-. The separation probabilities for different hyperplanes, however, are not independent: a sample $A \subseteq \mathcal{A}, B \subseteq \mathcal{B}$ may be separated by many different hyperplanes, and the algorithm needs to "assign" each separable sample to a unique hyperplane. We will assign a *canonical* separator for every pair $(A+z, B)$ of separable samples and then sum the probabilities over all possible canonical separators. Geometrically, our canonical separator is the hyperplane that separates $A+z$ from B and lies at *maximum distance* from the anchor z. Before we formalize the definition of a canonical separator and prove its uniqueness (cf. Section 3.2), we need the following important concept of a shadow cone.

3.1 The Shadow Cone

Given two points $u, v \in \mathbb{R}^d$, let $\mathit{shadow}(u,v) = \{v+\lambda(v-u) \mid \lambda \geq 0\}$ be the ray originating at v and directed along the line uv away from u. (If we place a light source at u, then this is the shadow cast by the point v.) Let $CH(P)$ denote the convex hull of a point set P. Given

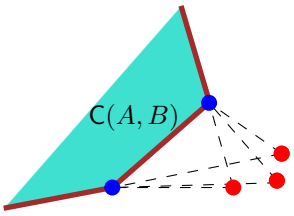


Figure 3 A shadow cone in two dimensions.

two sets of points A and B, with $A \cap B = \emptyset$, we define their *shadow cone* $\mathsf{C}(A, B)$ as the union of $shadow(u, v)$ for all $u \in CH(A)$ and $v \in CH(B)$.

In other words, if we place light sources at all points of $CH(A)$, then $\mathsf{C}(A, B)$ is the shadow cast by the convex hull $CH(B)$. (The shadow cone $\mathsf{C}(A, B)$ includes both umbra and penumbra of the shadow.) In the trivial case of $A = \emptyset$, we define $\mathsf{C}(\emptyset, B)$ to be the same as $CH(B)$. Figure 3 gives an illustration in two dimensions.

▶ **Lemma 4.** *The shadow cone $\mathsf{C}(A, B)$ is a (possibly unbounded) convex polytope, and if A and B are nonempty, $\mathsf{C}(A, B)$ is the convex hull of the union of $shadow(u, v)$, for all $u \in A, v \in B$.*

Each face of $\mathsf{C}(A, B)$ is *defined* by a subset of (at most d) points in $A \cup B$, and the defining set always includes at least one point of B. When all the points defining the face are in B, the face must be a (bounded) face of $CH(B)$; otherwise, it is an unbounded face. (See Figure 3.) We will use the following simple but important fact: if u is a point of A and $p \in CH(B)$, then $shadow(u, p)$ is contained in $\mathsf{C}(A, B)$. We are now ready to state and prove the important connection between the shadow cone and hyperplane separability of two subsets $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$, with $A \cap B = \emptyset$.

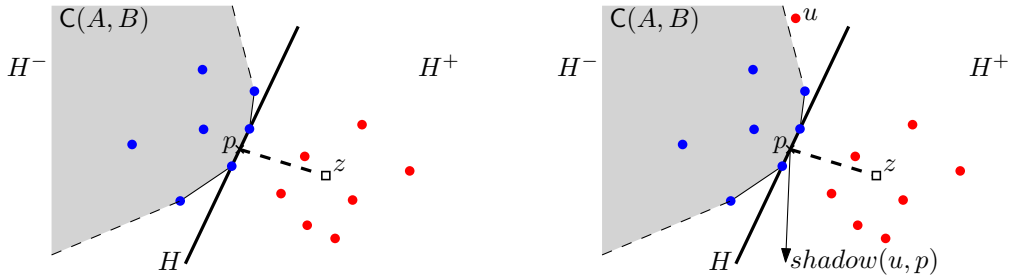
▶ **Lemma 5.** *$A + z$ and B can be separated by a hyperplane if and only if $z \notin \mathsf{C}(A, B)$.*

Proof. First, suppose there is a separating hyperplane H with $A + z \subset H^+$ and $B \subset H^-$. Then, we must also have $\mathsf{C}(A, B) \subset H^-$ because the shadow cone lies on the same side of H as B. Since $z \in H^+$ by assumption, this implies $z \notin \mathsf{C}(A, B)$.

For the converse, we assume $z \notin \mathsf{C}(A, B)$ and exhibit a separating hyperplane. Let $p \in \mathsf{C}(A, B)$ be the point in the shadow cone with minimum distance to the anchor z. Then the hyperplane H passing through p and orthogonal to the vector $p - z$ necessarily has z and the shadow cone on opposite sides, which follows from the convexity of $\mathsf{C}(A, B)$: if $z \subset H^+$, then $\mathsf{C}(A, B)$ is in the closure of the halfspace H^-. See Figure 4a.

It still remains to show that we can achieve *open* half-space separability of the sets $A + z$ and B. This depends crucially on the assumption of general position—indeed, if degeneracies exist, $z \notin \mathsf{C}(A, B)$ is not sufficient to prove strict separability. First, observe that no point of A can be in the open halfspace H^-. If such a point $u \in A$ were to exist, then the ray $shadow(u, p)$ would be contained in $\mathsf{C}(A, B)$, and there is a point on this ray that is closer to z than p, contradicting the minimality of p. See Figure 4b.

Because H is a supporting hyperplane of the shadow cone, the intersection $F = H \cap \mathsf{C}(A, B)$ is a face of $\mathsf{C}(A, B)$. Let $I = A \cap H$ and $J = B \cap H$ be the subsets of the sample points defining F (at most d due to general position). Since F contains at least one point of B, we have $|J| \geq 1$ and, therefore, $|I| < d$. Because no point of J is contained in the affine span of I by our non-degeneracy assumption, we can perform an infinitesimal rotation of H around the subface determined by I in the direction of z, so that all points of J (and thus



(a) The separator H. **(b)** $shadow(u, p)$ contains a point closer to z.

Figure 4 Illustration for the proof of Lemma 5.

B) lie in the open halfspace H^-. We then can translate the hyperplane by an infinitesimal amount away from z to ensure that all points of I (and thus A) lie in the open halfspace H^+. We now have a hyperplane whose open halfspaces separate the sample sets. ◀

3.2 Canonical Separating Hyperplanes

By Lemma 5, sets $A + z$ and B can be separated by a hyperplane if and only if $z \notin \mathsf{C}(A, B)$. Since $\mathsf{C}(A, B)$ is a convex set, there is a *unique* nearest point $p = \mathrm{np}(z, \mathsf{C}(A, B))$ on the boundary of $\mathsf{C}(A, B)$ with minimum distance to z. We define our *canonical hyperplane* $H(z, A, B)$ as the one that passes through p and is orthogonal to the vector $p - z$. Indeed, the proof of Lemma 5 already argues that $H(z, A, B)$ is a separating hyperplane in the sense that with infinitesimal rotation and translation it achieves open half-space separation between $A + z$ and B.

Our main idea now is to turn the separation question around and instead of asking "which hyperplane separates a particular sample pair A, B," we ask "for which pairs of samples A, B is H a canonical separator?" The latter formulation allows us to compute the separation probability $\mathbf{Pr}[\sigma(\mathcal{A} + z, \mathcal{B})]$ by considering at most $O(n^d)$ possible hyperplanes. The following lemmas encapsulate our definition of canonical separators.

▶ **Lemma 6.** *Let C be a d-dimensional convex polyhedron and z a point not contained in C. Then there is a unique point $p \in C$ that minimizes the distance to z, and a unique face of C whose relative interior contains p.*

▶ **Lemma 7.** *Let C be a d-dimensional convex polyhedron, z a point not contained in C, and p the point of C at minimum distance from z. If p lies in the relative interior of the face F of C, then the hyperplane H through p that is orthogonal to $p - z$ contains F. This hyperplane contains C in one of its closed halfspaces, and is the hyperplane farthest from z with this property.*

3.3 The Algorithm

Consider a random sample of input points $A \cup B$ such that $A + z$ and B are linearly separable. By Lemma 5, we know that $z \notin \mathsf{C}(A, B)$. Since $\mathsf{C}(A, B)$ is a convex polyhedron, there is a unique face F with $p = \mathrm{np}(z, \mathsf{C}(A, B))$ in its relative interior, by Lemma 6. Finally, by Lemma 7, F lies in the canonical hyperplane $H(z, A, B)$, which is the hyperplane passing through p and orthogonal to $p - z$.

We now consider the defining set of F, which consists of two subsets $I \subseteq A$ and $J \subseteq B$, with $|I \cup J| \leq d$ and $|J| \geq 1$. It follows from the definition of the shadow cone that

$F = \mathsf{C}(I, J)$. If F is finite, then it is the convex hull of its vertices, all of which belong to B, so $I = \emptyset$ and $F = CH(J) = \mathsf{C}(I, J)$. On the other hand, if F is unbounded, it is the convex hull of its finite vertices and a constant number of shadow rays. If F has dimension k, general position implies that its affine span includes exactly $k+1$ vertices of $A \cup B$. F is the shadow hull of these $k+1$ vertices, which constitute sets I and J.

Since F is the face of *smallest dimension* in $\mathsf{C}(A, B)$ containing p in its relative interior, we conclude that $I \cup J$ is the smallest subset of $A \cup B$ for which p lies in the relative interior of $\mathsf{C}(I, J)$. Equivalently, $I \cup J$ is the smallest subset of $A \cup B$ for which the canonical hyperplane $H(z, I, J)$ is the same as $H(z, A, B)$. Note that these sets are unique since the input is in general position.

This last property is the key to our algorithm: we simply enumerate all subsets $I \subseteq \mathcal{A}$ and $J \subseteq \mathcal{B}$, with $|I \cup J| \leq d$ and $|J| \geq 1$, and assign to the hyperplane $H(z, I, J)$ the separation probability of *all those samples $A \cup B$ that are separable and for which $H(z, I, J)$ is the canonical separator $H(z, A, B)$*. In particular, let us define the following function for the probability that the points defining the hyperplane $H(z, I, J)$ are in the sample and none of the remaining points of $\mathcal{A} \cup \mathcal{B}$ lies on its *incorrect side*.

$$\mathbf{Pr}\big[H(z, I, J)\big] = \prod_{u \in I \cup J} \pi(u) \cdot \prod_{u \in \mathcal{A} \cap H^-} (1 - \pi(u)) \cdot \prod_{u \in \mathcal{B} \cap H^+} (1 - \pi(u)).$$

The first term in the product is the joint probability that all points of $I \cup J$ are in the sample, while the second and third terms are the probabilities that none of the points of $\mathcal{A}$ (resp. $\mathcal{B}$) that lie in the negative halfspace (resp. positive halfspace) of $H(z, I, J)$ are chosen.

Finally, to decide whether $H(z, I, J)$ is the canonical hyperplane for a sample, we just need to check if the point closest to z in $\mathsf{C}(I, J)$ lies in the relative interior of $\mathsf{C}(I, J)$. The following algorithm **AnchoredSep** implements this construction.

Algorithm AnchoredSep:

Input: The point sets $\mathcal{A} + z$ and $\mathcal{B}$
Output: Their separation probability $\alpha = \mathbf{Pr}\big[\sigma(\mathcal{A} + z, \mathcal{B})\big]$
$\alpha = \prod_{u \in \mathcal{B}} (1 - \pi(u))$;
forall $I \subseteq \mathcal{A}, J \subseteq \mathcal{B}$ *where* $|I \cup J| \leq d, J \neq \emptyset$ **do**
  let $p = \mathrm{np}(z, \mathsf{C}(I, J))$;
  if *p lies in the relative interior of* $\mathsf{C}(I, J)$ **then**
    $\alpha = \alpha + \mathbf{Pr}\big[H(z, I, J)\big]$;
return α;

▶ **Theorem 8.** ***AnchoredSep*** *correctly computes the probability* $\mathbf{Pr}\big[\sigma(\mathcal{A} + z, \mathcal{B})\big]$.

Proof. The initial assignment $\alpha = \prod_{u \in \mathcal{B}} (1 - \pi(u))$ accounts for the trivial case when none of the points of $\mathcal{B}$ is present. The separation probability of any other outcome is associated with the minimal defining set $I \cup J$, and computed exactly once within the **forall** loop. ◀

3.4 Implementation in $O(n^d)$ Time and $O(n)$ Space

A naïve implementation of Algorithm **AnchoredSep** runs in $O(n^{d+1})$ time and $O(n)$ space: there are $O(n^d)$ subset pairs $I \subseteq A, J \subseteq B$ with $|I \cup J| \leq d$, $J \neq \emptyset$, and evaluating $\mathbf{Pr}\big[H(z, I, J)\big]$ for each one individually takes $O(n)$ time. We show how to reduce the average evaluation time to $O(1)$ per subset pair, which reduces the overall running time to $O(n^d)$. The main result of our paper can be stated as follows.

▶ **Theorem 9.** *Let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^d$ be two probabilistic sets of n points in general position, for $d \geq 2$. We can compute their probability of hyperplane separation $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$ in $O(n^d)$ worst-case time and $O(n)$ space.*

Proof. We compute the separation probability for all subsets I, J with $|I \cup J| < d$ explicitly by the naïve algorithm. This takes $O(n)$ time for each of $O(n^{d-1})$ subset pairs, for a total time of $O(n^d)$. To handle subset pairs with $|I \cup J| = d$, we process instances in linear-size groups. Let $\mathcal{A} \cup \mathcal{B} = \mathcal{P} = \{p_1, \ldots, p_n\}$. We group d-element subsets of $\mathcal{P}$ according to their $(d-1)$-subsets with smallest indices, as follows. For each $(d-1)$-element subset $P \subseteq \mathcal{P}$, let p_k be the element with maximum index. For each $p \in P_{>k} = \{p_{k+1}, \ldots, p_n\}$, we compute $\mathbf{Pr}\big[H(z, I, J)\big]$ for $I \cup J = P \cup \{p\}$. As we show below, this can be done in $O(n)$ time for each $(d-1)$-subset of $\mathcal{P}$, for a total bound of $O(n^d)$.

The $d-1$ points in P define a $(d-2)$-dimensional subspace. The $n-d+1$ points in $\overline{P} = \mathcal{P} \setminus P$ can be rotationally ordered around this subspace. For the moment, let us assume this rotational order is known. If p_0 is an arbitrary element of $\overline{P}$, we compute $\mathbf{Pr}\big[H(z, I, J)\big]$ in $O(n)$ time for $I \cup J = P \cup \{p_0\}$. We then process the points of $\overline{P}$ in rotational order. Each point p contributes a multiplicative factor to $\mathbf{Pr}\big[H(z, I, J)\big]$: $\pi(p)$ if $p \in I \cup J$, $(1 - \pi(p))$ if $p \in ((\mathcal{A} \cap H^-) \cup (\mathcal{B} \cap H^+))$, and 1 otherwise. When H rotates from one point $p \in \overline{P}$ to the next, the multiplicative factors for those two points change, and we can update $\mathbf{Pr}\big[H(z, I, J)\big]$ with two multiplications and two divisions. Whenever the conditions for acceptance of I, J are met—$J \neq \emptyset$, $H \cap P_{>k} \neq \emptyset$, $\mathrm{np}(z, \mathsf{C}(I, J))$ is in the relative interior of $\mathsf{C}(I, J)$—then we add $\mathbf{Pr}\big[H(z, I, J)\big]$ to the separation probability.

If we compute the rotational order of the points in $\overline{P}$ by sorting, we spend $O(n \log n)$ time per $(d-1)$-subset of $\mathcal{P}$, for a total running time of $O(n^d \log n)$. To do better, we use the ideas of *duality* [12] and *topological sweep* [14]. *Duality* is an order-preserving, invertible mapping between points and hyperplanes in $\mathbb{R}^d$. Each point $p \in \mathcal{P}$ dualizes to a hyperplane p^*, and the hyperplane H spanning d points $p_1, \ldots, p_d$ dualizes to the point H^* in dual space that is the intersection of the d hyperplanes $p_1^*, \ldots, p_d^*$. A subset $P \subseteq \mathcal{P}$ with $|P| = d-1$ dualizes to a line ℓ, and the rotational order of $\overline{P}$ (as defined above) around the $(d-1)$-dimensional subspace defined by P corresponds exactly to the order of intersections of the dual hyperplanes p^* (for $p \in \overline{P}$) with the dual line ℓ.

Ordering intersections along a line is still a sorting problem, but we can reduce the time by a logarithmic factor by considering arrangements of lines in two-dimensional planes. We consider all subsets $P \subseteq \mathcal{P}$ with $|P| = d-2$. Let p_k be the maximum-index point in a given P, and define $P_{>k} = \{p_{k+1}, \ldots, p_n\}$, as above. The intersection $\cap_{p \in P} p^*$ is a dual plane Q, and the intersection of Q with each p^*, for $p \in \overline{P}$, is a line. We use *topological sweep* [14] to visit the vertices of the arrangement of these $n-d+2$ lines in order along each line. We initialize $\mathbf{Pr}\big[H(z, I, J)\big]$ at the first vertex of each line, then update it in constant time per vertex during the sweep. At every vertex corresponding to two points in $P_{>k}$, if the acceptance criteria are met, we add the corresponding $\mathbf{Pr}\big[H(z, I, J)\big]$ to the separation probability. Topological sweep takes linear space and $O(n^2)$ time for each of the $O(n^{d-2})$ subsets $P \subseteq \mathcal{P}$ with $|P| = d-2$, so the total processing time is $O(n^d)$, and the total space is $O(n)$, for solving anchored separability. Since general separability is solved by two instances of anchored separability and a $(d-1)$-dimensional instance of general separability (that is solved recursively), this establishes the main result of our paper. ◀

4 Handling Input Degeneracies

Our algorithm so far has relied on the assumption that the input points or hyperplanes are in *general* (non-degenerate) position. That is, no $(k+2)$ points lie on a k-dimensional affine

space, or no $k+1$ hyperplanes meet in a $(d-k)$-dimensional subspace. These assumptions, while convenient for theory, are rarely satisfied in practice. They are especially troublesome in our analysis because of the need to define unique *canonical* sets. Indeed, when the input is degenerate, the need to choose a single canonical subset is the reason why the convex hull membership algorithm of [6] does not work—there is no efficient way to isolate *witness faces*. In this section, we show how to handle inputs that are not in general position.

Let us consider computing the probability of anchored separability $\mathbf{Pr}\big[\sigma(\mathcal{A}+z,\mathcal{B})\big]$ when the input sets are in a degenerate position. Our characterization of separable instances using shadow cones, Lemma 5, fails in the presence of degeneracy. As a concrete example, consider the case when $B \subseteq \mathcal{B}$ consists of a single point that lies in the convex hull of points $A \subseteq \mathcal{A}$, and all points of $A \cup B$ lie on a hyperplane that does not contain z. Although z lies outside $\mathsf{C}(A,B)$, we clearly cannot separate $A+z$ from B.

To address the problem of degenerate inputs, we apply a symbolic perturbation to the points. Part of our solution is standard Simulation of Simplicity [15], but the more important part is problem-specific. We convert degenerate non-separable samples into non-degenerate samples that are still non-separable. We first choose the anchor z above all points in $\mathcal{P} = \mathcal{A} \cup \mathcal{B}$ and outside the affine span of every d-tuple of $\mathcal{P}$. This can be done in $O(n^d)$ time. For each point $a \in \mathcal{A}$, we define a perturbed point $a' = a + \epsilon \cdot (a - z)$, for an infinitesimal $\epsilon > 0$. This point lies on the line supporting $\overline{az}$, but slightly farther from z than a. Similarly, for each $b \in \mathcal{B}$, define $b' = b + \epsilon \cdot (z - b)$, a point contained in $\overline{bz}$, but slightly closer to z than b. Let $\mathcal{A}', \mathcal{B}'$ be the sets of perturbed points corresponding to $\mathcal{A}$ and $\mathcal{B}$.

▶ **Lemma 10.** *Let $A \subseteq \mathcal{A}$ and $B \subseteq \mathcal{B}$ be two sample sets, and let A', B' be the corresponding perturbed sets. Then $A+z$ and B are strictly separable by a hyperplane if and only if $A'+z$ and B' are. Furthermore, if some hyperplane H with $z \notin H$ is a non-strict separator of $A'+z$ and B' for some ϵ, then H is a strict separator for any $\epsilon_0 < \epsilon$.*

Proof. First, suppose $A+z$ and B are strictly separable by a hyperplane H, with $A+z \subseteq H^+$. Let δ be the minimum distance between H and any point in $A \cup \{z\} \cup B$, and let Δ be the maximum distance between z and any point in $A \cup B$. The choice of any $\epsilon < \delta/\Delta$ ensures that $A'+z \subseteq H^+$ and $B' \subseteq H^-$. Conversely, if $A'+z \subseteq H^+$ and $B' \subseteq H^-$, then *a fortiori* $A+z \subseteq H^+$ and $B \subseteq H^-$ because each a is closer to z than a' and each b is farther from z than b' (and hence farther from H).

To prove the second part of the lemma, we simply note that if any point of $A' \cup B'$ lies on the separating hyperplane H, choosing any $\epsilon_0 < \epsilon$ moves the point off of H and into the desired halfspace. This completes the proof. ◀

We apply Simulation of Simplicity [15] to the point sets $\mathcal{A}'$ and $\mathcal{B}'$, with the symbolic perturbation of each point chosen to be of smaller order than the ϵ perturbation applied to produce $\mathcal{A}'$ and $\mathcal{B}'$. Simulation of Simplicity breaks any remaining degeneracies in the point set, so Lemma 5 holds and the algorithm of Section 3 works without modification. By Lemma 10, every separable point set in the symbolically perturbed data corresponds to a separable point set in the original data, and vice versa, so the Simulation of Simplicity computation correctly solves the original problem.

▶ **Theorem 11.** *Let $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^d$ be two probabilistic sets of n points, possibly in degenerate position, for $d \geq 2$. We can compute $\mathbf{Pr}\big[\sigma(\mathcal{A},\mathcal{B})\big]$, their probability of hyperplane separation, in $O(n^d)$ worst-case time and $O(n)$ space.*

5 Lower Bounds

In this section we argue that the running time of any algorithm that computes the probability of hyperplane separability must have an exponential dependence on dimension d. For any fixed d, we show that the separability problem is at least as hard as the k-SUM problem for $k = d + 1$. The proof of this result is interesting in that it uses Simulation of Simplicity [15], a technique for removing geometric degeneracies, as a *means to detect degeneracies*. In addition, we show that the problem is $\#P$-hard when $d = \Omega(n)$.

The k-SUM problem is a generalization of 3-SUM, which is a classical hard problem in computational geometry [9, 16, 17, 18]. The current conjectures state that the 3-SUM problem requires time at least $\Omega(n^{2-o(1)})$ [17, 18, 22], and that the k-SUM problem, for $k > 3$, has a lower bound of $\Omega(n^{\lceil k/2 \rceil})$ under some models of computation [9, 16, 17, 18]. We use the following variant of the k-SUM problem:

> **Problem k-SUM:** Given k sets containing a total of n real numbers, grouped into a single set Q and $k-1$ sets $R_1, R_2, \ldots, R_{k-1}$, determine whether there exist $k-1$ elements $r_i \in R_i$, one per set R_i, and an element $q \in Q$ such that $\sum_{i=1}^{k-1} r_i = q$.

▶ **Theorem 12.** *The d-dimensional hyperplane separability problem is at least as hard as $(d+1)$-SUM.*

Proof. Let P be a regular $(d-1)$-simplex, embedded in the hyperplane $x_d = 0$ in $\mathbb{R}^d$. Let $p_1, p_2, \ldots, p_d$ be the vertices of P, and let c be its barycenter. Given an instance $(Q, R_1, R_2, \ldots, R_d)$ of $(d+1)$-SUM, we define $d+1$ sets of d-dimensional points, one for each of the input sets, as follows. The sets $\mathcal{B}_i = \{p_i + (0, \ldots, 0, r) \mid r \in R_i\}$ correspond to the input sets R_i, for $i = 1, 2, \ldots, d$; let $\mathcal{B} = \cup_i \mathcal{B}_i$. The set $\mathcal{A} = \{c + (0, \ldots, 0, q/d) \mid q \in Q\}$ corresponds to the input set Q. Finally, add one extra point z to $\mathcal{A}$ that is higher than all other points (to serve as anchor) and lies on the same line as all points of $\mathcal{A}$. All points in $\mathcal{A} \cup \mathcal{B}$ lie on $d+1$ parallel lines perpendicular to the hyperplane $x_d = 0$. By construction, the $(d+1)$-SUM instance has a TRUE value if and only if there exists a hyperplane H defined by d vertices, one from each set $\mathcal{B}_i$, and a vertex $a \in \mathcal{A}$, where $a \neq z$, that lies in H.

We solve the separability problem twice, for two symbolically perturbed versions of $\mathcal{A}$ and $\mathcal{B}$. In particular, let $\mathbf{v}$ be the unit vector in direction x_d. For a given real parameter $\epsilon > 0$, denote by $\mathcal{A}^{+\epsilon}$ the set $\{a + \epsilon\mathbf{v} \mid a \in \mathcal{A}\}$; this is the result of slightly shifting the entire set $\mathcal{A}$ in direction $\mathbf{v}$. Define sets $\mathcal{A}^{-\epsilon}$, $\mathcal{B}^{+\epsilon}$, and $\mathcal{B}^{-\epsilon}$ analogously.

We assign every point in $\mathcal{A} \cup \mathcal{B}$ a probability of $1/2$, except z, which is assigned probability 1. We then compute the probability that $\mathcal{A}^{+\epsilon}$ is separable from $\mathcal{B}^{-\epsilon}$ by a non-vertical hyperplane H. We use Simulation of Simplicity [15] to compute the result for an infinitesimal perturbation value ϵ. An algorithm with running time $T(n)$ on ordinary points will run in time $O(T(n))$ on the symbolically perturbed points. Similarly, we compute the probability of separability for $\mathcal{A}^{-\epsilon}$ and $\mathcal{B}^{+\epsilon}$.

If there exists a hyperplane defined by d points of $\mathcal{B}$ that contains a point of $\mathcal{A}$, then the probability values returned by the two computations will differ—the $(d+1)$-tuple is strictly separable in $(\mathcal{A}^{+\epsilon}, \mathcal{B}^{-\epsilon})$ and strictly not separable in $(\mathcal{A}^{-\epsilon}, \mathcal{B}^{+\epsilon})$ (because z must lie above the hyperplane). If no such hyperplane exists, then the probability values will be equal, because the only sets $A \subseteq \mathcal{A}$, $B \subseteq \mathcal{B}$ whose separation probabilities are affected by the perturbation are those containing such a hyperplane.

By computing a separation probability twice, we solve an instance of $(d+1)$-SUM: the $(d+1)$-SUM instance is TRUE if and only if the two probabilities are not equal. Thus d-dimensional probabilistic separability is at least as hard as $(d+1)$-SUM. ◀

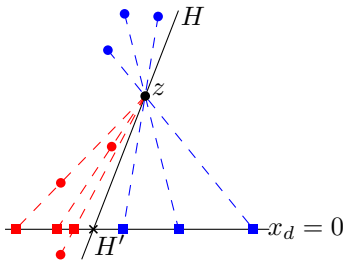


Figure 5 Projection from convex hull membership to separability.

The reduction from $(d+1)$-SUM to our problem is evidence that the algorithm of Section 3 is nearly optimal in two dimensions, and that an algorithm with running time $n^{o(d)}$ is unlikely for $d > 2$. Finally, we prove that the problem is $\#P$-hard if d can be as large as $\Omega(n)$.

▶ **Lemma 13.** *Computing* $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$ *is* $\#P$*-hard if the dimension* d *is not a constant.*

Proof. We reduce the $\#P$-hard problem of counting independent sets in a graph [31] to the separability problem. Consider an undirected graph $G = (V, E)$ on the vertex set $\{1, 2, \ldots, n\}$. For each i, we construct an n-dimensional point a_i as the unit vector along the ith axis. The collection of points $\{a_1, \ldots, a_i, \ldots, a_n\}$, each with associated probability $\pi_i = 1/2$, is our point set $\mathcal{A}$. Next, for each edge $e = (i, j) \in E$, we construct a point b_{ij} at the midpoint of the line segment connecting a_i and a_j. The set of points b_{ij}, each with associated probability 1, is the set $\mathcal{B}$. It is easy to see that there is a one-to-one correspondence between separable subsets of $\mathcal{A} \cup \mathcal{B}$ and the independent sets of G. Each separable sample occurs precisely with probability $(1/2)^n$, and therefore we can count the number of independent sets using the separation probability $\mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$. ◀

6 Convexity, Halfspace Emptiness and Related Problems

Given the fundamental role of hyperplanes in geometry, it is not surprising that many other problems can be reduced to hyperplane separability of points, possibly in a transformed space. In the following, we discuss a few sample problems that can be solved by reducing them to hyperplane separability of point sets.

Convex Hull Membership

Given a probabilistic set of points $\mathcal{P}$, the convex hull membership probability of a query point z is the probability that z lies in the convex hull of $\mathcal{P}$. We write this as

$$\mathbf{Pr}\big[z \in CH(\mathcal{P})\big] \;=\; \sum_{P \subseteq \mathcal{P},\ z \in CH(P)} \mathbf{Pr}\big[P\big].$$

Without loss of generality, assume that the query point is $z = (0, 0, \ldots, 0, 1)$. We further assume that none of the points of $\mathcal{P}$ has dth coordinate equal to 1, which is easily achieved by a rotation of the space. As a result, none of the lines pz, for $p \in \mathcal{P}$, is parallel to the hyperplane $x_d = 0$.

Given a point $p \in \mathcal{P}$, we define its *central projection* as the point p' at which the line pz meets the plane $x_d = 0$; see Figure 5. Let set $\mathcal{A}$ (resp. $\mathcal{B}$) be the central projections of all those points in $\mathcal{P}$ with $x_d > 1$ (resp. with $x_d < 1$), where each point inherits the associated probability of its corresponding point in $\mathcal{P}$. The sets $\mathcal{A}$ and $\mathcal{B}$ are sets of $(d-1)$-dimensional probabilistic points, with $|\mathcal{A}| + |\mathcal{B}| = n$. We get the following relation.

▶ **Lemma 14.** $\mathbf{Pr}\big[z \in CH(\mathcal{P})\big] = 1 - \mathbf{Pr}\big[\sigma(\mathcal{A}, \mathcal{B})\big]$.

Thus, convex hull membership in $\mathbb{R}^d$ is equivalent to point set separability in $\mathbb{R}^{d-1}$, resulting in the following bound.

▶ **Theorem 15.** *Given a probabilistic set of n points $\mathcal{P}$ in general position in $\mathbb{R}^d$, for any fixed $d \geq 3$, and a query point z in general position with $\mathcal{P}$, we can compute the convex hull membership probability $\mathbf{Pr}\big[z \in CH(\mathcal{P})\big]$ in time $O(n^{d-1})$.*

Our lower bounds imply that the time complexity is nearly optimal for $d = 3$, and that the convex hull membership problem is also $\#P$-hard when $d = \Omega(n)$.

Halfspace Emptiness and Linear Programming

Suppose we are given a set of n probabilistic halfspaces in $\mathbb{R}^d$, defined by a set of hyperplanes $\mathcal{H}$, where each hyperplane H is associated with an independent probability $\pi(H)$. What is the probability that a random sample of these halfspaces has non-empty common intersection? By using an order-preserving duality between points and hyperplanes, we can map this problem to an instance of point set separability, obtaining the following result:

▶ **Theorem 16.** *Given a set of n probabilistic halfspaces in $\mathbb{R}^d$, we can compute the probability that their common intersection is non-empty in time $O(n^d)$.*

7 Concluding Remarks

We considered the problem of hyperplane separability for probabilistic point sets. Our main result is that given two sets of n probabilistic points in $\mathbb{R}^d$, we can compute in $O(n^d)$ time the exact probability that their random samples are linearly separable. The same technique and result lead to similar bounds for several other problems, including the probability that a query point lies inside the convex hull of n probabilistic points, or the probability that n probabilistic halfspaces have non-empty intersection. One of the interesting connections we establish is the equivalence between d-dimensional hyperplane separability and $(d+1)$-dimensional convex hull containment. Another useful feature of our approach is its ability to handle degeneracies in input.

We also proved that the d-dimensional separability problem is at least as hard as the $(d+1)$-SUM problem [9, 16, 17, 18], which implies that our $O(n^2)$ algorithms for 2-dimensional separability or 3-dimensional convex hull membership are nearly optimal.

A number of open problems are suggested by our work. Our lower bounds suggest that an exponential dependence on d is probably unavoidable for the exact computation of separability, but better bounds may be possible for $d > 2$. In particular, can the hyperplane separability problem be solved with running time $\tilde{O}(n^{\lceil (d+1)/2 \rceil})$, instead of $O(n^d)$? Another important direction is to explore algorithms that can compute the probability within a small *multiplicative* factor.

References

1. A. Abdullah, S. Daruki, and J. M. Phillips. Range counting coresets for uncertain data. In *Proc. 29th SCG*, pages 223–232. ACM, 2013.
2. P. Afshani, P. K. Agarwal, L. Arge, K. G. Larsen, and J. M. Phillips. (Approximate) uncertain skylines. *Theory of Comput. Syst.*, 52:342–366, 2013.
3. P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty II. In *Proc. 32nd ACM PODS*, pages 115–126, 2013.

4 P. K. Agarwal, S.-W. Cheng, and K. Yi. Range searching on uncertain data. *ACM Trans. on Algorithms*, 8(4):43:1–43:17, 2012.

5 P. K. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest-neighbor searching under uncertainty. In *Proc. 31st ACM PODS*, pages 225–236. ACM, 2012.

6 P. K. Agarwal, S. Har-Peled, S. Suri, H. Yıldız, and W. Zhang. Convex hulls under uncertainty. In *Proc. 22nd ESA*, pages 37–48, 2014.

7 C. C. Aggarwal. *Managing and Mining Uncertain Data*. Springer, 2009.

8 C. C. Aggarwal and P. S. Yu. A survey of uncertain data algorithms and applications. *IEEE TKDE.*, 21(5):609–623, 2009.

9 N. Ailon and B. Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

10 C. Böhm, F. Fiedler, A. Oswald, C. Plant, and B. Wackersreuther. Probabilistic skyline queries. In *Proc. CIKM*, pages 651–660, 2009.

11 K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995.

12 M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

13 M. de Berg, A. D. Mehrabi, and F. Sheikhi. Separability of imprecise points. In *Proc. 14th SWAT*, pages 146–157. Springer, 2014.

14 H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38(1):165–194, 1989.

15 H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. on Graphics*, 9(1):66–104, 1990.

16 J. Erickson. Lower bounds for linear satisfiability problems. *Chicago J. Theoret. Comp. Sci.*, 1999(8), August 1999.

17 A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *CGTA*, 5(3):165–185, 1995.

18 A. Gronlund and S. Pettie. Threesomes, degenerates, and love triangles. In *Proc. 55th FOCS*, pages 621–630, 2014.

19 A. Jørgensen, M. Löffler, and J. M. Phillips. Geometric computations on indecisive and uncertain points. *CoRR*, abs/1205.0273, 2012.

20 P. Kamousi, T. M. Chan, and S. Suri. Stochastic minimum spanning trees in Euclidean spaces. In *Proc. 27th SCG*, pages 65–74, 2011.

21 P. Kamousi, T. M. Chan, and S. Suri. Closest pair and the post office problem for stochastic points. *CGTA*, 47(2):214–223, 2014.

22 T. Kopelowitz, S. Pettie, and E. Porat. Higher lower bounds from the 3SUM conjecture. In *SODA*, 2016.

23 H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic nearest-neighbor query on uncertain objects. In *Advances in Databases: Concepts, Systems and Applications*, volume 4443, pages 337–348. Springer, 2007.

24 Y. Li, J. Xue, A. Agrawal, and R. Janardan. On the arrangement of stochastic lines in $\mathbb{R}^2$. Unpublished manuscript (personal communication), 2015.

25 N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31(1):114–127, 1984.

26 C. F. Olson. Probabilistic indexing for object recognition. *IEEE PAMI*, 17(5):518–522, 1995.

27 J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.

28 S. Suri and K. Verbeek. On the most likely Voronoi diagram and nearest neighbor searching. In *Proc. 25th ISAAC*, pages 338–350, 2014.

29 S. Suri, K. Verbeek, and H. Yıldız. On the most likely convex hull of uncertain points. In *Proc. 21st ESA*, pages 791–802, 2013.

30 S. P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31:398–427, 1997.

31 L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979.

Subexponential Algorithms for Rectilinear Steiner Tree and Arborescence Problems*

Fedor Fomin[1], Sudeshna Kolay[2], Daniel Lokshtanov[3], Fahad Panolan[4], and Saket Saurabh[5]

1 **University of Bergen, Norway**
`fedor.fomin@ii.uib.no`
2 **Institute of Mathematical Sciences, India**
`skolay@imsc.res.in`
3 **University of Bergen, Norway**
`daniello@ii.uib.no`
4 **University of Bergen, Norway**
`fahad.panolan@ii.uib.no`
5 **University of Bergen, Norway; and Institute of Mathematical Sciences, India**
`saket@imsc.res.in`

Abstract

A *rectilinear Steiner tree* for a set T of points in the plane is a tree which connects T using horizontal and vertical lines. In the Rectilinear Steiner Tree problem, input is a set T of n points in the Euclidean plane ($\mathbb{R}^2$) and the goal is to find an rectilinear Steiner tree for T of smallest possible total length. A *rectilinear Steiner arborecence* for a set T of points and root $r \in T$ is a rectilinear Steiner tree S for T such that the path in S from r to any point $t \in T$ is a shortest path. In the Rectilinear Steiner Arborescense problem the input is a set T of n points in $\mathbb{R}^2$, and a root $r \in T$, the task is to find an rectilinear Steiner arborescence for T, rooted at r of smallest possible total length. In this paper, we give the *first* subexponential time algorithms for both problems. Our algorithms are deterministic and run in $2^{\mathcal{O}(\sqrt{n}\log n)}$ time.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity.

Keywords and phrases Rectilinear graphs, Steiner arborescence, parameterized algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.39

1 Introduction

In the Steiner Tree problem we are given as input a connected graph G, a non-negative weight function $w : E(G) \to \{1, 2, \ldots, W\}$, and a set of terminal vertices $T \subseteq V(G)$. The task is to find a minimum-weight connected subgraph of G, which is a tree, containing all terminal nodes T. Steiner Tree is one of the central and best-studied problems in Computer Science, we refer to the books of Hwang, Richards, and Winter [13] and Prömel and Steger [19] for thorough introductions to the problem.

In this paper we give the first *subexponential* algorithm for an important geometric variant of Steiner Tree, namely Rectilinear Steiner Tree. Here, for a given set of terminal points T in the Euclidean plane with ℓ_1-norm, the goal is to construct a network of minimum

* This work was partially supported by the European Research Council (ERC) via grants Rigorous Theory of Preprocessing, reference 267959 and PARAPPROX, reference 306992.

© Fedor Fomin, Sudeshna Kolay, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh;
licensed under Creative Commons License CC-BY

length connecting all points in T. This variant of the problem is extremely well studied, see Chapter 3 of the recent book of Brazil and Zachariasen [2] for an extensive overview of various applications of Rectilinear Steiner Tree.

For the purposes of this paper, it is convenient to define Rectilinear Steiner Tree as the Steiner Tree problem on a special class of graphs called Hanan grids. Recall that for two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ in the Euclidean plane $\mathbb{R}^2$, the *rectilinear* (ℓ_1, *Manhattan* or *taxicab*) distance between p_1 and p_2 is $d_1(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$.

▶ **Definition 1** (Hanan grid [11])**.** Given a set T of n terminal points in the Euclidean plane $\mathbb{R}^2$, the *Hanan grid* G of T is defined as follows. The vertex set $V(G)$ of G is the set of intersection points obtained by drawing a horizontal line (line parallel to x-axis) and a vertical line (line parallel to y-axis) through each point of T. For every $u, v \in V(G)$, there is an edge between u and v in G, if and only if u and v are adjacent along a horizontal or vertical line; the weight of edge uv is the rectilinear distance between u and v. For a Hanan grid G we define a weight function $\mathsf{recdist}_G$ from the edge set $E(G)$ to $\mathbb{R}$ such that for an edge $uv \in E(G)$, $\mathsf{recdist}_G(uv) = d_1(u, v)$. If the graph G is clear from the context we drop the subscript from $\mathsf{recdist}_G$ and only use $\mathsf{recdist}$.

Let us note that when G is the Hanan grid of a set T of n points, then $T \subseteq V(G)$, $|V(G)| \leq n^2$, and for every $u, v \in V(G)$, the weight of a shortest path between u and v is equal to $d_1(u, v)$. For an edge $uv \in E(G)$, we say that uv is a *horizontal (vertical) edge* if both points u and v are on the same horizontal (vertical) line.

It was shown by Hanan [11] that the Rectilinear Steiner Tree problem can be defined as the following variant of Steiner Tree.

Rectilinear Steiner Tree
Input: A set T of n terminal points, the Hanan grid G of T and $\mathsf{recdist}_G$.
Output: A minimum Steiner tree for T in G.

Previous work on Rectilinear Steiner Tree. Though the Rectilinear Steiner Tree problem is a very special case of the Steiner Tree problem, the decision version of the problem is known to be NP-complete [9]. A detailed account of various algorithmic approaches applied to this problem can be found in books of Brazil and Zachariasen [2] and Hwang, Richards, and Winter [13]. In particular, several exact algorithms for this problem can be found in the literature. The classic algorithm of Dreyfus and Wagner [5] from 1971 solves Steiner Tree on general graphs in time $3^n \cdot \log W \cdot |V(G)|^{\mathcal{O}(1)}$, where W is the maximum edge weight in G. For Rectilinear Steiner Tree, an adaptation of Dreyfus-Wagner algorithm provides an algorithm of running time $\mathcal{O}(n^2 \cdot 3^n)$. The survey of Ganley [7] summarizes the chain of improvements based on this approach concluding with the $\mathcal{O}(n^2 \cdot 2.62^n)$-time algorithm of Ganley and Cohoon [8].

Thobmorson et al. [21] and Deneen et al. in [4] gave randomized algorithms with running time $2^{\mathcal{O}(\sqrt{n} \log n)}$ for the special case of Rectilinear Steiner Tree when the terminal points T are drawn from a uniform distribution on a rectangle.

It is also worth mentioning relevant parameterized algorithms for Steiner Tree on general graphs. Fuchs et al. [6] provide an algorithm with run time $\mathcal{O}((2+\varepsilon)^n |V(G)|^{f(1/\varepsilon)} \log W)$. Björklund et al. [1] and Nederlof [16] gave $2^n |V(G)|^{\mathcal{O}(1)} \cdot W$ time algorithms for Steiner Tree. Let us remark that, since the distances between adjacent vertices in Hanan grid can be exponential in n, the algorithms of Björklund et al. and of Nederlof do not outperform the Dreyfus-Wagner algorithm for the Rectilinear Steiner Tree problem. Interesting recent developments also concern Steiner Tree on planar graphs, and more generally, on graphs

of bounded genus. While the existence of algorithms running in time subexponential in the number of terminals on these graph classes is still open, Pilipczuk et al. [17, 18] showed that Steiner Tree can be solved in time subexponential in the size of the Steiner tree on graphs of bounded genus.

In spite of the long history of research on Rectilinear Steiner Tree and Steiner Tree, whether Rectilinear Steiner Tree can be solved in time subexponential in the number of terminals remained open. In this paper we give the first such algorithm. The running time of our algorithm is $2^{\mathcal{O}(\sqrt{n}\log n)}$. Further, our techniques also yield the first subexponential algorithm for the related Rectilinear Steiner Arborescence problem.

▶ **Definition 2.** Let G be a graph, $T \subseteq V(G)$ a set of terminals, and $r \in T$ be a root vertex. A *Steiner arborescence of T in G* is a subtree $H \subseteq G$ rooted at r with the following properties:

- H contains all vertices of T, and
- For every vertex $t \in T \setminus \{r\}$, the unique path in H connecting r and t is also the shortest r–t path in G.

Let us note that if H is a Steiner arborescence of T in G, then for every vertex $v \in V(H)$, the unique path connecting r and v in H is also a shortest r–v path in G. The Rectilinear Steiner Arborescence problem is defined as follows.

Rectilinear Steiner Arborescence
Input: A set T of n terminal points, the Hanan grid G of T, a root $r \in T$ and $\mathsf{recdist}_G$.
Output: A minimum length Steiner arborescence of T.

Rectilinear Steiner Arborescence was introduced by Nastansky, Selkow, and Stewart [15] in 1974. Interestingly, the complexity of the problem was open until 2005, when Shu and Su [20] proved that the decision version of Rectilinear Steiner Arborescence is NP-complete. No subexponential algorithm for this problem was known prior to our work.

Our method. Most of the previous exact algorithms for Rectilinear Steiner Tree exploit Hwang's theorem [12], which describes the topology of so-called full rectilinear trees. Our approach here is entirely different. The main idea behind our algorithms is inspired by the work of Klein and Marx [14], who obtained a subexponential algorithm for Subset Traveling Salesman Problem on planar graphs. The approach of Klein and Marx was based on the following two steps: (1) find a locally optimal solution such that its union with some optimal solution is of bounded treewidth, and (2) use the first step to guide a dynamic program. While our algorithm follows this general scheme, the implementations of both steps for our problems are entirely different from [14].

We give a high level description of the algorithm for Rectilinear Steiner Tree. The algorithm for Rectilinear Steiner Arborescence is similar. In the first step we build in polynomial time a (possibly non-optimal) solution. The constructed tree $\widehat{S}$ can be seen as a "shortest path" rectilinear Steiner tree. The property of $\widehat{S}$ which is crucial for the algorithm is that there is an optimal Steiner tree S_{opt} such that graph $S = \widehat{S} \cup S_{\mathsf{opt}}$ is of treewidth $\mathcal{O}(\sqrt{n})$. For the second step we have $\widehat{S}$ at hand and know that there exists a subgraph S of G of treewidth $\mathcal{O}(\sqrt{n})$, which contains an optimal Steiner tree S_{opt} and $\widehat{S}$. Note that we only know that such a subgraph S exists, albeit with the extra information that $\widehat{S} \cup S_{\mathsf{opt}} \subseteq S$. It turns out that this is sufficient in order to mimic the dynamic programming algorithm for bounded treewidth, to solve Rectilinear Steiner Tree in time $2^{\mathcal{O}(\sqrt{n}\log n)}$.

Recall the dynamic programming algorithm for Steiner Tree on a rooted tree decomposition $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of the input graph, see e.g. [3, Theorem 7.8]. For each

node $t \in V(\mathbb{T})$, let V_t be the union of vertices contained in all bags corresponding to nodes of the subtree of $\mathbb{T}$ rooted at t and let S_t be the subgraph induced by V_t. Then, in the dynamic programming algorithm, for each t we store a set of states, capturing the interaction between a minimal Steiner tree and subgraph S_t; in particular the weight of a tree and the information about its connected components in S_t. It is possible to ensure that all the information carried out in each state is "locally" defined, i.e., the information can be encoded by the elements of the bag X_t only. Therefore, at the root node, there is a state that corresponds to an optimal Steiner tree.

In our algorithm, we define *types*, which are analogous to the states stored at a node of a tree decomposition. A type stores all the information of its corresponding state. Since we do not know the tree decomposition $\mathcal{T}$, a type stores more "local" information, to take care of the lack of definite information about S. We guess some structural information about the virtual tree decomposition $\mathcal{T}$ of S. For example, we guess the height h of the rooted tree $\mathbb{T}$. In a rooted tree decomposition, the level of a node t is defined by the height of the subtree rooted at t. In our algorithm, we generate types over h levels. The intuition is that, for a node $t \in \mathbb{T}$ of level h', for each state, of t, that was required for the dynamic programming over $\mathcal{T}$, there is an equivalent type generated in level h' of our algorithm. This implies that, at level h, there is a type equivalent to a state that corresponds to an optimal Steiner tree in S. In fact, we show that any Steiner tree corresponds to exactly one type $\widehat{D}$. During the iterative generation of types, the type $\widehat{D}$ may be generated many times. One such generation corresponds to an optimal solution. So the final step of the algorithm involves investigating all the occurrences of type $\widehat{D}$ in the iterative generation, and finding the weight of a minimum Steiner tree. As in dynamic programming, a backtracking step will enable us to retrieve a minimum Steiner tree of S and therefore of G. Due to paucity of space, many proofs have been ommitted from this version.

2 Preliminaries

For a positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For point u in the Euclidean plane $\mathbb{R}^2$, its position is denoted by the coordinates (u_x, u_y). For a set V, a partition $\mathcal{P}$ of V is a family of subsets of V, such that the subsets are pairwise disjoint and the union of the subsets is V. Each subset of a partition is called a block. Given two partitions $\mathcal{P}_1$ and $\mathcal{P}_2$, the join operation results in the partition $\mathcal{P}$, that is the most refined partition of V such that each block of $\mathcal{P}_1$ and $\mathcal{P}_2$ is contained in a single block of $\mathcal{P}$. The resultant partition $\mathcal{P}$ is often denoted as $\mathcal{P}_1 \sqcup \mathcal{P}_2$. Given a block B of $\mathcal{P}$, by $\mathcal{P} \setminus B$ denotes removing the block B from $\mathcal{P}$.

Given a graph G, its vertex and edge sets are denoted by $V(G)$ and $E(G)$ respectively. For a vertex $v \in V(G)$, the degree of a vertex, denoted as $\mathsf{degree}_G(v)$, is the number of edges of $E(G)$ incident with v. Given a vertex subset $V' \subseteq V(G)$, the induced subgraph of V', denoted by $G[V']$, is the subgraph of G, with V' as the vertex set and the edge set defined by the set of edges that have both endpoints in V'. An edge between two vertices u, v is denoted as uv. A path, where vertices $\{u_1, u_2, \ldots, u_\ell\}$ appear in sequence, is denoted by $u_1u_2 \cdots u_\ell$. Similarly, a path, where vertices u and v are the endpoints, is called a u–v path. Given a path P its non endpoint vertices are referred to as internal vertices. For an edge uv, an edge contraction in G results in a graph G' defined as follows. The vertex set $V(G') = V(G) \setminus \{u, v\} \cup v_{new}$, where v_{new} is a new vertex. The edge set $E(G') = E(G[V(G) \setminus \{u, v\}]) \cup \{wv_{new} | wu \in E(G) \vee wv \in E(G)\}$. By $G' \leq_s G$, we mean that the graph G' is a subgraph of G. Given a weight function $w : E(G) \to \mathbb{R}$, for a subgraph

H of G, we use $w(H)$ to denote the number $\sum_{e \in E(H)} w(e)$. Furthermore, for two vertices s and t in $V(G)$, by the term *shortest path* between s and t we mean the shortest path with respect to the weight function w. Given two subgraphs G_1, G_2 of G, a shortest path between G_1 and G_2 is a path P between a vertex $u \in V(G_1)$ and a vertex $v \in V(G_2)$ such that, among the shortest paths for each possible pair $\{u' \in V(G_1), v' \in V(G_2)\}$, P has minimum length. Given two graphs G_1 and G_2, the union graph $G_1 \cup G_2$ has $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$, while the edge set $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$.

Treewidth. Let G be a graph. A *tree-decomposition* of a graph G is a pair $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$, where $\mathbb{T}$ is a tree whose every node $t \in V(\mathbb{T})$ is assigned a subset $X_t \subseteq V(G)$, called a bag, such that the following conditions hold.

- $\bigcup_{t \in V(\mathbb{T})} X_t = V(G)$,
- for every edge $xy \in E(G)$ there is a $t \in V(\mathbb{T})$ such that $\{x, y\} \subseteq X_t$, and
- for any $v \in V(G)$ the subgraph of $\mathbb{T}$ induced by the set $\{t \mid v \in X_t\}$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(\mathbb{T})} |X_t| - 1$. The *treewidth* of G is the minimum width over all tree decompositions of G and is denoted by $\mathsf{tw}(G)$.

A tree decomposition $(\mathbb{T}, \mathcal{X})$ is called a *nice tree decomposition* if $\mathbb{T}$ is a tree rooted at some node r where $X_r = \emptyset$, each node of $\mathbb{T}$ has at most two children, and each node is of one of the following kinds:

- **Introduce node**: a node t that has only one child t' where $X_t \supset X_{t'}$ and $|X_t| = |X_{t'}| + 1$.
- **Introduce edge node** a node t labeled with an edge uv, with only one child t' such that $\{u, v\} \subseteq X_{t'} = X_t$. This bag is said to introduce uv.
- **Forget vertex node**: a node t that has only one child t' where $X_t \subset X_{t'}$ and $|X_t| = |X_{t'}| - 1$.
- **Join node**: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.
- **Leaf node**: a node t that is a leaf of $\mathbb{T}$, and $X_t = \emptyset$.

We additionally require that every edge is introduced exactly once. One can show that a tree decomposition of width t can be transformed into a nice tree decomposition of the same width t and with $\mathcal{O}(t|V(G)|)$ nodes, see e.g. [3].

Planar Graph Embeddings and Minors. A graph is *planar* if can be embedded in the plane. That is, it can be drawn on the plane in such a way that its edges intersect only at their endpoints. Formally, a planar embedding Π of a graph G consists of an injective mapping $\Pi : V(G) \to \mathbb{R}^2$ and a mapping Π of edges $uv \in E(G)$ to simple curves in $\mathbb{R}^2$ that join $\Pi(u)$ and $\Pi(v)$. Also, for $e, f \in E(G)$, $\Pi(e) \cap \Pi(f)$ contains only the images of common end vertices and for $e \in E(G)$ and $v \in V(G)$, $\Pi(v)$ is not an internal point of $\Pi(e)$. Now we define the notion of a minor of a graph G.

▶ **Definition 3.** A graph H is a minor of a graph G, denoted as $H \leq_m G$, if it can be obtained from a subgraph of G by a sequence of edge contractions.

Notice that this implies that H can be obtained from G by a sequence of vertex deletions, followed by a sequence of edge deletions and finally a sequence of edge contractions. We will need the following folklore observation.

▶ **Observation 4.** Suppose G, H are connected graphs such that H is a minor of G. Then H can be obtained from G only by edge deletions and contractions.

We also will be using the notion of a minor model.

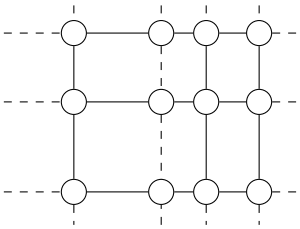


Figure 1 The union of solid edges define a subgrid of a grid.

▶ **Definition 5.** Let G and H be two connected graphs, and $H \leq_m G$. A *minor model* or simply a *model* of a graph H is a collection of pairwise disjoint vertex subsets $\mathcal{P}(H) = \{C_v \subseteq V(G) \mid v \in V(H)\}$ such that,

(a) $V(G) = \biguplus_{v \in V(H)} C_v$,

(b) for each $v \in V(H)$, $G[C_v]$ is connected, and

(c) for any $uv \in E(H)$, there exists $w \in C_u$ and $w' \in C_v$ such that $ww' \in E(G)$.

▶ Remark. It is important to point out that in general the definition of minor model does not demand that the vertex sets in $\mathcal{P}(H) = \{C_v \subseteq V(G) \mid v \in V(H)\}$ form a partition of $V(G)$. However, when both G and H are connected one can easily show that even this extra property can be assumed.

Grids and subgrids play an important role in this article. For a subset $W \subseteq [n]$, by $\max W$ ($\min W$) we denote the maximum (minimum) element of W.

▶ **Definition 6.** Let n, m be two positive integers. An $n \times m$ grid is a graph G such that $V(G) = \{v_{i,j} \mid i \in [n], j \in [m]\}$ and $E(G) = \{v_{ij}v_{i'j'} \mid |i - i'| + |j - j'| = 1\}$. For any $i \in [n]$, we call $\{v_{i1}, \ldots, v_{im}\}$ to be the i-th row of the grid G and for any $j \in [m]$, we call $\{v_{1j}, \ldots, v_{nj}\}$ to be the j-th a column of the grid G. The vertices in the first row, n-th row, the first column and m-th columns are called the *boundary vertices* of the grid. The vertices that are not boundary vertices are called *internal vertices*.

The graph H is called a subgrid of G, if there exist subsets $R \subseteq [n], C \subseteq [m]$ such that $V(H) = \{v_{ij} \in V(G) \ : \ (\min R \leq i \leq \max R) \wedge (\min C \leq j \leq \max C) \wedge (i \in R \vee j \in C)\}$ and $E(H) = \{v_{ij}v_{i'j'} \in E(G) \ : \ v_{ij}, v_{i'j'} \in V(H) \wedge (i = i' \in R \vee j = j' \in C)\}$. The set of vertices $\{v_{ij} \in V(H) \ : \ i \notin \{\min R, \max R\} \vee j \notin \{\min C, \max C\}\}$ are called the internal vertices of H. The set of vertices $\{v_{ij} \in V(H) \ : \ i \in R \wedge j \in C\}$ are called cross vertices. Finally, the set of vertices $\{v_{ij} \in V(H) \ : \ i \notin R \vee j \notin C\}$ are called subdivision vertices of H. (See Figure 1).

Given a planar graph G with an embedding Π, we call the vertices of the outer face as boundary vertices and all other vertices as internal vertices.

▶ **Definition 7.** Let G be a planar graph with a planar embedding Π, and C be a simple cycle of G. Let p_∞ be a point in the outer face of G in the embedding Π. Then removal of C from $\mathbb{R}^2$ divides the plane into two regions. The region that does not contain the point p_∞ is called the *internal region* of C, and the region containing p_∞ is called the *outer/external region* of C. A vertex in $V(G)$ is called internal if it lies in the internal region of C, and external if it lies in the external region of C. An edge in $E(G)$ is called an external edge if there is at least one point on its curve that lies in the external region. It is called an internal edge if there is at least one point on its curve that lies in the internal region.

By definition of a planar embedding, an edge of $V(G)$ can be exactly one of the three kinds: an edge of C, and external edge or an internal edge. Similarly a vertex can be exactly one of the three kinds: a vertex of C, an external vertex or an internal vertex.

▶ **Observation 8.** Let G be a planar graph with a planar embedding Π in $\mathbb{R}^2$. Let p_∞ be a point in the outer face of Π. Let H be a minor of G, and $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H. Then H is a planar graph. Furthermore, a planar embedding Π' of H can be obtained from Π that satisfies the following properties:

- Every vertex $v \in H$ is positioned in the place of a vertex in C_v.
- The point p_∞ is on the outer face of Π'.

We call such a planar embedding Π' as the *embedding derived* from Π.

On the proof of the first step of the algorithm, we will use the following auxiliary lemma.

▶ **Lemma 9.** *Let G and H be two connected planar graphs such that $H \leq_m G$ and let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H in G. Let also Π' be an embedding of H derived from a planar embedding Π of G. Suppose that H contains an induced subgraph H' isomorphic to a 3×3 grid. Let C' be the cycle formed by boundary vertices of H' and let v be the vertex of H' in the internal region of C'. Then there is a simple cycle C in G, such that:*

1. $V(C) \subseteq \bigcup_{u \in V(H'); u \neq v} C_u$.
2. *For each vertex $w \in G$ that is contained in the internal region of C in Π, there is a vertex $u \in H'$ with $w \in C_u$.*
3. *All vertices of C_v are completely contained in the internal region of C in Π.*
4. *There is a vertex $w \in C_v$ such that $\mathsf{degree}_G(w) \geq 3$.*

Our proof will also need the following Planar grid-minor theorem.

▶ **Proposition 10** ([10])**.** *Let t be a nonnegative integer. Then every planar graph G of treewidth at least $9t/2$ contains a $t \times t$ grid as a minor.*

Properties of shortest paths in the Hanan grid. Let G be the Hanan grid of a set of n points P. For a subgraph H of G and $v \in V(H)$, we say that v is a *bend vertex* if there exists at least one horizontal edge and at least one vertical edge from $E(H)$ incident with the vertex v. A path $R = u_1 \cdots u_\ell$, between u_1 and u_ℓ in G, is called a *monotone path* if there exists $i \in [\ell]$ such that the points $u_1, \ldots, u_i$ belong to a horizontal line and $u_i, \ldots, u_\ell$ belong to a vertical line or vice-versa. In other words, all the horizontal edges as well as all the vertical edges in R are contiguous.

The following observation contains some simple facts about monotone paths.

▶ **Observation 11.** Let u and v be two vertices of a Hanan grid G. Then,

(a) There is at least one and at most 2 monotone u–v paths,

(b) If the x-coordinates of u and v are equal, then there is only one monotone u–v path and all the edges in this path are vertical. Similarly, if the y-coordinates of u and v matches, the unique monotone u–v path consists of horizontal edges only.

(c) If there are two monotone paths between u and v, then one path has a horizontal edge incident with u and the other path has a vertical edge incident with u.

▶ **Definition 12.** Suppose we are given a Hanan grid G of a set of terminals T and two vertices $u, v \in V(G)$. Let $x_1 = \min\{u_x, v_x\}$, $x_2 = \max\{u_x, v_x\}$, $y_1 = \min\{u_y, v_y\}$, and $y_2 = \max\{u_y, v_y\}$. Let $V' = \{w \in V(G) | w_x \in [x_1, x_2], w_y \in [y_1, y_2]\}$. Then $G' = G[V']$, the subgraph of G induced by V', is called a grid defined by the two vertices u, v as its diagonal points.

▶ **Observation 13.** Given a Hanan grid G, a shortest path between any two vertices u, v has the property that the sequence of the x-coordinates of the vertices of the path is a monotone sequence and the sequence of their y coordinates is also a monotone sequence.

▶ **Observation 14.** Given a grid G, all shortest paths between two vertices u, v are contained in the grid $G' \leq_s G$ that is defined by u, v as its diagonal points. In fact, any path, with the property that the sequence of the x-coordinates of the vertices of the path is a monotone sequence and the sequence of their y coordinates is also a monotone sequence, and which is fully contained inside G', is a shortest path between u and v.

3 Subexponential Algorithm for Rectilinear Steiner Tree

In this section we give a subexponential algorithm for Rectilinear Steiner Tree. Let T be an input set of terminals (points in $\mathbb{R}^2$), $|T| = n$, and G be the Hanan grid of T. Furthermore, let $\mathsf{recdist}_G$ denote the weight function on the edge set $E(G)$. For brevity we will use $\mathsf{recdist}$ for $\mathsf{recdist}_G$. Our algorithm is based on a dynamic programming over vertex subsets of size $\mathcal{O}(\sqrt{n})$ of G. To reach the stage where we can apply the dynamic programming algorithm we do as follows. First, we define a rectilinear Steiner tree, called *shortest path RST*, and describe some of its properties. Next, we show that for a shortest path RST $\widehat{S}$, there is an optimal Steiner tree S_{opt} such that $\widehat{S} \cup S_{\mathsf{opt}}$ has bounded treewidth. Finally, keeping a hypothetical tree decomposition of $\widehat{S} \cup S_{\mathsf{opt}}$ in mind, we design a dynamic programming algorithm to obtain the size of a minimum rectilinear Steiner tree of G.

3.1 Shortest Path RST and its properties

In this part, we define a shortest path RST for a set $T = \{t_1, \ldots, t_n\}$ of input terminals and prove some useful properties of such a Steiner tree. We define a *shortest path RST* as follows.

Let G be the Hanan grid of T. We define a shortest path RST $\widehat{S}$ through the following constructive greedy process. Initially, we set S_1 to the graph $(\{t_1\}, \emptyset)$, which is a rectilinear Steiner tree of $\{t_1\}$. In the i^{th} step, we compute a rectilinear Steiner tree S_{i+1} of $\{t_1, \ldots t_{i+1}\}$ from S_i as follows. If $t_{i+1} \in V(S_i)$, then we set $S_{i+1} = S_i$. Otherwise, let v be a vertex in S_i such that $\mathsf{recdist}(v, t_{i+1}) = \min_{u \in V(S_i)} \mathsf{recdist}(u, t_{i+1})$. If there is only one monotone t–v path, then let Q be this path. Otherwise there are two monotone t–v paths such that one path has a horizontal edge incident with v and the other has a vertical edge incident with v. If there is a horizontal edge in S_i which is incident with v, then we choose Q to be the monotone t–v path such that the edge in Q incident with v is a horizontal edge. Otherwise we choose Q to be the monotone t–v path such that the edge in Q incident with v is a vertical edge. Then we construct S_{i+1} by adding a monotone path Q to S_i. After $n-1$ iterations, we construct a tree $\widehat{S} = S_n$ of G, which is a Steiner tree of T. This is our shortest path RST. It easy to see that one can construct a shortest path RST in polynomial time.

▶ **Lemma 15.** *Given a set T of terminal points and the Hanan grid G of T, a shortest path RST $\widehat{S}$ of T can be constructed in polynomial time.*

Next we give an upper bound on the number of bend vertices in a shortest path RST.

▶ **Lemma 16.** *The number of bend vertices in $\widehat{S}$ is at most n.*

3.2 Supergraph of an optimal RST with bounded treewidth

In this section we view the Hanan grid G as a planar graph and use this viewpoint to obtain the required upper bound on the treewidth of a subgraph of G. In particular, given a shortest

path RST $\widehat{S}$, we show the existence of an optimal Steiner tree S_{opt} such that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is sublinear in the number of terminal points T. First, we show that there is an optimal Steiner tree in G that has a bounded number of bends.

▶ **Lemma 17.** *Let T be a set of n points in $\mathbb{R}^2$ and G be the Hanan grid of T. Then there is an optimal rectilinear Steiner tree of T such that the number of bend vertices in the rectilinear Steiner tree is at most $3n$.*

With respect to a shortest path RST $\widehat{S}$, of G, we prove the next Lemma. In particular we show that the treewidth of $S = \widehat{S} \cup S_{\mathsf{opt}}$ is at most $41\sqrt{n}$. Here, S_{opt} is a carefully chosen optimal Steiner tree for T. In order to get the desired upper bound on the treewidth of S, we show that it does not contain $\mathcal{O}(\sqrt{n}) \times \mathcal{O}(\sqrt{n})$ grid as a minor. Towards this we prove that if there is a large grid then we can find a "clean part of the grid" (subgrid not containing vertices of T and bend vertices of either $\widehat{S}$ or S_{opt}) and reroute some of the paths in either $\widehat{S}$ or S_{opt}, that contradicts either the way $\widehat{S}$ is constructed or the optimality of S_{opt}.

▶ **Lemma 18.** *Given a set T of n points and a shortest path RST $\widehat{S}$ of T, there is an optimal rectilinear Steiner tree S_{opt} of T with the property that the treewidth of $\widehat{S} \cup S_{\mathsf{opt}}$ is bounded by $41\sqrt{n}$.*

Proof. Among the optimal Steiner trees of T with the minimum number of bend vertices, we select a tree S_{opt} which has maximum edge intersection with $E(\widehat{S})$. From Lemma 17, it follows that the number of bend vertices in S_{opt} is at most $3n$.

Let $S = \widehat{S} \cup S_{\mathsf{opt}}$. Let $\widehat{B}$ and B_{opt} be the set of bend vertices in $\widehat{S}$ and S_{opt} respectively. Let $U = T \cup \widehat{B} \cup B_{\mathsf{opt}}$ and $N = V(G) \setminus U$. Since $|T| = n$, $|\widehat{B}| \leq n$, and $|B_{\mathsf{opt}}| \leq 3n$, $|U| \leq 5n$. Let Π_S be a planar embedding of S, obtained by deleting all the edges and vertices not in S from the planar embedding Π of G. We show that the treewidth of S is at most $41\sqrt{n}$. We can assume that $n \geq 4$, as otherwise we can greedily find out the best rectilinear Steiner tree from the constant sized Hanan grid. For the sake of contradiction, assume that $\mathsf{tw}(S) > 41\sqrt{n}$. Then, by Proposition 10, there is a $9\sqrt{n} \times 9\sqrt{n}$ grid H appearing as a minor of S. Let $\mathcal{P}(H) = \{C_v | v \in V(H)\}$ be a minor model of H. Since H and G are connected graphs, $\mathcal{P}(H)$ is a partition of the vertex set $V(G)$. We identify a 3×3 subgrid H' of H by the following process. For any $v \in V(H)$, we mark the vertex v if $C_v \cap U \neq \emptyset$ (i.e, C_v contains a terminal or a bend vertex from $\widehat{S}$ or S_{opt}). Since $|U| \leq 5n$, the number of marked vertices in H is at most $5n$. Since H is a $9\sqrt{n} \times 9\sqrt{n}$ grid, there are at least $6n$ vertex disjoint 3×3 subgrids in H. This implies that there is a 3×3 subgrid H' in H such that each vertex of H' is unmarked. The fact that for $u \in V(H')$, $C_u \cap U = \emptyset$ implies the following observation.

▶ **Observation 19.** Let $u \in V(H')$ and $w \in C_u$.

- **(i)** $\mathsf{degree}_{\widehat{S}}(w), \mathsf{degree}_{S_{\mathsf{opt}}}(w) \in \{0, 2\}$. If for any $S_i \in \{\widehat{S}, S_{\mathsf{opt}}\}$, $\mathsf{degree}_{S_i}(w) = 2$, then the two edges in S_i incident with w are of same kind (either horizontal or vertical).
- **(ii)** If one horizontal (vertical) edge incident with w is present in S, then the other horizontal (vertical) edge incident with w is also present in S. Hence $\mathsf{degree}_S(w) \in \{2, 4\}$.

Note that H' is a connected graph and is a minor of a connected graph S. Let $\Pi_{H'}$ be a planar embedding derived from Π_S. By Lemma 9, we know that there is a simple cycle C' in S with the following properties.

- **(i)** $V(C') \subseteq \bigcup_{u \in V(H')} C_u$.
- **(ii)** For each vertex $w \in G$ that is contained in the internal region of C' in Π, there is a vertex $u \in H'$ with $w \in C_u$. In particular, all the vertices of $V(S) \setminus \bigcup_{v \in V(H')} C_v$ (which includes U) are not in the internal region of C'.

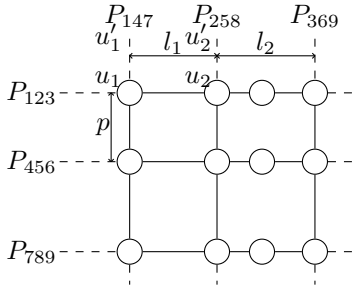


Figure 2 The subgrid G'.

(iii) For the internal vertex $v \in V(H')$, all the vertices in C_v are in the internal region of C'.

(iv) Finally, there is a vertex $w \in C_v$, in the internal region of C', such that $\mathsf{degree}_S(w) \geq 3$. By Observation 19, $\mathsf{degree}_S(w) = 4$.

That is, there is a cycle C' in the Hanan grid G such that $V(C') \subseteq V(S) \setminus U$, every point in the internal region of C' does not correspond to any vertex in U and there is a vertex w of degree 4 in S, which is in the internal region of C'. The following claim follows from Observation 19.

▶ **Claim 20.** *Let $u, v \in V(G)$ such that the points u and v are on the same horizontal line or on the same vertical line. If the line segment L connecting u and v does not intersect with the outer region of C', and there is an edge $v_1v_2 \in E(S)$ on the line L, then all the edges on the line segment L belong to $E(S)$.*

Let C' be a minimum-weight cycle satisfying properties $(i), (ii)$ and (iv) and w' be a vertex of degree 4 in the internal region of C'. Let $E_{w'}$ be the set of edges of S not in the outer region of C' and each edge either belongs to the horizontal line or vertical line containing w'.

▶ **Claim 21.** *Graph $G' = C' \cup E_{w'}$ is a subgrid of G. Moreover, $V(G') \subseteq V(G) \setminus U$, $E(G') \subseteq E(S)$, and all the subdivision vertices in G' have degree exactly 2 in S.*

The next claim provides us with the insight on how subpaths of $\widehat{S}$ and S_{opt} behave in G'.

▶ **Claim 22.** *Let F_h and F_v be the sets of horizontal and vertical edges in $G' = C' \cup E_{w'}$ respectively. Then exactly one of the following conditions is true.*

1. *$F_h \subseteq E(\widehat{S})$ and $F_v \subseteq E(S_{\mathsf{opt}})$.*
2. *$F_v \subseteq E(\widehat{S})$ and $F_h \subseteq E(S_{\mathsf{opt}})$.*

Note that G' is a 3×3 subgrid of G. Let G' be formed by horizontal paths $P_{123}, P_{456}, P_{789}$ and vertical paths $P_{147}, P_{258}, P_{369}$. Let $u_1, \ldots, u_9$ be the 9 vertices in G' such that the path P_{ijk}, where $i, j, k \in [9]$, contains the vertices u_i, u_j and u_k. Due to Claim 22, without loss of generality, we may assume that the horizontal paths belong to S_{opt} and the vertical paths belong to $\widehat{S}$. For a path P_{ijk}, we use P_{ij} and P_{jk} to denote the sub-paths of P_{ijk} connecting u_i and u_j, and u_j and u_k respectively. Let the length of the sub-path P_{12} be l_1 and the length of the sub-path P_{23} be l_2. By the definition of G', the length of P_{45} is also l_1, and the length of P_{56} is l_2. Let the length of P_{14} be p. The length of both P_{25} and P_{36} is p. (See Figure 2).

Suppose $l_1 + l_2 > 2p$. Then we consider the graph S^* formed by deleting in S_{opt} the path P_{123} and adding the two paths P_{14} and P_{36}. Since all the subdivision vertices of G' are of degree 2 in S, we have that S^* is a Steiner tree of weight strictly less than the weight of S_{opt}. This contradicts the choice of S_{opt}. Hence, this is not possible.

Suppose $l_1 + l_2 \leq 2p$. Without loss of generality, let $l_1 \leq l_2$. Thus, $l_1 \leq p$. Consider the two paths P_{147} and P_{258}. They are vertical paths of $\widehat{S}$, such that all the vertices in these paths belong to $V(G) \setminus U$ (that is, non-terminals and non-bend vertices) and have degree 2 in $\widehat{S}$ (by Observation 19). This implies that all the edges in any path $R \in \{P_{147}, P_{258}\}$ are added in a single step while constructing $\widehat{S}$. Since both the paths are parallel, by Observation 13, if a path R_1 in $\widehat{S}$ has both P_{147} and P_{258} as sub-paths, then R cannot be a shortest path between its endpoints. Thus, by construction of $\widehat{S}$, both P_{147} and P_{258} could not have been added to $\widehat{S}$ in a single step of the construction. Also by construction, one of them is added to $\widehat{S}$ before the other. Without loss of generality, let P_{147} be added before P_{258}. Again by construction, a path, containing P_{258} as a sub-path, was added in the i^{th} step, to connect a terminal t to the already constructed $\widehat{S}_{i-1}$. Let R^* be the path added to S_{i-1}. By definition of G', this terminal t must lie outside the region formed by the subgrid G'. Since P_{258} was part of a shortest path between $\widehat{S}_{i-1}$ and t, by Observation 13, t must lie on a row strictly higher than or strictly lower than the rows in G'. Suppose that this terminal lies above P_{123}. By Observation 19, both the vertical edges incident on u_1 belong to $\widehat{S}$. Since u_1 and u_2 are of degree 2 in $\widehat{S}$, both the vertical edges incident with u_1 as well as u_2 are added, when the path containing P_{147} or P_{258} is added to construct $\widehat{S}$. This implies that both the vertical edges incident with u_1 are present in S_{i-1}. Let u_1u_1' be the vertical edge present in S_{i-1} and not in $E(P_{147})$. Let u_2u_2' be the vertical edge not present in P_{258}. Now, consider the path R_2, between u_2' and t, obtained by concatenating the horizontal path between u_1' and u_2', and the sub-path of R^* connecting u_2' and t. The length of the path R_2 is strictly less than that of R^* and $u_1' \in S_{i-1}$. This is a contradiction. The case when t lies below any row in G' is identical to the other case. Thus, there is no such subgrid G' of G, such that G' is a subgraph of S. This implies that there is no $9\sqrt{n} \times 9\sqrt{n}$ grid as a minor in S. Due to Proposition 10, the treewidth of S must be at most $\frac{9}{2} \cdot 9\sqrt{n} = 41\sqrt{n}$. ◀

3.3 Dynamic Programming Algorithm for Rectilinear Steiner Tree

In this section we utilize all the results proved in the previous sections and design our algorithm for Rectilinear Steiner Tree. By Lemma 18, we known that given a shortest path RST $\widehat{S}$, there exists an optimum Steiner tree S_{opt} such that the treewidth of $S = \widehat{S} \cup S_{\mathsf{opt}}$ is bounded by $41\sqrt{n}$. The idea of the algorithm is to *implicitly* do a dynamic programming over a tree decomposition of S, *even though we do not know what S is*, to compute an optimum Steiner tree for T.

To give a proper intuition to our algorithm, we first recall the important step of the dynamic programming algorithm for Steiner tree over the tree decomposition of the graph S (see [3, Theorem 7.8] for more details). Let $(\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S where $\mathbb{T}$ is a rooted tree. For a node t, let V_t be the union of all the bags present in the subtree of $\mathbb{T}$ rooted at t. For a node t, we define a graph $S_t = (V_t, E_t = \{e \in E(S) \ : \ e \text{ is introduced in the subtree rooted at } t\})$. The important step in the algorithm for Steiner Tree is to compute the following information: for each bag X_t, $X \subseteq X_t$ and a partition $\mathcal{P} = (P_1, \ldots, P_q)$ of X, the value $c[t, X, \mathcal{P}]$ is the minimum weight of a subgraph F of S_t with the following properties:

1. F has exactly q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$ for all $i \in [q]$. That is, $\mathcal{P}$ corresponds to connected components of F.
2. $X_t \cap V(F) = X$. That is, the vertices of $X_t \setminus X$ are untouched by F.
3. $T \cap V_t \subseteq V(F)$. That is, all the terminal vertices in S_t belong to F.

For our purposes the second step is redundant but we still carry it out to give a proper analogy with the known algorithm for Steiner Tree over graph of bounded treewidth we

are referring to. Note that the number of blocks in the partition $\mathcal{P}$ is q. Throughout this section, q will denote the number of blocks of the partition in question.

It is known that computing the values $c[t, X, \mathcal{P}]$ for each tuple $(t, X, \mathcal{P})$, where $t \in V(\mathbb{T}), X \subseteq X_t$ and $\mathcal{P}$ is a partition of X, is enough to compute the value of an optimum Steiner tree of T in S and this can be computed in time $\mathsf{tw}^{\mathcal{O}(\mathsf{tw})}|V(S)|^{\mathcal{O}(1)}$ (See chapter 7 in the book [3]). In our case, we do not know the graph $S = \widehat{S} \cup S_{\mathsf{opt}}$ and a tree decomposition of S, but we know that the treewidth of S is at most $41\sqrt{n}$. This implies that the number of choices for bags in a tree decomposition of S is bounded by $n^{\mathcal{O}(\sqrt{n})}$. Consider the properties 1 and 2 mentioned above. They are *local* properties of the witness subgraph F with respect to the bag X_t. But the property 3 says that all the terminals in the subgraph S_t should be present in F. In fact we can bound the *potential* sets $T \cap V(S_t)$ using the rectilinear Steiner tree $\widehat{S}$. Observe that any bag X_t is a separator of size $\mathcal{O}(\sqrt{n})$ for S and thus for $\widehat{S}$, which implies that *for every connected component C of $\widehat{S} - X_t$, either $T \cap V(C)$ is fully in V_t or no vertex in $T \cap V(C)$ belongs to V_t.* Since each vertex in G has degree at most 4 and X_t is a bag in a tree decomposition, the number of connected components in $\widehat{S} - X_t$ is at most $4|X_t| \leq 164(\sqrt{n}+1)$. As observed before, for any connected component C of $\widehat{S} - X_t$, either $T \cap V(C)$ is fully in V_t or no vertex in $T \cap V(C)$ belongs to V_t. This implies that the potential sets $T' \subseteq T$ such that $T' = (V_t \setminus X_t) \cap T$ is bounded by $2^{164(\sqrt{n}+1)}$ and we can enumerate them in sub-exponential time. Using this observation we could keep track of property 3 as well, even though we do not know the graph S and its tree decomposition. Now, we move towards the formal explanation of our algorithm. Towards that we define a notion of *type* which is the analogue of a tuple $(t, X, \mathcal{P})$ in the dynamic programming we explained above.

▶ **Definition 23.** A type is a tuple $(Y, Y' \subseteq Y, \mathcal{P}, T')$ such that the following holds.

(i) Y is a subset of $V(G)$ of size at most $41\sqrt{n} + 2$.

(ii) $\mathcal{P}$ is a partition of Y'.

(iii) There exists a set of components $C_1, \ldots, C_q$ in $\widehat{S} - Y$ such that $T' = T \cap (Y' \cup \bigcup_{i=1}^{q} V(C_i))$.

Informally, in a type $(Y, Y', \mathcal{P}, T')$, Y represents a potential bag Y of a node (say t) in a tree decomposition of S. The set Y' and partition $\mathcal{P}$ have the same meaning as that of $(t, Y', \mathcal{P})$ in the normal dynamic programming algorithm for Steiner Tree. The set T' is the set of terminals in the graph S_t. The next lemma gives an upper bound on the number of types.

▶ **Lemma 24.** *There is a $2^{\mathcal{O}(\sqrt{n}\log n)}n^{\mathcal{O}(1)}$ time algorithm $\mathcal{B}$ enumerating all the types.*

Our algorithm is a dynamic programming algorithm over the types of S. As motivated earlier, this algorithm essentially describes the ideas of the dynamic programming algorithm for Steiner Tree over a tree decomposition of an input graph. Let $N = 3(|V(G)| + |E(G)|)$. Our algorithm computes values $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. We want the table $\mathcal{A}[,]$ to contain all the information that is necessary for the correctness of the dynamic programming algorithm for Steiner Tree over a tree decomposition of S. To motivate the definition of $\mathcal{A}[,]$ we assume a *hypothetical tree decomposition* $\mathcal{T} = (\mathbb{T}, \mathcal{X} = \{X_t\}_{t \in V(\mathbb{T})})$ of S. For ease of understanding, let it be a nice tree decomposition and let the tree be rooted at a node $r \in \mathbb{T}$. The level of a vertex $t \in \mathbb{T}$ is the height of the subtree of $\mathbb{T}$ rooted at t. The *height* of a node t is the number of vertices in the longest downward path to a leaf from that node. Note that the level of any node of $\mathbb{T}$ is at most N. Suppose $t \in \mathbb{T}$ is a node at level i, and corresponds to the bag X_t. Let V_t be the union of bags present in the subtree rooted at t. Let the graph S_t be defined as $(V_t, \{e \mid e \text{ is introduced in a the subtree rooted at } t\})$. Let $T' = V_t \cap T$. Then, for any $X \subseteq X_t$, and a partition $\mathcal{P}$ of X having q blocks, $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')] = c[t, X, \mathcal{P}]$. As mentioned before, $c[t, X, \mathcal{P}]$ is the minimum weight of the subgraph F of S_t such that the

following hold: (i) F has q connected components $C_1, \ldots, C_q$ such that $\emptyset \neq P_i = X_t \cap V(C_i)$, (ii) $X_t \cap V(F) = X$, and (iii) $T \cap V_t \subseteq V(F)$. For other pairs (i, D), we do not guarantee that the value of $\mathcal{A}[i, D]$ is meaningful. However, it is enough to maintain reasonable values for only the above subset of pairs (i, D). Of course we do not know S and thus we do not know the tree decomposition $\mathcal{T}$, so we store values in the table $\mathcal{A}[,]$ in such a way that given *any* nice tree decomposition of S, we have information pertaining to it. Thus, given a pair (i, D) where $D = (Y, Y' \subseteq Y, \mathcal{P}, T')$, we view Y as a bag of some hypothetical nice tree decomposition, $\mathcal{T}$, of S and assume that the level of the bag corresponding to Y in $\mathcal{T}$ is i. At a level i of this hypothetical nice tree decomposition, any bag is one of at most five kinds. We guess the relationship between a bag at level i and its children, which must be at level $i-1$. For example, if our hypothetical node t corresponds to an introduce vertex bag X_t, then we pretend that we know X_t, the child node t', the bag $X_{t'}$, and the vertex v that is being introduced at node t. Thereafter, for a subset $X \subseteq X_t$, and a partition $\mathcal{P}$ of X, we try to compute $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')]$ using that values of $\mathcal{A}$ calculated at step $i-1$ of the algorithm. The calculation ensures that $\mathcal{A}[i, (X_t, X, \mathcal{P}, T')] = c[t, X, \mathcal{P}]$. In what follows we give a formal definition of $\mathcal{A}[,]$. We write a recurrence relation for $\mathcal{A}[i, D]$, where $i \in [N]$ and D is a type. We fix a terminal t^* in T

$$\mathcal{A}[1, D] = \begin{cases} 0 & \text{if } D = (\{t^*\}, \{t^*\}, \{\{t^*\}\}, \{t^*\}) \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

To define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and a type $D = (Y, Y', \mathcal{P}, T')$, we first define many intermediate values and take the minimum over all such values.

We first try to *view* Y as an introduce node in some nice tree decomposition of S and having level i. This viewpoint results in the following recurrence. For all $v \in Y$,

$$I_v[i, D] = \begin{cases} \infty & \text{if } v \notin Y' \text{ and } v \in T \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y', \mathcal{P}, T')] & \text{if } v \notin Y' \text{ and } v \notin T \\ \mathcal{A}[i-1, (Y \setminus \{v\}, Y' \setminus \{v\}, \mathcal{P} \setminus \{\{v\}\}, T' \setminus \{v\})] & \text{if } v \in Y' \end{cases} \tag{2}$$

Intuitively, if Y is a bag corresponding to a node t in a tree decomposition of S and T' is the set of terminals in S_t, then Equation 2 corresponds to the computation of $c[t, Y', \mathcal{P}]$ in the dynamic programming algorithm of Steiner Tree. See [3, Theorem 7.8] for more detailed explanation.

For all $u, v \in Y$ and $uv \in E(G)$,

$$I_{uv}[i, D] = \min \left\{ \min_{\mathcal{P}'} \{\mathcal{A}[i-1, (Y, Y', \mathcal{P}', T')] + \mathsf{recdist}(uv)\}, \mathcal{A}[i-1, D] \right\}, \tag{3}$$

where $\mathcal{P}'$ varies over partitions of Y' such that u and v are in different blocks of $\mathcal{P}'$ and by merging these two blocks we get the partition $\mathcal{P}$. Note that if $\{u, v\} \not\subseteq Y'$ or u and v are in same block of $\mathcal{P}$, then Equation 3 gives $I_{uv}[i, D] = \mathcal{A}[i-1, D]$. Equation 3 corresponds to the computation of values in the introduce edge node where the edge uv is introduced.

For all $w \in V(G)$,

$$F_w[i, D] = \min\{\min_{\mathcal{P}'}\{\mathcal{A}[i-1, (Y \cup \{w\}, Y' \cup \{w\}, \mathcal{P}', T')]\}, \mathcal{A}[i-1, (Y \cup \{w\}, Y', \mathcal{P}, T')]\}, \tag{4}$$

where $\mathcal{P}'$ in the inner minimum varies over all the partitions obtained by adding w to one of the existing blocks. Equation 4 corresponds to computation in a forget node where w is forgotten.

$$J[i, D] = \min_{\substack{\mathcal{P} = \mathcal{P}_1 \sqcup \mathcal{P}_2 \\ T' = T'_1 \cup T'_2}} \{\mathcal{A}[i-1, (Y, Y', \mathcal{P}_1, T'_1)] + \mathcal{A}[i-1, (Y, Y', \mathcal{P}_2, T'_2)]\} \tag{5}$$

Equation 5 corresponds to a computation in a join node. We define $\mathcal{A}[i, D]$ for $i \in [N] \setminus \{1\}$ and type $D = (Y, Y', \mathcal{P}, T')$ as,

$$\mathcal{A}[i, D] = \min \begin{cases} \min_{v \in Y} \ I_v[i, D] \\ \min\limits_{\substack{uv \in E(G) \\ u,v \in Y}} I_{uv}[i, D] \\ \min_{w \in V(G)} F_w[i, D] \\ J[i, D] \end{cases} \tag{6}$$

For each $i \in [N]$ and each type D, we associate with $A[i, D]$ a subgraph of S. We say that a subgraph F is of type $(Y, Y', \mathcal{P}, T')$, where $\mathcal{P} = \{P_1, \dots P_q\}$ if the following holds.

(a) The number of connected components in F is equal to $|\mathcal{P}| = q$. We can order the connected components $C_1, \dots, C_q$ of F such that $V(C_i) \cap Y = P_i$.

(b) $V(F) \cap T = T'$.

In the following lemma, we show the connection between $A[i, D]$ and a graph of type D.

▶ **Lemma 25.** *Let $i \in [N]$ and D be a type. Furthermore, let $A[i, D]$ be computed by the Equation 6, and have a finite value ℓ. Then there is a subgraph F, of type D, such that* $\mathsf{recdist}(F) \leq \ell$.

The next lemma relates an optimal rectilinear Steiner tree to the values computed for the table $\mathcal{A}[,]$. Using induction on the level of nodes in $\mathcal{T}$ we prove the following.

▶ **Lemma 26.** *Let $\mathcal{T} = (\mathbb{T}, \{X_t\}_{t \in V(\mathbb{T})})$ be a nice tree decomposition of S. For a node t, let X_t be the corresponding bag, $X \subseteq X_t$, $\mathcal{P}$ be a partition of X, V_t be the union of bags in the subtree rooted at t, and $T' = T \cap V_t$. Then $A[i, (X_t, X, \mathcal{P}, T')] \leq c[t, X, \mathcal{P}]$.*

Finally, we describe the subexponential algorithm for Rectilinear Steiner Tree.

▶ **Theorem 27.** Rectilinear Steiner Tree *can be solved in time $2^{\mathcal{O}(\sqrt{n} \log n)} n^{\mathcal{O}(1)}$.*

Description of the Algorithm. We take as input a set T of n terminal points, the Hanan grid G of T and the weight function $\mathsf{recdist}$. Then using Lemma 15 we compute a shortest path RST $\widehat{S}$. By Lemma 18, we know that there is an optimal Steiner tree S_{opt} with $\mathsf{tw}(\widehat{S} \cup S_{\mathsf{opt}}) \leq 41\sqrt{n}$. Based on the shortest path RST $\widehat{S}$, we apply Lemma 24, to enumerate all possible types D of G. We fix an integer $N = 3(|V(G)| + |E(G)|)$ and a terminal t^* in T. For each $i \in [N]$ and each type D, the algorithm computes values $\mathcal{A}[i, D]$, according to Equations 1 and 6. The values in $\mathcal{A}[,]$ are filled in the increasing order of i. Finally, the algorithm outputs $\min_{i \in [N]} \mathcal{A}[i, (\{t^*\}, \{t^*\}, \{\{t^*\}\}, T)]$. The size of the table $\mathcal{A}[,]$ is $N \cdot 2^{\mathcal{O}(\sqrt{n} \log n)}$ and each entry can be filled in time $2^{\mathcal{O}(\sqrt{n} \log n)} n^{\mathcal{O}(1)}$. Thus, the running time of the algorithm is $2^{\mathcal{O}(\sqrt{n} \log n)} n^{\mathcal{O}(1)}$. Using standard back-tracking tricks we can also output an optimal RST. ◀

References

1. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–74, New York, 2007. ACM.
2. Marcus Brazil and Martin Zachariasen. *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications*. Springer, 2015.
3. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer-Verlag, Berlin, 2015.

4 Linda L. Deneen, Gary M. Shute, and Clark D. Thomborson. A probably fast, provably optimal algorithm for rectilinear Steiner trees. *Random Structures Algorithms*, 5(4):535–557, 1994.

5 Stuart E. Dreyfus and Robert A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

6 Bernhard Fuchs, Walter Kern, Daniel Mölle, Stefan Richter, Peter Rossmanith, and Xinhui Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007. doi:10.1007/s00224-007-1324-4.

7 Joseph L. Ganley. Computing optimal rectilinear Steiner trees: a survey and experimental evaluation. *Discrete Appl. Math.*, 90(1-3):161–171, 1999.

8 Joseph L. Ganley and James P. Cohoon. Improved computation of optimal rectilinear Steiner minimal trees. *Internat. J. Comput. Geom. Appl.*, 7(5):457–472, 1997. doi:10.1142/S0218195997000272.

9 M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

10 Qian-Ping Gu and Hisao Tamaki. Improved bounds on the planar branchwidth with respect to the largest grid minor size. *Algorithmica*, 64(3):416–453, 2012. doi:10.1007/s00453-012-9627-5.

11 M. Hanan. On Steiner's problem with rectilinear distance. *SIAM J. Appl. Math.*, 14:255–265, 1966.

12 Frank K Hwang. On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, 30(1):104–114, 1976.

13 Frank K. Hwang, Dana S. Richards, and Pawel Winter. *The Steiner tree problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland Publishing Co., Amsterdam, 1992.

14 Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1812–1830. SIAM, 2014.

15 L. Nastansky, S. M. Selkow, and N. F. Stewart. Cost-minimal trees in directed acyclic graphs. *Z. Operations Res. Ser. A-B*, 18:A59–A67, 1974.

16 Jesper Nederlof. Fast polynomial-space algorithms using inclusion-exclusion. *Algorithmica*, 65(4):868–884, 2013. doi:10.1007/s00453-012-9630-x.

17 Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for Steiner tree on planar graphs. In *Proc. of the 30th International Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 20 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 353–364. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

18 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for Steiner problems on planar and bounded-genus graphs. In *Proc. 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 276–285. IEEE, 2014.

19 Hans Jürgen Prömel and Angelika Steger. *The Steiner Tree Problem*. Advanced Lectures in Mathematics. Friedr. Vieweg & Sohn, Braunschweig, 2002.

20 Weiping Shi and Chen Su. The Rectilinear Steiner Arborescence Problem Is NP-Complete. *SIAM J. Comput.*, 35(3):729–740, 2005. doi:10.1137/S0097539704371353.

21 Clark D Thomborson, Linda L Deneen, and Gary M Shute. Computing a rectilinear steiner minimal tree in $n^{O(\sqrt{n})}$ time. In *Parallel Algorithms and Architectures*, pages 176–183. Springer, 1987.

Random Sampling with Removal*

Bernd Gärtner[1], Johannes Lengler[2], and May Szedlák[3]

1 **Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland**
gaertner@inf.ethz.ch

2 **Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland**
johannes.lengler@inf.ethz.ch

3 **Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland**
may.szedlak@inf.ethz.ch

Abstract

Random sampling is a classical tool in constrained optimization. Under favorable conditions, the optimal solution subject to a small subset of randomly chosen constraints violates only a small subset of the remaining constraints. Here we study the following variant that we call random sampling with removal: suppose that after sampling the subset, we remove a fixed number of constraints from the sample, according to an arbitrary rule. Is it still true that the optimal solution of the reduced sample violates only a small subset of the constraints?

The question naturally comes up in situations where the solution subject to the sampled constraints is used as an approximate solution to the original problem. In this case, it makes sense to improve cost and volatility of the sample solution by removing some of the constraints that appear most restricting. At the same time, the approximation quality (measured in terms of violated constraints) should remain high.

We study random sampling with removal in a generalized, completely abstract setting where we assign to each subset R of the constraints an arbitrary set $V(R)$ of constraints disjoint from R; in applications, $V(R)$ corresponds to the constraints violated by the optimal solution subject to only the constraints in R. Furthermore, our results are parametrized by the dimension δ, i.e., we assume that every set R has a subset B of size at most δ with the same set of violated constraints. This is the first time this generalized setting is studied.

In this setting, we prove matching upper and lower bounds for the expected number of constraints violated by a random sample, after the removal of k elements. For a large range of values of k, the new upper bounds improve the previously best bounds for LP-type problems, which moreover had only been known in special cases. We show that this bound on special LP-type problems, can be derived in the much more general setting of violator spaces, and with very elementary proofs.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases LP-type problem, violator space, random sampling, sampling with removal

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.40

* Research supported by the Swiss National Science Foundation (SNF Project 200021_150055/1).

© Bernd Gärtner, Johannes Lengler, and May Szedlák;
licensed under Creative Commons License CC-BY

1 Introduction

On a high level, random sampling can be described as an efficient way of learning something about a problem, by first solving a subproblem of much smaller size. A classical example is the problem of finding the smallest element in a *sorted compact list* [4, Problem 11-3]. Such a list stores its elements in an array, but in arbitrary order. Additional pointers are used to link each element to the next smaller one in the list. Given a sorted compact list of size n, the smallest element can be found in expected time time $O(\sqrt{n})$ as follows: sample a set of $\lfloor\sqrt{n}\rfloor$ array elements at random. Starting from their minimum, follow the predecessor pointers to the global minimum. The key fact is that the expected number of pointers to be followed is bounded by $\sqrt{n}$, and this yields the expected runtime.

On an abstract level, the situation can be modeled as follows. Let H be a set of size n that we can think of as the set of constraints in an optimization problem, for example the elements in a sorted compact list. Let $V : 2^H \to 2^H$ be a function that assigns to each subset $R \subseteq H$ of constraints a set $V(R) \subseteq H \setminus R$. We can think of $V(R)$ as the set of constraints violated by the optimal solution subject to only the constraints in R. This abstract setting (H, V), denoted by *consistent space*, is introduced here for the first time (for a formal definition see Definition 2.3 below.) In the sorted compact list example, $V(R)$ is the set of elements that are smaller than the minimum of R.

In this setting, the above "key fact" is a concrete answer to the following abstract question: Suppose that we sample a set $R \subseteq H$ of size $r \leq n$ uniformly at random. What can we say about the quantity v_r, the expected size of $V(R)$? What are conditions on V under which v_r is small?

The main workhorse in this context is the *Sampling Lemma* [8]. It states that $v_r = \frac{n-r}{r+1} \cdot x_{r+1}$, where x_r is the expected size of $X(R) = \{h \in R : h \in V(R \setminus \{h\})\}$. In other words, $h \in X(R)$ is a constraint that is not automatically satisfied if the problem is solved without enforcing it. In the sorted compact list example, every nonempty set R has one such "extreme" constraint, namely its minimum. Consequently, we have $x_{r+1} = 1$, and hence $v_r = (n-r)/(r+1)$. With $r = \lfloor\sqrt{n}\rfloor$, $v_r < \sqrt{n}$ follows.

The Sampling Lemma has many other applications in computational geometry when x_{r+1} can be bounded; in a number of relevant cases, we do not only know the expected value v_r but the complete probability distribution $p_\ell = \Pr[|V(R)| = \ell], \ell \leq n$, or tail estimates for $|V(R)|$ [8].

In this paper, we address the following more general question in the abstract setting: Suppose that we sample a set $R \subseteq H$ of size $r \leq n$ uniformly at random, but then we remove a subset $K_R \subseteq R$ of a fixed size k, according to an arbitrary but fixed rule. What can we still say about the expected size of $V(R \setminus K_R)$? If K_R is a random subset of R, the expectation is v_{r-k}, but if K_R is chosen by another (deterministic) rule, then $R \setminus K_R$ is no longer a uniformly random subset, and the Sampling Lemma does not apply.

Our work is originally motivated by *chance-constrained optimization*. In this setting, we have a probability distribution over a (possibly infinite) set of constraints. The goal is to compute a *sufficiently feasible* solution, one that satisfies a randomly chosen constraint with high probability. Such a solution can be obtained by optimizing over a finite sample of constraints drawn from the distribution [2]. Here, a-posteriori removal of constraints is shown to yield a tradeoff between solution quality and violation probability [3]. We are trying to understand the combinatorial essence of this tradeoff.

Intuitively, one would think that if k is constant, the change in the expected number of violated constraints under the removal of k constraints is small. This intuition was proved

to be correct if the pair (H, V) is induced by a *nondegenerate LP-type problem*, where the results are parametrized by the dimension δ [5], (for definition of dimension see Definition 4 below). LP-type problems have been introduced and analyzed by Matoušek, Sharir and Welzl as a combinatorial framework that encompasses linear programming and other geometric optimization problems [11, 9]. The quantitative result was that under removal of k elements, the expected number of violated constraints increases by a factor of δ^k at most, which is constant if both δ and k are constant. It was left open whether this factor can be improved for interesting sample sizes (for very specific and rather irrelevant values of δ, r, k, it was shown to be best possible).

In this paper, we improve over the results in [5] in several respects. In Section 3, Theorem 10 we show that the increase factor δ^k can be replaced by $\log n + k$, which is a vast improvement for a large range of values of k. Moreover, the new bound neither requires the machinery of LP-type problems, nor nondegeneracy. It holds in the completely abstract setting of consistent spaces considered above. In this setting, we can also show that the bound is best possible for all sample sizes of the form $r = n^\alpha, 0 < \alpha < 1$ (see Section 5). We also show that this bound is best possible for violator spaces, in the case where $k = \Omega(\delta \log n)$. In general, for violator spaces the gap to the lower bound is $\log n$.

Hence, if anything can be gained over the new bound, additional properties of the violator function V have to be used. Indeed, for small values of k, the increase factor in [5] is better than our new bound for nondegenerate LP-type problems, and most notably, it does not depend on the problem size n. We show in Section 4, Theorem 14 that the same factor can be derived under the much weaker conditions of a *nondegenerate violator space*, and with a much simpler proof, based on a "removal version" of the Sampling Lemma. Furthermore the proof of [5] is given for a specific rule to remove k, whereas our proof works for any rule.

Intuitively, violator spaces are LP-type problems without objective function, and they were introduced to show that many combinatorial properties of LP-type problems and algorithms for LP-type problems do not require the objective function at all [6, 1].

In Section 6, Theorem 18 we show tight upper and lower bounds for the case $\delta = 1$, which shows that the improved bound for nondegenerate violater spaces is best possible for *all* violator spaces. For smaller (and in particular constant) k, the quest for the best bound on the increase factor remains open. In particular, it is not clear whether the exponential growth in k actually happens.

What also remains open is the role of nondegeneracy. In many geometric situations, nondegeneracy can be attained through symbolic perturbation and can therefore be assumed without loss of generality for most purposes. In the abstract setting, this is not necessarily true, as there are examples of LP-type problems for which any "combinatorial perturbation" increases the dimension [10].

2 Basics and Definitions

Throughout the paper we will work with three combinatorial concepts, the LP-type problem, the violator space and the consistent space. The LP-type problem was first introduced by Sharir and Welzl [11], the generalized concept of violator spaces by Gärtner, Matoušek, Rüst, and Škovroň [6]. In this paper we introduce an even more general concept of consistent spaces.

2.1 LP-type Problems

▶ **Definition 1.** An *LP-type* problem is a triple $\mathcal{P} = (H, \Omega, \omega)$ that satisfies the following. H is a finite set (the constraints), Ω a totally ordered set with a smallest element $-\infty$ and $\omega : 2^H \to \Omega$ a function that assigns an objective function value to $G \subseteq H$, such that $\omega(\emptyset) = -\infty$. For all $F \subseteq G \subseteq H$ and $h \in H$, the following hold.

1. $\omega(F) \leq \omega(G)$, and
2. If $\omega(F) = \omega(G) > -\infty$, then $\omega(G \cup \{h\}) > \omega(G) \Rightarrow \omega(F \cup \{h\}) > \omega(F)$.

The first condition is called *monotonicity*, the second *locality*.

Observe that using locality, by simple induction one can show that if $\omega(F \cup \{h\}) = \omega(F) > -\infty$ for all $h \in G \setminus F$, then $\omega(F) = \omega(G)$.

The classic example of an LP-type problem is the problem of computing the smallest enclosing ball of a finite set of points P in $\mathbb{R}^d$. Let us denote this problem by SEB. We can write this as an LP-type problem by setting $H = P$ and $\Omega = \mathbb{R} \cup \{-\infty\}$. For $G \subseteq H$, $\omega(G)$ is defined as the radius of the smallest enclosing ball of G, with the convention that the radius of the empty set is $-\infty$. Since the smallest enclosing ball of a nonempty set of points exists and is unique [13], ω is well defined.

Monotonicity is clear. To see locality, observe that for $F \subseteq G$, $\omega(F) = \omega(G)$ means that both F and G have the same smallest enclosing ball. If $\omega(G \cup \{h\}) > \omega(G)$, then h is outside this ball, so $\omega(F \cup \{h\}) > \omega(F)$.

▶ **Definition 2.** A constraint $h \in H \setminus G$ is violated by G if $\omega(G \cup \{h\}) > \omega(G)$. We denote the set of violated constraints by $V(G)$.

For SEB, the violated constraints of G are exactly the points lying outside the smallest enclosing ball of G.

2.2 Violator Spaces

Intuitively a violator space is an LP-type problem without an objective function. The advantage is that many things one can prove about LP-type problems do not require the concept of order.

▶ **Definition 3.** A *violator space* is a pair (H, V), $|H| = n$ finite and V a function $2^H \to 2^H$ such that the following is satisfied for all $F \subseteq G \subseteq H$.

1. $G \cap V(G) = \emptyset$.
2. If $G \cap V(F) = \emptyset$, then $V(G) = V(F)$.

The first condition is called *consistency*, the second *locality*.

Observe that the locality condition implies that if $E \subseteq F \subseteq G$ and $V(E) = V(G)$, then $V(E) = V(F) = V(G)$.

The notion of a violator space is more general than the LP-type problem, since every LP-type problem (H, Ω, ω) can naturally be converted into a violator space through $V(R) = \{x \in H \setminus R \mid \omega(R \cup \{x\}) \neq \omega(R)\}$, for $R \subseteq H$. On the other hand, not every violator space can be converted into an LP-type problem. Any unique sink orientation (USO) [12] of a cube or the grid [7] corresponds to a violator space, but not to an LP-type problem in general [6].

▶ **Definition 4.** Let (H, V) be a violator space.

1. $B \subseteq H$ is called a *basis* in (H, V), if for all $F \subsetneq B$, $B \cap V(F) \neq \emptyset$ (or equivalently, $V(F) \neq V(B)$).

2. A basis of $G \subseteq H$ is an inclusion-minimal subset $B \subseteq G$ such that $V(B) = V(G)$. (In particular, a basis of G is a basis in (H, V), and every basis in (H, V) is a basis of itself.)
3. The *combinatorial dimension* of (H, V), denoted $\delta := \delta(H, V)$ is defined by the size of the largest basis in (H, V).

Again, let us illustrate this on SEB. A basis of G is a minimal subset of points with the same enclosing ball of G. In particular all points of the basis are on the ball's boundary. In d-dimensional space, the combinatorial dimension of any SEB-instance is at most $d+1$, since any enclosing ball can be defined by at most $d+1$ points on its boundary. However, a basis can be smaller than the combinatorial dimension, and a point set can have more than one basis: in $\mathbb{R}^2$ the set of four corners of a square has two bases, the two pairs of diagonally opposite points.

▶ **Definition 5.** The set of *extreme constraints* $X(G) \subseteq G$ is defined by

$$r \in X(G) \Leftrightarrow r \in V(G \setminus \{r\}).$$

In the SEB case, h is extreme in G if its removal allows for a smaller enclosing ball. Therefore h is necessarily on the boundary of the smallest enclosing ball, but this is not sufficient. For the case $\mathbb{R}^2$, if G consists of the four corners of a square, then G has no extreme point.

It is not hard to see that $X(G)$ is the intersection of all bases of G, hence $|X(G)| \leq \delta$. To bound the expected number of violators, the following result from [8] is known.

▶ **Lemma 6** (Sampling Lemma). *Let (H, V) be a violator space with combinatorial dimension δ. Let $R \subseteq H$ a u.a.r. set of size r, $v_r = \mathbb{E}[|V(R)|]$ and $x_r = \mathbb{E}[|X(R)|]$. Then*

$$v_r = \frac{n-r}{r+1} \cdot x_{r+1} \leq \frac{n-r}{r+1} \cdot \delta.$$

The Sampling Lemma can be used to argue that v_r is small if the expected number x_{r+1} of extreme constraints of a random sample of size $r+1$ is small.

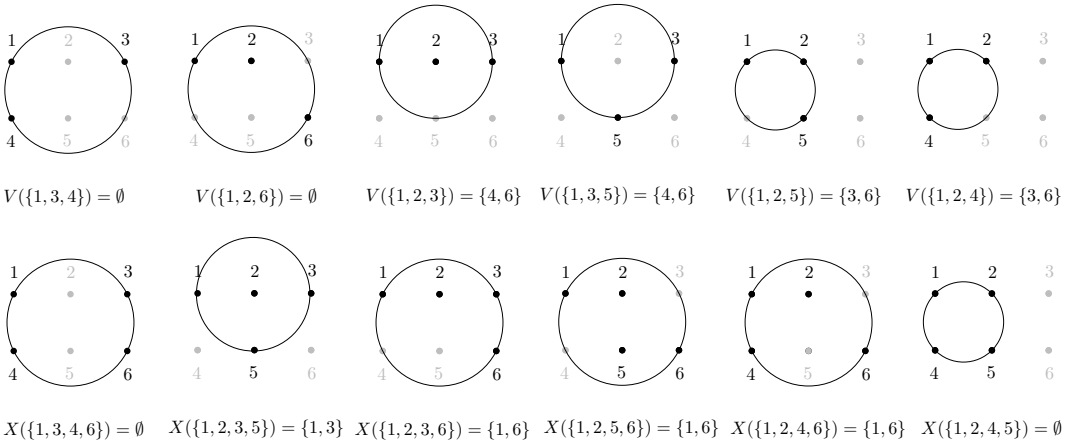
In SEB case in $\mathbb{R}^d$, one can use Helly's theorem to show that every set has at most $d+1$ extreme points and therefore $v_r \leq \frac{n-r}{r+1} \cdot (d+1)$. If $d = 2$, then the smallest enclosing ball of a random sample of size $\sqrt{n}$ has in expectation at most $3\sqrt{n}$ points outside.

The figure below shows an example of SEB, on the 2×3 regular grid, in particular $n = 6$. We fix $r = 3$ and therefore look at the violators of sets of size three and extreme points of sets of size four, which are all depicted (up to symmetry). It is not hard to see that eight sets of size three have no violators (corresponding to the first two cases in the figure), while all others have two violators. This implies $v_3 = 1.2$. Looking at the extreme points the sets $\{1,3,4,6\}, \{1,2,4,5\}$ and $\{2,3,5,6\}$ have no extreme elements and all other sets of size four have exactly two extreme elements. Therefore $v_3 = \frac{6-3}{3+1} x_4 = \frac{3}{4} \cdot 1.6 = 1.2$. For six points in general position $v_3 = \frac{3}{4} \cdot 3 = 2.25$.

▶ **Definition 7.** A violator space (H, V) is called *nondegenerate* if every set $G \subseteq H$ has a unique basis.

Note that SEB it not nondegenerate, since as mentioned in $\mathbb{R}^2$, four points on a square have two bases. For a nondegenerate example see the d-smallest number violator space (Definition 16 below).

It is shown in [10] that in abstract settings one cannot without loss of generality assume nondegeneracy. This is shown by constructing LP-type problems of dimension δ for infinitely many integers δ such that in oder to remove degeneracies, one has to increase the dimension to at least 2δ.



2.3 Consistent Spaces

We now move on to the even more general concept of consistent spaces.

▶ **Definition 8.** A *consistent space* is a pair (H, V), $|H| = n$ finite and V a function $2^H \to 2^H$ such that the following is satisfied for all $G \subseteq H$.

1. $G \cap V(G) = \emptyset$.

Hence a consistent space is a violator space without the locality condition. The basis, combinatorial dimension and extreme constraints of a consistent space can be defined equivalently as in the violator space.

In consistent spaces the first equality $v_r = \frac{n-r}{r+1} \cdot x_{r+1}$ of the Sampling Lemma 6 still holds. However, in general it does not hold that $|X(R)| \leq \delta$ for all $R \subseteq H$. We give an example of a consistent space of dimension 1 such that for some $R \subseteq H$ each element is extreme. Let $R = \{1, 2, \ldots, 2m\} \subseteq n$ be of even size. For $i \in [m]$ let $V(i) = \{i + m\}$ and $V(i + m) = \{i\}$. For every $x \in R$ define $V(R \setminus \{x\}) = \{x\}$ and for all other sets define the violators as the empty set. Then this is a consistent space. By definition it also follows that every element in R is extreme. For $x \in R$ it holds that $V(R \setminus \{x\}) = \{x\} = V(x + m \mod 2m)$ and since $x + m \mod 2m \in R$ it follows that the combinatorial dimension is 1.

2.4 Sampling with Removal

As already introduced in [5] for LP-type problems, we are interested in sampling with removal. We define the concept here for the most general case of consistent spaces. All results will then naturally extend for violator spaces and LP-type problems. Suppose we sample uniformly at random $R \subseteq H$ of size r. By some fixed rule P_k, we remove $k < r$ elements of R and obtain a set R_{P_k} of size $r - k$. We define $V_{P_k}(R) := V(R_{P_k})$. Note that in general (H, V_{P_k}) is not a consistent space. We are interested in $\mathbb{E}[|V_{P_k}(R)|]$, for which we will give different bounds in the next two chapters.

Before proceeding to the bounds, we discuss some possible rules for the removal of the k elements. If k elements are removed uniformly at random from R, then $\mathbb{E}[|V_{P_k}(R)|] = v_{r-k}$. Another way to remove k elements is to maximize the number of violators after the removal. In the case of LP-type problems it is intuitive to remove in such a way that the objective function is minimized [5]. For this last rule [5] establishes a bound of $\mathbb{E}[|V_{P_k}(R)|] = O(\frac{n-r}{r+1}\delta^{k+1})$ for fixed k, if the LP-type problem is nondegenerate.

3 An Upper Bound for Consistent Spaces

The main result of this section is Theorem 10, where we show an upper bound on $\mathbb{E}[|V_{P_k}(R)|]$ for consistent spaces. In Lemma 15 and Lemma 17 we show asymptotically matching lower bounds for most relevant values of r.

We start with the following technical lemma.

▶ **Lemma 9.** *Let (H, V), with $|H| = n$, a consistent space of dimension δ and P_k some fixed rule to remove k points. Let R be u.a.r. from all sets of size r, c some suitable constant (e.g. $c = 33$), and $x = c \cdot \max\left\{\frac{n}{r}\delta \log n, \frac{n}{r}k\right\}$. We assume $r = o(n)$, $\delta = o(r)$, $k \le \frac{r}{c}$, and $r + x \le n$. Then*

1. *$Pr[|V_{P_k}(R)| \ge x] \le \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta} \frac{\binom{n}{\alpha}}{\binom{n}{r}}\binom{x}{i}\binom{n-x-\alpha}{r-\alpha-i}$.*
2. *Furthermore for all $0 \le \alpha \le \delta$ and $0 \le i \le k$,*

$$\frac{\binom{n}{\alpha}}{\binom{n}{r}}\binom{x}{i}\binom{n-x-\alpha}{r-\alpha-i} \le n^{-3}.$$

Proof of Lemma 9. For $R \subseteq H$ define $\mathcal{B}_{P_k}(R) := \{B \subseteq H \mid B \text{ basis of } R_{P_k}\} \neq \emptyset$, the set of all bases of R_{P_k}. Let $R \subseteq H$ uniformly at random of size r. Then

$$\begin{aligned}\Pr[|V_{P_k}(R)| \ge x] &= \Pr[\exists B \in \mathcal{B}_{P_k}(R) : |V(B)| \ge x] \\ &\le \Pr[\exists B \subseteq H : |V(B)| \ge x \wedge |B| \le \delta \wedge B \subseteq R \wedge V(B) = V_{P_k}(R)] \\ &\le \Pr[\exists B \subseteq H : |V(B)| \ge x \wedge |B| \le \delta \wedge B \subseteq R \wedge |V(B) \cap R| \le k].\end{aligned}$$

The last inequality follows since by $V_{P_k}(R) = V(B)$ it follows that $R_{P_k} \cap V(B) = \emptyset$ and because R_{P_k} is obtained from R by removing k elements $|R \cap V(B)| \le k$. Using union bound we obtain

$$\begin{aligned}\Pr[|V_{P_k}(R)| \ge x] &\le \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta} \Pr[\exists B \subseteq H : |V(B)| \ge x \wedge |B| = \alpha \wedge B \subseteq R \wedge |V(B) \cap R| = i] \\ &\le \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta} \sum_{\substack{B \in \binom{n}{\alpha} \\ |V(B)| \ge x}} \Pr[B \subseteq R \wedge |V(B) \cap R| = i] \\ &= \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta} \sum_{\substack{B \in \binom{n}{\alpha} \\ |V(B)| \ge x}} \frac{1}{\binom{n}{r}} \underbrace{\binom{|V(B)|}{i}\binom{n-|V(B)|-\alpha}{r-\alpha-i}}_{(*)}.\end{aligned}$$

We now show that $(*)$ is maximized if $|V(B)| = x$, which concludes the proof. For $i = 0$ this is obvious, hence assume $i \ge 1$.

The claim follows since by basic calculations we can show that for all $y \ge \frac{i(n-\alpha)-r+i+\alpha}{r-\alpha}$,

$$\binom{y}{i}\binom{n-y-\alpha}{r-\alpha-i} \ge \binom{y+1}{i}\binom{n-(y+1)-\alpha}{r-\alpha-i}.$$

Using that $\alpha = o(r)$ and $r, i, \alpha = o(n)$ we get

$$x > (1+o(1))\frac{n}{r}k \ge (1+o(1))\frac{n}{r}i = \frac{i(n-\alpha)-r+i+\alpha}{r-\alpha},$$

and the first part follows.

For the second part again the case of $i = 0$ is easy, hence assume that $i \geq 1$.

$$\begin{aligned}
&\frac{\binom{n}{\alpha}}{\binom{n}{r}} \underbrace{\binom{x}{i}}_{\leq \left(\frac{xe}{i}\right)^i} \binom{n-x-\alpha}{r-\alpha-i} \\
&\qquad \leq \left(\frac{xe}{i}\right)^i \cdot \frac{n!}{\underbrace{\alpha!}_{\geq 1}(n-\alpha)!} \cdot \frac{r!(n-r)!}{n!} \cdot \frac{(n-x-\alpha)!}{(r-\alpha-i)!(n-x-r+i)!} \\
&\qquad \leq \left(\frac{xe}{i}\right)^i \cdot \underbrace{\frac{r \cdots (r-\alpha-i+1)}{(n-\alpha)\cdots(n-\alpha-i+1)}}_{\leq r^\alpha \left(\frac{r}{n}\right)^i} \cdot \frac{(n-r)!}{(n-\alpha-i)!} \cdot \frac{(n-x-\alpha)!}{(n-x-r+i)!} \\
&\qquad \leq \left(\frac{xe}{i}\right)^i \cdot r^\alpha \left(\frac{r}{n}\right)^i \underbrace{\frac{(n-x-\alpha)\cdots(n-x-r+i+1)}{(n-\alpha-i)\cdots(n-r+1)}}_{\leq \left(1-(1+o(1))\frac{x}{n}\right)^{r-\alpha-i} \leq e^{-\frac{x}{2n}r}} \\
&\qquad \leq \left(\frac{xe}{i}\right)^i \cdot r^\alpha \left(\frac{r}{n}\right)^i e^{-\frac{x}{2n}r}.
\end{aligned}$$

Now in the first case we assume that $k \leq \delta \log n$, hence $x = c\frac{n}{r}\delta \log n$. It follows that

$$\begin{aligned}
\frac{\binom{n}{\alpha}}{\binom{n}{r}}\binom{x}{i}\binom{n-x-\alpha}{r-\alpha-i} &\leq \left(\frac{c\frac{n}{r}\delta \log ne}{i}\right)^i \cdot r^\alpha \left(\frac{r}{n}\right)^i e^{-\frac{c}{2}\delta \log n} \leq \left(\frac{c\delta \log ne}{i}\right)^i \cdot r^\alpha e^{-\frac{c}{2}\delta \log n} \\
&= e^{\overbrace{i \log c}^{\leq \delta \log n \log c} + i \log \delta + i \log\log n + \overbrace{i}^{\leq \delta \log n} - i \log i + \overbrace{\alpha \log r}^{\leq \delta \log n} - \frac{c}{2}\delta \log n} \\
&\leq e^{i \log \delta + i \log\log n - i \log i - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n}.
\end{aligned}$$

To show the claim it remains to show that

$$i \log \delta + i \log\log n - i \log i - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n \leq -3 \log n.$$

Since $i \leq k \leq \delta \log n$ we can write $i = \beta\delta \log n$ for some $\beta \in (0, 1]$. Then

$$\begin{aligned}
&i \log \delta + i \log\log n - i \log i - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n \\
&\qquad = \beta\delta \log n \log \delta + \beta\delta \log n \log\log n - \beta\delta \log n(\log \beta + \log \delta + \log\log n) \\
&\qquad\quad - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n \\
&\qquad = -\beta \log \beta\delta \log n - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n.
\end{aligned}$$

It remains to bound $\beta \log \beta$ from below. By taking the derivative we observe that $\beta \log \beta$ attains its minimum when $\beta = \frac{1}{e}$ and hence $\beta \log \beta \geq -\frac{1}{e}$. It therefore follows that

$$\begin{aligned}
&i \log \delta + i \log\log n - i \log i - \left(\frac{c}{2} - 2 - \log c\right)\delta \log n \\
&\qquad \leq -\left(\frac{c}{2} - 2 - \log c - \frac{1}{e}\right)\delta \log n \leq -3 \log n \text{ for } c \geq 33.
\end{aligned}$$

In the second case $k \geq \delta \log n$, hence $x = c \cdot \frac{n}{r} \cdot k$. Again for $i \geq 1$ it follows that

$$\frac{\binom{n}{\alpha}}{\binom{n}{r}}\binom{x}{i}\binom{n-x-\alpha}{r-\alpha-i} \leq \left(\frac{xe}{i}\right)^i \cdot r^\alpha \left(\frac{r}{n}\right)^i e^{-\frac{x}{2n}r} = \left(c\frac{n}{r}k\frac{e}{i}\right)^i r^\alpha \left(\frac{r}{n}\right)^i e^{-\frac{c}{2}k}$$
$$= e^{\overbrace{i\log c}^{\leq k\log c} + i\log k + \overbrace{i}^{\leq k} - i\log i + \overbrace{\alpha\log r}^{\leq \delta\log n \leq k} - \frac{c}{2}k}$$
$$\leq e^{i(\log k - \log i) - (\frac{c}{2} - 2 - \log c)k}$$

Now as before it suffices to show that

$$i(\log k - \log i) - \left(\frac{c}{2} - 2 - \log c\right)k \leq -3\log n.$$

Since $i \leq k$ we can write $i = \beta k$ for some $\beta \in (0, 1]$. Then

$$i(\log k - \log i) - \left(\frac{c}{2} - 2 - \log c\right)k$$
$$= \beta k(\log k - \log k - \log \beta) - \left(\frac{c}{2} - 2 - \log c\right)k$$
$$\leq -\left(\frac{c}{2} - 2 - \log c - \frac{1}{e}\right)\underbrace{k}_{\geq \delta \log n} \leq -3\log n \text{ for } c \geq 33,$$

where we again used that $\beta \log \beta \geq -\frac{1}{e}$. ◀

▶ **Theorem 10.** *Let (H, V), with $|H| = n$, a* consistent space *of dimension δ and P_k some fixed rule to remove k points. Let $R \subseteq H$ u.a.r. of all sets of size r, for some $r \leq n$. Then*

$$\mathbb{E}[|V_{P_k}(R)|] \leq c \cdot \max\left\{\frac{n}{r}\delta\log n, \frac{n}{r}k\right\} =: x,$$

where c is some suitable constant (e.g. $c = 33$).

Observe that compared to Lemma 6, for most relevant r (e.g. $r = n^\beta$, $\beta \in (0, 1)$) and $k = o(\delta \log n)$, there is an additional $\log n$ term.

Proof of Theorem 10. We may assume $r = o(n)$, $\delta = o(r)$, $k \leq \frac{r}{c}$, and $r + x \leq n$, since otherwise the bound is trivial and there is nothing to prove. By definition of expectation

$$\mathbb{E}[|V_{P_k}(R)|] \leq \Pr[|V_{P_k}(R)| < x] \cdot (x-1) + \Pr[|V_{P_k}(R)| \geq x] \cdot n$$
$$\leq x - 1 + \Pr[|V_{P_k}(R)| \geq x] \cdot n.$$

We will now show that $\Pr[|V_{P_k}(R)| \geq x] \leq n^{-1}$ which will conclude the proof. By Lemma 9,

$$\Pr[|V_{P_k}(R)| \geq x] \leq \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta}\frac{\binom{n}{\alpha}}{\binom{n}{r}}\binom{x}{i}\binom{n-x-\alpha}{r-\alpha-i} \leq \sum_{i=0}^{k}\sum_{\alpha=0}^{\delta} n^{-3} \leq n^{-1}.$$

as desired. ◀

4 An Upper Bound for Violator Spaces

In this section we give an upper bound on $\mathbb{E}[|V_{P_k}(R)|]$ for nondegenerate violator spaces, Theorem 14. This is an improvement of the bound given in [5], which stated the same bound for nondegenerate LP-type problems and the specific rule P_k to minimize the objective function after removal. Matching lower bounds for special cases are known [5]. The bound in Theorem 14 is stronger than the (more general) bound in Theorem 10 if δ and k are very small, e.g., if $\delta^k < \log n$; for large δ and k, Theorem 10 is stronger.

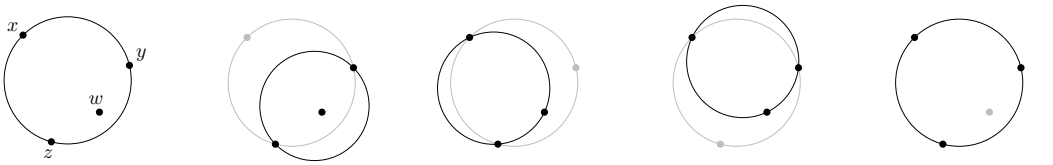
4.1 Extreme Constraints after Removal

Let (H, V) be a violator space of combinatorial dimension δ. In particular, every set has at most δ extreme constraints. For a given natural number k, we want to understand the following quantity:

$$\Delta_k(H, V) := \max_{R \subseteq H} |\{X(R \setminus K) : K \subseteq R, |K| = k\}|.$$

In other words, how many sets of extreme constraints can we get by removing k elements from some set R?

The figure below shows an instance of SEB with $R = \{x, y, z, w\}$, where by removing one point from R we can get four different sets of extreme constraints ($\{x, y, z\}, \{y, z\}, \{x, z, w\}$ and $\{x, y, w\}$. Therefore we conclude that $\Delta_1(H, V) \geq 4$. We will see below that for nondegenerate violator spaces $\Delta_1(H, V) \leq \delta + 1$, so this bound is actually tight.



$X(R) = \{x, y, z\}$ $X(R \setminus \{x\}) = \{y, z\}$ $X(R \setminus \{y\}) = \{x, z, w\}$ $X(R \setminus \{z\}) = \{x, y, w\}$ $X(R \setminus \{w\}) = X(R)$

▶ **Lemma 11.** *Let (H, V) be a nondegenerate violator space. For $R \subseteq H$ let B_R be its unique basis. Then $X(R) = B_R$. Furthermore for $x \in R \setminus X(R)$ we have that $X(R) = X(R \setminus \{x\})$.*

Proof. Let $x \in R \setminus X(R)$. Then by definition $x \notin V(R \setminus \{x\})$ and therefore by locality $V(R) = V(R \setminus \{x\})$. Now $V(R \setminus \{x\}) = V(B_{R\setminus\{x\}})$. By nondegeneracy, $B_R = B_{R\setminus\{x\}}$ and therefore $x \notin B_R$. For the other direction assume $x \in X(R)$. Then $x \in V(R \setminus \{x\})$. If $x \notin B_R$, then by locality $V(B_R) = V(R) = V(R \setminus \{x\})$, which is a contradiction.

The second part follows since by nondegeneracy $X(R) = B_R = B_{R\setminus\{x\}} = X(R \setminus \{x\})$. ◀

Let's bound the easy cases of $\Delta_k(H, V)$ first. We obviously have $\Delta_0(H, V) = 1$ for any violator space (H, V). Moreover for (H, V) nondegenerate we have, $\Delta_1(H, V) \leq \delta + 1$. Indeed, if we remove a non-extreme element x from R, by Lemma 11 we end up with the same set $X(R \setminus \{x\}) = X(R)$ of extreme elements, so only in at most δ cases, we will get a different set. Note that in general this bound does not hold. Consider SEB and assume we have four points on a square and one point in the middle. It is not hard to see that for each point its removal generates a different set of extreme points. Hence $\Delta_1(H, V) \geq 5 > \delta + 1 = 4$.

▶ **Lemma 12.** *Let (H, V) be a nondegenerate violator space. Then $\Delta_k(H, V) \leq \sum_{i=0}^{k} \delta^i$.*

Proof. Let us fix R and a set $K \subseteq R, |K| = k$ to be removed. We claim that we can order the elements of K as $e_1, \ldots, e_k$ such that for some $\ell \in \{0, \ldots, k\}$,

$$e_i \in X(R \setminus \{e_1, \ldots, e_{i-1}\}), \quad i \leq \ell, \text{ and } X(R \setminus K) = X(R \setminus \{e_1, \ldots, e_\ell\}).$$

Indeed, we can do this greedily: as long as we can remove an extreme element from the current set, we do so. At some points, all elements that remain to be removed are non-extreme, and by repeated use of Lemma 11 at this point the removal of all of them does not change the extreme elements anymore.

It follows that all sets $X(R \setminus K)$ can be obtained from R by repeatedly removing an extreme element from the current set, up to k times. In the first round, we therefore have at most δ choices, and for each of them, we have at most δ choices in the second round, and so on. The bound follows. ◀

4.2 Sampling Lemma after Removal

Let (H, V) be a violator space. For $R \subseteq H$ and a natural number k, we define the following two quantities.

$$\begin{aligned} V_k(R) &= \{x \in H \setminus R : x \in V(R \setminus K) \text{ for some } K \subseteq R, |K| = k\}, \\ X_k(R) &= \{x \in R : x \in X(R \setminus K) \text{ for some } K \subseteq R, |K| = k\}. \end{aligned}$$

Clearly, $V(R) = V_0(R)$ and $X(R) = X_0(R)$.

Furthermore, we let $v_{r,k}$ denote the expected size of $V_k(R)$ over a randomly chosen set of size r. Similarly, $x_{r,k}$ is the expected size of $X_k(R)$.

▶ **Lemma 13.** *[Sampling Lemma after Removal]*

$$v_{r,k} = \frac{n-r}{r+1} x_{r+1,k}.$$

Proof. This goes like for the "normal" Sampling Lemma 6 [8]. We define a bipartite graph on the vertex set $\binom{H}{r} \cup \binom{H}{r+1}$, where we connect R and $R \cup \{x\}$ with an edge if and only if $x \in V_k(R)$. Let $x \in H \setminus R$. We have the following equivalences:

$$\begin{aligned} x \in V_k(R) &\Leftrightarrow x \in V(R \setminus K) \text{ for some } K \subseteq R, |K| = k \\ &\Leftrightarrow x \in X((R \setminus K) \cup \{x\}) \text{ for some } K \subseteq R, |K| = k \\ &\Leftrightarrow x \in X((R \cup \{x\}) \setminus K) \text{ for some } K \subseteq R, |K| = k,\ x \notin K \\ &\Leftrightarrow x \in X((R \cup \{x\}) \setminus K) \text{ for some } K \subseteq R \cup \{x\}, |K| = k,\ x \notin K \\ &\Leftrightarrow x \in X((R \cup \{x\}) \setminus K) \text{ for some } K \subseteq R \cup \{x\}, |K| = k \\ &\Leftrightarrow x \in X_k(R \cup \{x\}), \end{aligned}$$

where in the fifth step we used that $x \in X((R \cup \{x\}) \setminus K)$ can only occur if $x \in (R \cup \{x\}) \setminus K$.

So we can also define the graph as having an edge between R and $R \cup \{x\}$ if and only if $x \in X_k(R \cup \{x\})$. Since the sum of the degrees of the vertices in $\binom{H}{r}$ is the same as the sum of degrees of the vertices in $\binom{H}{r+1}$, we get

$$\binom{n}{r} v_{r,k} = \sum_{a \in \binom{H}{r}} \deg(a) = \sum_{b \in \binom{H}{r+1}} \deg(b) = \binom{n}{r+1} x_{r,k},$$

which is the claimed result. ◀

Note that as in the "normal" Sampling Lemma, the result holds as well for consistent spaces.

4.3 Violators after Removal

Suppose we sample R at random, and then remove an arbitrary set of k elements K_R according to some fixed rule P_k, and obtain the set $R_{P_k} = R \setminus K_R$. The expected number of violators of R_{P_k} is bounded by $v_{r,k} + k$. This follows since $v_{r,k}$ counts the expected number of violators in $H \setminus R$ that we can possibly get by removing *any* set of k elements and the removed points K_R can also be in $V(R_{P_k})$. Therefore $E[|V_{P_k}(R)|] \leq v_{r,k} + k$.

▶ **Theorem 14.** *Let (H, V) be a nondegenerate violator space of dimension δ, and let R be sampled u.a.r. from all subsets of H of size r. Let P_k be a fixed rule to remove k elements from the random sample. Then*

$$\mathbb{E}[|V_{P_k}(R)|] \leq v_{r,k} + k \leq \sum_{i=1}^{k+1} \delta^i \cdot \frac{n-r}{r+1} + k.$$

Proof. By Lemma 13, we need to bound $x_{r,k}$. To this end, we show that for all R,

$$|X_k(R)| \leq \sum_{i=1}^{k+1} \delta^i.$$

This holds, since by Lemma 12, at most $\sum_{i=0}^{k} \delta^i$ many *sets* of extreme elements can be obtained by removing k elements from R, and each of these sets has at most δ elements. ◀

By [5, Section 7.2], there exists an LP-type problem and a rule P_k, such that $|X_k(R)| = \Theta(\delta^{k+1})$, for $|R| = n - 1$. However, the behavior of the bound is unknown for general r. To get better bounds on the expectation $x_{r,k}$, other methods need to be applied.

5 Matching Lower Bounds for Consistent Spaces

In this section we show the matching lower bound of Theorem 10 for consistent spaces for most relevant sizes of r, δ and k.

▶ **Lemma 15.** *Let $r = n^\alpha$, let $\alpha \in (0,1)$, $0 < \epsilon < \alpha$, $\gamma < \alpha - \epsilon$ be constants, and $1 \leq \delta \leq n^\gamma$. Let k, P_k as in Theorem 10. Let $x = \epsilon \frac{n}{r} \delta \log n = o(n)$. Then there exists a consistent space (H, V) of dimension δ, such that*

$$\mathbb{E}[|V_{P_k}(R)|] = (1 + o(1))\epsilon \frac{n}{r} \delta \log n = (1 + o(1))x.$$

Proof. Define (H, V) consistently as follows. The violator set of the empty set is defined as the empty set, $V(\emptyset) = \emptyset$. For all $B \subseteq H$ with $0 < |B| \leq \delta$, its violators are chosen u.a.r. of size $\epsilon \frac{n}{r} \delta \log n$ from $H \setminus B$.

For $R \subseteq H$ of size r we define the violators as follows. If there exists $B \subseteq R$, $0 < |B| \leq \delta$, such that $V(B) \cap R = \emptyset$, then $V(R) = V(B)$. If there exists more than one such B, choose the lexicographically smallest. If no such B exists then set $V(R) = V(\emptyset) = \emptyset$. Therefore for all R we have a basis of size at most δ. Denote the basis for R by B_R.

First we show that it suffices to treat the "worst case" $k = 0$. For $k > 0$ we can reduce the problem to the case $k = 0$ by the following construction, where for all R of size r it holds that $|V(R_{P_k})| \geq |V(R)|$: For R with $V(R) \neq \emptyset$, fix P_k such that none of the k removed elements are in B_R, i.e., $B_R \subseteq R_{P_k}$. Since $R \cap V(B) = \emptyset$ it follows that $R_{P_k} \cap V(B) = \emptyset$ and we can choose $V(R_{P_k}) = V(R)$. If there exists multiple sets of size r with nonempty violator set that are mapped to the same R_{P_k}, choose $V(R_{P_k})$ arbitrary from the set of their violator spaces. For all other sets of size $r - k$, choose their violators as the empty set. It follows that for all R of size r, $|V(R_{P_k})| \geq |V(R)|$. For all other $S \subseteq H$ define $V(S) = \emptyset$.

Hence we may assume that $k = 0$.

$$\mathbb{E}[|V(R)|] = \Pr[|V(R)| = \epsilon \frac{n}{r} \delta \log n] \cdot \epsilon \frac{n}{r} \delta \log n = (1 - \Pr[|V(R)| = 0]) \cdot \epsilon \frac{n}{r} \delta \log n.$$

We now show that $\Pr[|V(R)| = 0] = o(1)$, which concludes the proof. Because we chose the violators of the bases independently

$$\begin{aligned}\Pr[|V(R)| = 0] &= \Pr[\forall B \subseteq R, 0 < |B| \leq \delta \mid V(B) \cap R \neq \emptyset] \\ &= \prod_{\substack{B \subseteq R \\ 0<|B|\leq\delta}} \Pr[V(B) \cap R \neq \emptyset] = \prod_{\substack{B \subseteq R \\ 0<|B|\leq\delta}} (1 - \Pr[V(B) \cap R = \emptyset]).\end{aligned}$$

Now we bound $\Pr[V(B) \cap R = \emptyset]$ from below. For B of size β with $0 < \beta \leq \delta$ and $x = \epsilon\frac{x}{r}\delta \log n$, we get

$$\begin{aligned}\Pr[V(B) \cap R = \emptyset] &= \frac{\binom{n-x-\beta}{r-\beta}}{\binom{n-\beta}{r-\beta}} = \frac{(n-x-\beta)\cdots(n-x-r+1)}{(n-\beta)\cdots(n-r+1)} \\ &= e^{\left((1+o(1))\frac{x}{n}+\Theta(\frac{x^2}{n^2})\right)r} = n^{-(1+o(1))\epsilon\delta}.\end{aligned}$$

Using that $\sum_{i=1}^{\delta}\binom{r}{i} \geq \frac{(r-\delta)^\delta}{\delta^\delta}$ and $\delta = o(r)$ we get

$$\begin{aligned}\Pr[|V(R)| = 0] &\leq \prod_{\substack{B \subseteq R \\ 0<|B|\leq\delta}} (1 - n^{-(1+o(1))\epsilon\delta}) \leq (1 - n^{-(1+o(1))\epsilon\delta})^{\frac{(r-\delta)^\delta}{\delta^\delta}} \\ &\leq \exp\left(-(1+o(1))n^{-(1+o(1))\epsilon\delta} \cdot \frac{(1+o(1))^\delta r^\delta}{\delta^\delta}\right).\end{aligned}$$

Plugging in $r = n^\alpha$, we observe that is sufficient to show that $n^{(\alpha-\epsilon+o(1))\delta}(1+o(1))^\delta \cdot \frac{1}{\delta^\delta} = \omega(1)$. By using $\delta \leq n^\gamma$ we get

$$\frac{n^{(\alpha-\epsilon+o(1))\delta} \cdot (1+o(1))^\delta}{\delta^\delta} \geq \left(n^{(\alpha-\epsilon-\gamma+o(1))} \cdot (1+o(1))\right)^\delta = \omega(1),$$

since $\gamma < \alpha - \epsilon$. ◀

We show that using one of the simplest violator spaces, namely the d-smallest number problem, we obtain the bound of $\mathbb{E}[|V_{P_k}(R)|] = \Theta(\frac{n}{r} \cdot (\delta + k))$.

▶ **Definition 16.** We define the *d-smallest number problem* as follows. Let $H = [n] = \{1, 2, \ldots, n\}$. For $R \subseteq H$, define $\min_d(R)$ as the d-smallest number in R. Let $V(R) = \{r \in H \setminus R \mid r < \min_d(R)\}$, i.e. all elements smaller than the d-smallest.

We observe that (H, V) is a violator space, with combinatorial dimension d. The basis of R consists of the d smallest elements of R.

▶ **Lemma 17.** *Let $H = [n]$ and $\delta + k \leq r$. For the δ-smallest number problem there exists a rule P_k such that $\mathbb{E}[|V_{P_k}(R)|] = \frac{n-r}{r+1} \cdot (\delta + k) + k$, and therefore for $r = o(n)$, $\mathbb{E}[|V_{P_k}(R)|] = \Theta(\frac{n}{r} \cdot (\delta + k))$.*

Proof of Lemma 17. Let $R \subseteq H$. To maximize the number of violators after the removal of k elements, we remove the k-smallest elements of R. We call this rule P_k and the removed set R_k. Then

$$V_{P_k}(R) = \{r \in H \setminus R \mid r < \min_{\delta+k}(R)\} \cup R_k.$$

We observe that $\{r \in H \setminus R \mid r < \min_{\delta+k}(R)\}$ is exactly the set of violators of R, for the $\delta + k$ smallest problem, whose expected size we know by the Sampling Lemma 6. Hence

$$\mathbb{E}[|V_{P_k}(R)|] = \frac{n-r}{r+1}(\delta + k) + k.$$ ◀

Lemma 15 and Lemma 17 show that for consistent spaces the bound of Theorem 10 is tight up to a constant factor for most relevant values of r, δ and k, (i.e., if r, δ and k satisfy the conditions of Lemma 15 or Lemma 17). Furthermore by Lemma 17 if $k \geq \delta \log n$, then the upper bound of Theorem 10 is tight for violator spaces.

6 Tight Bounds for Violator Spaces with Combinatorial Dimension 1

In the case of violator spaces it is open whether (or when) the upper bound of Theorem 10 is tight for $k < \delta \log n$. In this case, there is a gap of up to $\log n$ between upper and lower bounds. Theorem 14 shows that for *non-degenerate* violator spaces, the bound of Theorem 10 is not tight for very small δ and k, but it could still be tight in the general case. For $k = 0$ we know a stronger upper bound of $O(\frac{n-r}{r+1}\delta)$ by the Sampling Lemma 6.

In the following we prove a stronger upper bound for $\delta = 1$ for violator spaces, and we give an example showing that this bound is tight. We prove that there exists only one class of violator spaces of dimension 1, namely the class of the smallest number with repetitions violator space.

▶ **Theorem 18.** *For $\delta = 1$, $\mathbb{E}[|V_{P_k}(R)|] = O(\frac{n}{r}k)$, and this bound is tight.*

It follows that the upper bound given in Theorem 14 is tight for all violator spaces of dimension 1.

▶ **Definition 19.** We define the class of *smallest number with repetitions* violator space as follows. Let $|H| = n$ and H a multiset of $[n]$, i.e., every element of H is in $[n]$ and there might be repetitions. For $R \subseteq H$, let $V(R) = \{x \in H \mid x < \min_{i \in R} i\}$. Finally we require that either $V(\emptyset) = H$ or $V(\emptyset) = V(i)$ for some $i \in H$.

Observe that this a a violator space of dimension 1 and similarly as in the proof of Lemma 17, we can show that $\mathbb{E}[|V_{P_k}(R)|] = O(\frac{n}{r}k)$.

▶ **Lemma 20.** *Let (H, V) be a violator space of dimension 1. If for all $i \in H$, $V(\emptyset) = H$ or $V(i) = \{x \in H \mid x < i\}$ and $V(\emptyset) = V(i)$ for some i, then $V(R) = \{x \in H \mid x < \min_{i \in R} i\}$ for all $R \subseteq H$. This means that the violators of sets of size 0 and 1 uniquely define all other violators.*

Proof. Let $R \subseteq H$, $|R| \geq 2$ with $y := \min_{i \in R} i$. Because the dimension of the violator space is 1 we have $V(R) = V(x)$ for some $x \in R$. Now for all $x > y$, we have $y \in V(x)$, hence $V(R) = V(y)$. ◀

▶ **Lemma 21.** *Every violator space (H, V) of dimension 1 with $|H| = n$, is homeomorphic to an instance of smallest number with repetitions, i.e., those are the only violator spaces of dimension 1 that exist.*

Proof. Let (H, V) be a violator space with $|H| = n$. Assume that $V(\emptyset) \neq H$. Then there exists $i \in H \setminus V(\emptyset)$ and therefore $V(i) = V(\emptyset)$. The following holds for all $i \neq j \in H$.

1. If $V(i) \neq V(j)$ then $i \in V(j)$ or $j \in V(i)$.
2. $V(i) \subseteq V(j)$ or $V(j) \subseteq V(i)$.

For the first part assume that $i \notin V(j)$ and $j \notin V(i)$. Then by locality $V(i, j) = V(i) = V(j)$.

For the second part assume that there exists $k \neq l$ such that $k \in V(i) \setminus V(j)$ and $l \in V(j) \setminus V(i)$. We consider $V(i, j, k, l)$. Since $\delta = 1$ we have that $V(i, j, k, l) = V(m)$ for

some $m \in \{i,j,k,l\}$. By consistency we know $V(i,j,k,l) \neq V(i)$ since $k \in V(i)$. Similarly $V(i,j,k,l) \neq V(j)$. Therefore w.l.o.g. assume that $V(i,j,k,l) = V(k)$. Then $j \notin V(k)$ and $k \notin V(j)$ and hence by the first part $V(j) = V(k) = V(i,j,k,l)$, which is a contradiction.

We now construct a mapping $f : H \to [n]$, such that $j \in V(i)$ if and only if $f(j) < f(i)$. By Lemma 20 this concludes the proof.

We construct a sequence of pairwise disjoint nonempty sets $V_1, V_2, \dots V_m \subseteq H$, $m \leq n$, $V_1 \cup \dots \cup V_m = H$ such that the following holds for all $i \in [m]$: For all $x \in V_i$ we have $V(x) = V_1 \cup \dots \cup V_{i-1}$.

By setting $f^{-1}(i) = V_i$ for all $i \in [m]$, this is one instance of minimum number with repetitions violator space.

Suppose that for some $i \geq 1$ we have constructed $V_1, \dots, V_{i-1}$ and $H \setminus (V_1, \dots, V_{i-1}) \neq \emptyset$. Let V_i be the subset of $H \setminus (V_1, \dots, V_{i-1})$ with inclusion-minimal violator sets, i.e., $x \in H \setminus (V_1, \dots, V_{i-1})$ is in V_i if and only if there exists no $y \in H \setminus (V_1, \dots, V_{i-1})$ such that $V(y) \subsetneq V(x)$. Then obviously V_i is nonempty. We need to show that for such x, $V(x) = V_1 \cup \dots \cup V_{i-1}$. Let $y \in V(x)$. Since $y \notin V(y)$ condition 2. implies that $V(y) \subsetneq V(x)$, hence $y \in V_1 \cup \dots \cup V_{i-1}$. Now let $y \notin V(x)$. If $x \notin V(y)$ then by condition 1. it follows that $V(x) = V(y)$ and hence $y \in V_i$. Otherwise $x \in V(y)$ and therefore by condition 2. $V(x) \subsetneq V(y)$. It follows that $y \notin V_1 \cup \dots \cup V_{i-1}$. ◀

Proof of Theorem 18. The bound follows immediately from Lemma 21 and by the fact that $\mathbb{E}[|V_{P_k}(R)|] = O(\frac{n}{r}k)$, for the smallest number with repetitions LP-type problem. Tightness follows from Lemma 17. ◀

Acknowledgments. The authors are grateful to Kenneth Clarkson and Emo Welzl for sharing important insights. Furthermore we thank Luis Barba for useful discussions.

References

1. Y. Brise and B. Gärtner. Clarkson's algorithm for violator spaces. *Computational Geometry*, 44(2):70–81, 2011. Special issue of selected papers from the 21st Annual Canadian Conference on Computational Geometry. doi:10.1016/j.comgeo.2010.09.003.
2. M. C. Campi and S. Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM J. Optim.*, 19:1211–1230, 2008.
3. M. C. Campi and S. Garatti. A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *J. Optim. Theory Appl.*, 148:257–280, 2011.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA., 1990.
5. B. Gärtner. Sampling with removal in LP-type problems. *Journal of Computational Geometry*, 6(2):93–112, 2015.
6. B. Gärtner, J. Matoušek, L. Rüst, and P. Škovroň. Violator spaces: Structure and algorithms. *Discrete Appl. Math.*, 156(11):2124–2141, June 2008. doi:10.1016/j.dam.2007.08.048.
7. B. Gärtner, W. D. Morris, Jr., and L. Rüst. Unique sink orientations of grids. In *Proc. 11th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 3509 of *Lecture Notes in Computer Science*, pages 210–224. Springer-Verlag, 2005.
8. B. Gärtner and E. Welzl. A simple sampling lemma: Analysis and applications in geometric optimization. *Discrete & Computational Geometry*, 25(4):569–590, 2001.
9. J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

10 J. Matoušek. Removing degeneracy in LP-type problems revisited. *Discrete & Computational Geometry*, 42(4):517–526, 2009. doi:10.1007/s00454-008-9085-7.

11 M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, STACS'92, pages 569–579, London, UK, UK, 1992. Springer-Verlag. doi:10.1007/3-540-55210-3_213.

12 T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 547–555, 2000.

13 E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *Results and New Trends in Computer Science*, pages 359–370. Springer-Verlag, 1991.

The Planar Tree Packing Theorem

Markus Geyer[1], Michael Hoffmann[*2], Michael Kaufmann[3], Vincent Kusters[†4], and Csaba D. Tóth[‡5]

1 **Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
geyer,@informatik.uni-tuebingen.de
2 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
hoffmann@inf.ethz.ch
3 **Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Tübingen, Germany**
mk@informatik.uni-tuebingen.de
4 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
vincent.kusters@inf.ethz.ch
5 **California State University Northridge, Los Angeles, USA**
cdtoth@acm.org

Abstract

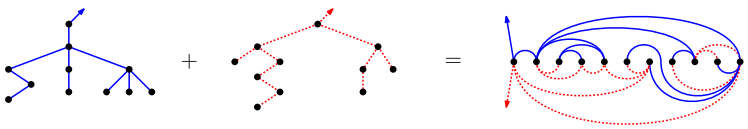
Packing graphs is a combinatorial problem where several given graphs are being mapped into a common host graph such that every edge is used at most once. In the planar tree packing problem we are given two trees T_1 and T_2 on n vertices and have to find a planar graph on n vertices that is the edge-disjoint union of T_1 and T_2. A clear exception that must be made is the star which cannot be packed together with any other tree. But according to a conjecture of García et al. from 1997 this is the only exception, and all other pairs of trees admit a planar packing. Previous results addressed various special cases, such as a tree and a spider tree, a tree and a caterpillar, two trees of diameter four, two isomorphic trees, and trees of maximum degree three. Here we settle the conjecture in the affirmative and prove its general form, thus making it the planar tree packing theorem. The proof is constructive and provides a polynomial time algorithm to obtain a packing for two given nonstar trees.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph drawing, simultaneous embedding, planar graph, graph packing

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.41

1 Introduction



The *packing problem* is to find a graph G on n vertices that contains a given collection $G_1, \ldots, G_k$ of graphs on n vertices each as edge-disjoint subgraphs. This problem has been

* Supported by the ESF EUROCORES programme EuroGIGA, CRP GraDR and the Swiss National Science Foundation, SNF Project 20GG21-134306.
† Supported by the ESF EUROCORES programme EuroGIGA, CRP GraDR and the Swiss National Science Foundation, SNF Project 20GG21-134306.
‡ Supported by the NSF awards CCF-1422311 and CCF-1423615.

© Markus Geyer, Michael Hoffmann, Michael Kaufmann, Vincent Kusters, and Csaba D. Tóth;
licensed under Creative Commons License CC-BY

studied in a wide variety of scenarios (see, e.g., [1, 3, 6]). Much attention has been devoted to the packing of trees (e.g., tree packing conjectures by Gyárfas [12] and by Erdős and Sós [5]). Hedetniemi [13] proved that any two nonstar trees can be packed into K_n. Teo and Yap [19] showed that *any* two graphs of maximum degree at most $n-1$ with a total of at most $2n-2$ edges pack into K_n unless they are one of thirteen specified pairs of graphs. Maheo et al. [14] characterized triples of trees that can be packed into K_n.

In the *planar packing* problem the graph G is required to be planar. García et al. [9] conjectured in 1997 that there exists a planar packing for any two nonstar trees, that is, for any two trees with diameter greater than two. The assumption that none of the trees is a star is necessary, since a star uses all edges incident to one vertex and so there is no edge left to connect that vertex in the other tree. García et al. proved their conjecture when one of the trees is a path and when the two trees are isomorphic. Oda and Ota [17] addressed the case that one of the trees is a caterpillar or that one of the trees is a spider of diameter at most four. A *caterpillar* is a tree that becomes a path when all leaves are deleted and a *spider* is a tree with at most one vertex of degree greater than two. Frati et al. [8] gave an algorithm to construct a planar packing of any spider with any tree. Frati [7] proved the conjecture for the case that both trees have diameter at most four. Finally, Geyer et al. [10] proved the conjecture for binary trees (maximum degree three). In this paper we settle the general conjecture in the affirmative:

▶ **Theorem 1.** *Every two nonstar trees of the same size admit a planar packing.*

Related work. Determining relationships between a graph and its subgraphs is one of the most studied topics in graph theory. The *subgraph isomorphism* problem asks to find a subgraph H in a graph G. The *graph thickness* problem [15] asks for the minimum number of planar subgraphs which the edges of a graph can be partitioned into. The *arboricity* problem [4] asks to determine the minimum number of forests which a graph can be partitioned into. Another related classical combinatorial problem is the k edge-disjoint spanning trees problem which dates back at least to Tutte [20] and Nash-Williams [16], who gave necessary and sufficient conditions for the existence of k edge-disjoint spanning trees in a graph. The interior edges of every maximal planar graph can be partitioned into three edge-disjoint trees, known as a *Schnyder wood* [18]. Gonçalves [11] proved that every planar graph can be partitioned in two edge-disjoint outerplanar graphs.

The study of relationships between a graph and its subgraphs can also be done the other way round. Instead of decomposing a graph, one can ask for a graph G that encompasses a given set of graphs $G_1, \ldots, G_k$ and satisfies some additional properties. This topic occurs with different flavors in the computational geometry and graph drawing literature. It is motivated by applications in visualization, such as the display of networks evolving over time and the simultaneous visualization of relationships involving the same entities. In the *simultaneous embedding* problem [2] the graph $G = \bigcup G_i$ is given and the goal is to draw it so that the drawing of each G_i is plane. The *simultaneous embedding without mapping* problem [2] is to find a graph G on n vertices such that: (i) G contains all G_i's as subgraphs, and (ii) G can be drawn with straight-line edges so that the drawing of each G_i is plane.

2 Notation and Overview

A *rooted tree* is a directed tree T with exactly one vertex of outdegree zero: its root, denoted $\uparrow(T)$. Every vertex $v \neq \uparrow(T)$ has exactly one outgoing edge $(v, \mathrm{p}_T(v))$. The target $\mathrm{p}_T(v)$ is the *parent* of v in T, and conversely v is a *child* of $\mathrm{p}_T(v)$. In figures we denote the root

of a tree by an outgoing vertical arrow. For a vertex v of a rooted tree T, denote by $\mathrm{t}_T(v)$ the *subtree rooted at* v, that is, the subtree of T induced by the vertices from which v can be reached on a directed path. The subscript is sometimes omitted if T is clear from the context. A *subtree of (or below)* v is a tree $\mathrm{t}_T(c)$, for a child c of v in T. For a tree T, denote by $|T|$ the *size* (number of vertices) of T. We denote by $\deg_T(v)$ the degree (indegree plus outdegree) of v in T. For a graph G we denote by $\mathrm{E}(G)$ the edge set of G. A *star* is a tree on n vertices that contains at least one vertex of degree $n-1$. Such a vertex is a *center* of the star. A star on $n \neq 2$ vertices has a unique center. For a star on two vertices, both vertices act as a center. When considered as a rooted tree, there are two different rooted stars on $n \geq 3$ vertices. A star rooted at a center is called *central-star*, whereas a star rooted at a leaf that is not a center is called a *dangling star*. In particular, every star on one or two vertices is a central-star. A *nonstar* is a graph that is not a star. A *one-page book embedding* of a graph G is an embedding of G into a closed halfplane such that all vertices are placed on the bounding line. This line is called the *spine* of the book embedding.

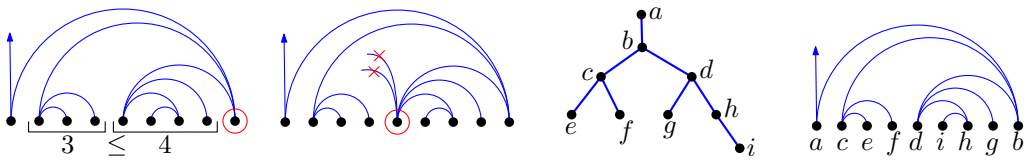
We embed vertices equidistantly along the positive x-axis and refer to them by their x-coordinate, that is, $P = \{1, \ldots, n\}$. An *interval* $[i, j]$ in P is a sequence of the form $i, i+1, \ldots, j$, for $1 \leq i \leq j \leq n$, or $i, i-1, \ldots, j$, for $1 \leq j \leq i \leq n$. Observe that we consider an interval $[i, j]$ as oriented and so we can have $i > j$. Denote the *length* of an interval $[i, j]$ by $|[i, j]| = |i - j| + 1$. To avoid notational clutter we often identify points from P with vertices embedded at them.

Overview. We construct a plane drawing of two n-vertex trees T_1 and T_2 on the point set $P = [1, n]$. We call T_1 the *blue tree*; its edges are shown as solid blue arcs in figures. The tree T_2 is called the *red tree*; its edges are shown as dotted red arcs. The algorithm first computes a preliminary one-page book embedding of T_1 onto P (the *blue embedding*) in Section 3. In the second step we recursively construct an embedding for the red tree to pair up with the blue embedding. In principle we follow a similar strategy as in the first step, but we take the constraints imposed by the blue embedding into account. During this process we may reconsider and change the blue embedding locally. For instance, we may *flip* the embedding of some subtree of T_1 on an interval $[i, j]$, that is, reflect the embedded tree at the vertical line $x = \frac{i+j}{2}$ through the midpoint of $[i, j]$. In some cases we also perform more drastic changes to the blue embedding. In particular, the blue embedding may not be a one-page book embedding in the final packing. Although neither of the two trees T_1 and T_2 we start with is a star, it is possible – in fact, unavoidable – that stars appear as subtrees during the recursion. We have to deal with stars explicitly whenever they arise, because the general recursive step works for nonstars only. We introduce the necessary concepts and techniques in Section 4 and give the actual proof in Section 5.

3 A preliminary blue embedding

We begin by defining a preliminary one-page book embedding $\pi : V_1 \to [1, n]$ for a tree $T_1 = (V_1, E_1)$ rooted at $r_1 \in V$. In every recursive step, we are given a tree T rooted at a vertex r and an interval $[i, j]$ of length $|T|$. Recall that we may have $i < j$ or $i > j$. We place r at position i and recursively embed the subtrees of r on pairwise disjoint subintervals of $[i, j] \setminus \{i\}$. The embedding is guided by two rules illustrated in Figure 1.

- The *larger-subtree-first rule* (LSFR) dictates that for any two subtrees of r, the larger of the subtrees must be embedded on an interval closer to r. Ties are broken arbitrarily.
- The *one-side rule* (1SR) dictates that for every vertex all neighbors are mapped to the



(a) LSFR **(b)** 1SR **(c)** A preliminary blue embedding

Figure 1 Illustrations for the two rules and an example embedding.

same side. That is, if $N_T(v)$ denotes the set of neighbors of v in T (including its parent), then either $\pi(u) < \pi(v)$ for all $u \in N_T(v)$ or $\pi(u) > \pi(v)$ for all $u \in N_T(v)$.

These rules imply that every subtree $T \subseteq T_1$ is embedded onto an interval $[i, j] \subseteq [1, n]$ so that $\{i, j\}$ is an edge of T and either i or j is the root of T. Together with $\pi(r_1) = 1$, these rules define the embedding (up to tiebreaking). See Figure 1c for an example.

4 A red tree and a blue forest

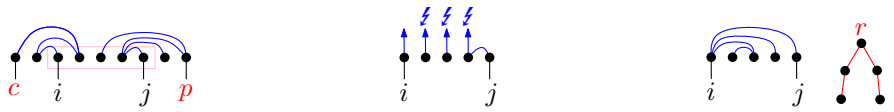
As common with inductive proofs, we prove a stronger statement than necessary. This stronger statement does not hold unconditionally but we need to impose some restrictions on the input. The goal of this section is to derive this more general statement – formulated as Theorem 2 – from which Theorem 1 follows easily.

Our algorithm receives as input a nonstar subtree R of the red tree and an interval $I = [i, j]$ of size $|R|$ along with a blue graph B embedded on I. Without loss of generality we assume $i < j$. In the initial call B is a tree, but in a general recursive call B is a *blue forest* that may consist of several components. For $k \in [i, j]$ let $B\langle k\rangle$ denote the component of B that contains k. For $[x, y] \subseteq [i, j]$ let $B[x, y]$ denote the subgraph of B induced by the vertices in $[x, y]$, and for $k \in [x, y]$ let $B[x, y]\langle k\rangle$ denote the component of $B[x, y]$ that contains k.

In general the algorithm sees only a small part of the overall picture because it has access to the vertices in I only. However, blue vertices in I may have edges to vertices outside of I and also vertices of R may have neighbors outside of I. We have to ensure that such *outside edges* are used by one tree only and can be routed without crossings. In order to control the effect of outside edges, we allow only one vertex in each component – that is, the root of R and the root of each component of B – to have neighbors outside of I. Whenever we change the blue embedding we need to maintain the relative order of these roots so as to avoid crossings among outside edges.

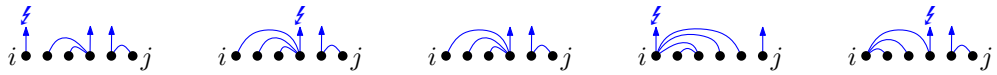
Conflicts. Typically $r := \uparrow(R)$ has at least one neighbor outside of I: its parent $\mathrm{p}_{T_2}(r)$. But r may also have children in $T_2 \setminus R$. We assume that all neighbors – parent and children – of r in $T_2 \setminus R$ are already embedded outside of I when the algorithm is called for R. There are two principal obstructions for mapping r to a point $v \in I$:

- A vertex $v \in I$ is in *edge-conflict* with r, if $\{v, r'\} \in \mathrm{E}(T_1)$ for some neighbor r' of r in $T_2 \setminus R$. Mapping r to v would make $\{v, r'\}$ an edge of both T_1 and T_2 (Figure 2a–2b). In figures we mark vertices in edge-conflict with r by a lightning symbol ↯.
- A vertex $v \in I$ is in *degree-conflict* with r on I if $\deg_R(r) + \deg_B(v) \geq |I|$. If we map r to v, then no child of r in R can be mapped to the same vertex as a child of v in B. With only $|I| - 1$ vertices available there is not enough room for both groups (Figure 2c).



(a) (b) (c)

Figure 2 An interval $[i, j]$ on which a tree $R = \mathrm{t}(r)$ is to be embedded. Two neighbors p and c of r in $T_2 \setminus R$ are already embedded (a). Then the situation on $[i, j]$ presents itself as in (b), where the three central vertices are in edge-conflict with r due to blue outside edges to p or c. In (c) the vertex i is in degree-conflict with r because $\deg_R(r) + \deg_B(i) = 2 + 3 = 5 \geq |[i, j]|$. We cannot map r to the blue vertex at i because there is not enough room for the neighbors of both in $[i, j]$.



(a) (b) (c) (d) (e)

Figure 3 An interval $[i, j]$ in edge-conflict (a)–(b), and examples where $[i, j]$ is not in edge-conflict (c)–(e). In (c) the center of B^* is not in edge-conflict; it may be in degree-conflict, though, if $\deg_R(r) \geq 3$. In both (d) and (e) the tree $B\langle i \rangle$ is not a central-star.

We cannot hope to avoid conflicts entirely and we do not need to. It turns out that is sufficient to avoid a very specific type of conflict involving stars.

- An interval $[i, j]$ is in *edge-conflict* (*degree-conflict*) with $R = \mathrm{t}(r)$ if $B^* := B\langle i \rangle$ is a central-star and the root of B^* is in edge-conflict (degree-conflict) with r (Figure 3).
- An interval I is in *conflict* with R if I is in edge-conflict or degree-conflict with R (or both).

We claim that R can be packed with B onto I unless I is in conflict with R. The following theorem presents a precise formulation of this claim. Only R and the graph $B\langle i \rangle$ determine whether or not an interval $[i, j]$ is in conflict with R. Therefore we can phrase the statement without referring to an embedding of B but just regarding it as a sequence of trees. The set C represents the set of roots from B that are in edge-conflict with r.

▶ **Theorem 2.** *Let R be a nonstar tree with $r = \uparrow(R)$ and let B be a nonstar forest with $|R| = |B| = n$, together with an ordering $b_1, \ldots, b_k$ of the $k \in \{1, \ldots, n\}$ roots of B and a set $C \subseteq \{b_1, \ldots, b_k\}$. Suppose (i) $\mathrm{t}_B(b_1)$ is not a central-star or (ii) $b_1 \notin C$ and $\deg_R(r) + \deg_B(b_1) < n$. Then there is a plane packing π of B and R onto any interval I with $|I| = n$ such that*

- *$\pi(r) \notin \pi(C)$ and*
- *we can access $b_1, \ldots, b_k, r$ in this order from the outer face of π, that is, we can add a new vertex v in the outer face of π and route an edge to each of $b_1, \ldots, b_k, r$ such that the resulting multigraph is plane and the circular order of neighbors around v is $b_1, \ldots, b_k, r$. (If $r = b_i$, for some $i \in \{1, \ldots, k\}$, then two distinct edges must be routed from v to r so that the result is a non-simple plane multigraph.)*

Such a packing π we call an ordered plane packing *of B and R onto I.*

Theorem 2 is a strengthening of Theorem 1 and so we obtain Theorem 1 as an easy corollary.

Proof of Theorem 1 from Theorem 2. Select roots arbitrarily so that $T_1 = \mathrm{t}(r_1)$ and $T_2 = \mathrm{t}(r_2)$. Then use Theorem 2 with $R = T_2$, $B = T_1$, $k = 1$, $b_1 = r_1$, and $C = \emptyset$. By assumption T_1 is not a star and so (i) holds. Therefore we can apply Theorem 2 and obtain the desired plane packing of T_1 and T_2. ◀

(a) $b_1 \in C$ **(b)** $\deg_R(r) + \deg_{\mathrm{t}(b_1)}(b_1) \geq n$

Figure 4 The statement of Theorem 2 does not hold without (i) or (ii). In the examples the trees of B are ordered from left to right so that $\mathrm{t}(b_1)$ is a central-star. Vertices in C are labeled with ↯.

It is not hard to see that forbidding conflicts in Theorem 2 is necessary: The example families depicted in Figure 4 do not admit an ordered plane packing.

5 Embedding the red tree

In this section we outline our recursive embedding algorithm to prove Theorem 2. We are given a red tree $R = \mathrm{t}(r)$, a blue forest B with roots $b_1, \ldots, b_k$, an interval $I = [i, j] \subseteq [1, n]$ with $|I| = |R| = |B|$, and a set C that we consider to be the vertices from B in edge-conflict with r. As a first step, we embed B onto I by embedding $\mathrm{t}(b_1), \ldots, \mathrm{t}(b_k)$ in this order from left to right, each time using the algorithm from Section 3.

▶ **Observation 3.** *We may assume that R, B and $I = [i, j]$ satisfy the following invariants:*

I1. *I is not in conflict with R.* (peace invariant)

I2. *Every component of B satisfies LSFR and 1SR. All edges of B are drawn in the upper halfplane (above the x-axis). All roots of B are visible from above (that is, a vertical ray going up from b_x does not intersect any edge of B).* (blue-local invariant)

I3. *i is not in edge-conflict with r.* (placement invariant)

Proof. 1 follows from the assumption (i) or (ii) in Theorem 2. 2 is achieved by using the embedding from Section 3. If i is in conflict with r, then 1 implies that $B\langle i\rangle$ is not a singleton (which would be a central-star). Therefore flipping $B\langle i\rangle$ establishes 3 without affecting 1 or 2. ◀

Theorem 2 ensures that all roots of B along with r appear on the outer face in the specified order. We cannot assume that we can draw an edge to any other vertex of B or R without crossing edges of the embedding given by Theorem 2. Therefore it is important that whenever the algorithm is called recursively,

I4. only the roots $b_1, \ldots, b_k$ and r have edges to the outside of I.

▶ **Observation 4.** *If B satisfies 2 and 4 on an interval I, then both invariants also hold for $B[x, y]$ on $[x, y]$, for every subinterval $[x, y] \subseteq I$.*

In the remainder of the proof we will ensure and assume that invariants 1–4 hold for every call of the algorithm.

The algorithm. Let s denote a child of r that minimizes $|\mathrm{t}_R(c)|$ among all children c of r in R. Denote $S = \mathrm{t}_R(s)$ and $R^- = R \setminus S$. If $|R^-| \geq 2$, then R^- cannot be a central-star: if it were, then $|S| = 1$ and R would be a star. Ideally, we can recursively embed S onto $[j, j - |S| + 1]$ and R^- onto $[i, j - |S|]$ (Figure 5a). But in general the invariants may not hold for the recursive subproblems. For instance, some of the subgraphs could be stars, or if

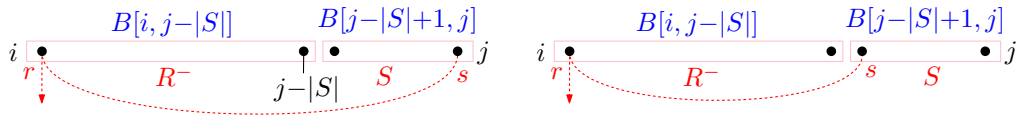


Figure 5 Our recursive strategy in an ideal world.

$\{i, j\} \in \mathrm{E}(B)$, then placing r at i may put $[j, j - |S| + 1]$ in edge-conflict with S. Therefore, we explore a number of alternative strategies, depending on which – if any – of the four forests R^-, S, $B[i, j - |S|]$ and $B[j - |S| + 1, j]$ in our decomposition is a star.

Due to space constraints we can only discuss a small part of the proof in the main text. We chose to present the general case, in which none of R^-, S, $B[i, j-|S|]$ and $B[j-|S|+1, j]$ is a star. The analysis for the remaining cases can be found in the full paper.

▶ **Lemma 5.** *If none of S, R^-, $B[i, j - |S|]$, and $B[j - |S| + 1, j]$ is a star, then there is an ordered plane packing of B and R onto I.*

Proof. As S is a minimum size subtree of r in R, and neither S nor R^- is a star, we know that r has at least one more subtree other than S and every subtree of r in R has size at least four. (All trees on three or less vertices are stars.) It follows that

$$\deg_{R^-}(r) \leq (|R^-| - 1)/4. \tag{1}$$

The general plan is to use one of the following two options. In both cases we first embed R^- recursively onto $[i, j - |S|]$. Then we conclude as follows.
Option 1: Embed S recursively onto $[j, j - |S| + 1]$ (Figure 5a).
Option 2: Embed S recursively onto $[j - |S| + 1, j]$ (Figure 5b).
In some cases neither of these two options works and so we have to use a different embedding.

As we embed S after R^-, the (final) mapping for s is not known when embedding R^-. However, we need to know the position of s in order to determine the conflicts for embedding R^-. Therefore, before embedding R^- we *provisionally* embed s at $\alpha := \uparrow(B[j - |S| + 1, j]\langle j\rangle)$ (Option 1) or $\alpha := \uparrow(B[j - |S| + 1, j]\langle j - |S| + 1\rangle)$ (Option 2). That is, for embedding R^- we pretend that some neighbor of r is embedded at α. In this way we ensure that S is not in edge-conflict with the interval in its recursive embedding. The final placement for s is then determined by the recursive embedding of S, knowing the definite position of its parent r.

For the recursive embeddings to work, we need to show that the invariants 1, 2 and 4 hold (3 then follows as in Observation 3). For 2 and 4 this is obvious by construction and Observation 4, as long as we do not change the embedding of B. As we do not change the embedding in Option 1 and 2, it remains to ensure 1. So suppose that for both options, 1 does not hold for at least one of the two recursive embeddings. There are two possible obstructions for 1: edge-conflicts and degree-conflicts. We discuss both types of conflicts, starting with edge-conflicts.

Case 1 $[i, j - |S|]$ is not in degree-conflict with R^- and $[j, j - |S| + 1]$ is not in degree-conflict with S. Then Option 1 works, unless $[i, j - |S|]$ is in edge-conflict with R^-. Recall that $[j, j - |S| + 1]$ is not in edge-conflict with S after embedding R^- onto $[i, j - |S|]$.

We claim that an edge-conflict between R^- and $[i, j - |S|]$ implies $\{i, j\} \in \mathrm{E}(B)$. To prove this claim, suppose that $[i, j - |S|]$ is in edge-conflict with R^-. Then $B[i, j - |S|]\langle i\rangle$ is a central-star whose root c is in edge-conflict with r. If $c = i$, then by 3 there was no such conflict initially. So, as claimed, the conflict can only come from a blue edge to s

Figure 6 A third embedding when the first two options fail.

(provisionally placed) at j. Otherwise, $c > i$ and by 1SR there is no edge in B from c to any point in $[c+1, j]$. It follows that $B[i, j-|S|]\langle i\rangle = B\langle i\rangle$. The conflict between c and r does not come from the edge to s but from an edge to a vertex outside of $[i, j]$. This contradicts 1 for R and $[i, j]$, which proves the claim.

The presence of the edge $\{i, j\}$ implies that B is a tree and by 4 only (the root) i or j may have edges out of $[i, j]$. Consider Option 2, which embeds S onto $[j-|S|+1, j]$, provisionally placing s at $\uparrow(B[j-|S|+1, j]\langle j-|S|+1\rangle)$. There are two possible obstructions: an edge-conflict for R^- or a degree-conflict for S. In both cases we face a central-star $B^* = B[j-|S|+1, b]$ with center $b \in [j-|S|+1, j-1]$. Due to 1SR and $\{i, j\} \in \mathrm{E}(B)$, we know that $b = \uparrow(B[j-|S|+1, j]\langle j-|S|+1\rangle)$. We distinguish three cases.

Case 1.1 $\{i, b\} \in \mathrm{E}(B)$. Then we consider a third option: provisionally place s at j, embed R^- recursively onto $[j-|S|, i]$ and then S onto $[j, j-|S|+1]$ (Figure 6a). The edge $\{i, b\}$ of B prevents any edge-conflict between $[j-|S|, i]$ and R^- (and, as before, for S). Given that we assume in Case 1 that $[j, j-|S|+1]$ is not in degree-conflict with S, we are left with $[j-|S|, i]$ being in degree-conflict with R^- as a last possible obstruction.
Then the tree $B[i, j-|S|]\langle j-|S|\rangle$ is a central-star A^* with root a such that

$$\deg_{A^*}(a) + \deg_{R^-}(r) \geq |R^-|. \tag{2}$$

The following lemma holds in general. Its proof is omitted due to space constraints.

▶ **Lemma 6.** *If* $\deg_R(r) \geq 2$, *then* $|R^-| \geq |S| + \deg_{R^-}(r)$.

Combining Lemma 6 with (2) we get $|A^*| = \deg_{A^*}(a) + 1 \geq |S| + 1 \geq 5$. Note that A^* can be huge, but we know that it does not include i (because $B[i, j-|S|]$ is not a star). We also know that $a \neq j-|S|$: If $a = j-|S|$, then by 1SR we have $\mathrm{p}_B(a) \in [i, j-|S|-1]$, in contradiction to $a = \uparrow(B[i, j-|S|]\langle j-|S|\rangle$. Therefore $a = j-|S|-|A^*|+1$ and by 1SR its parent is to the right. Due to $\{i, b\} \in \mathrm{E}(B)$ and since $B[j-|S|+1, b]$ is a tree rooted at b, we have $\mathrm{p}_B(a) = b$. As A^* is a subtree of b in B on at least five vertices, by LSFR b cannot have a leaf at $b-1$. Therefore, the star $B[j-|S|+1, j]\langle j-|S|+1\rangle$ consists of a single vertex only, that is, $b = j-|S|+1$ (Figure 6b). We consider two subcases. In both the packing is eventually completed by recursively embedding S onto $[j, j-|S|+1]$.

Case 1.1.1 $\{x, b\} \in \mathrm{E}(B)$, for some $x \in [i+1, a-1]$ (Figure 7a). Select x to be maximal with this property. Then we exchange the order of the two subtrees $\mathrm{t}(x)$ and A^* of b (Figure 7b). This may violate LSFR for B at b, but 2 holds for both $B[i, j-|S|]$ and $B[j-|S|+1, j]$. Clearly there is still no edge-conflict for $[j-|S|, i]$ with R^- after this change. We claim that there is no degree-conflict anymore, either.

To prove the claim, note that by LSFR at b we have $|\mathrm{t}(x)| \leq |A^*|$. As the size of both subtrees combined is at most $|R^-|$, we have $|\mathrm{t}(x)| \leq |R^-|/2$. Then, using (1), $|\mathrm{t}(x)| - 1 + \deg_{R^-}(r) < |R^-|/2 + \deg_{R^-}(r) < 3|R^-|/4 < |R^-|$. Therefore after the exchange $[j-|S|, i]$ is not in degree-conflict with R^-, which proves the claim and concludes this case.

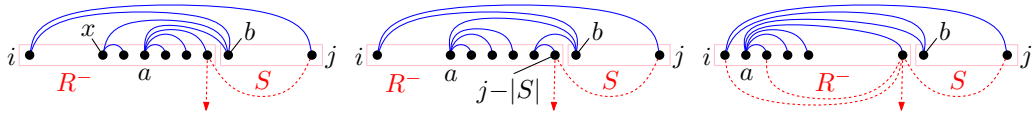


Figure 7 Swapping two subtrees of b in Case 1.1.1 and an explicit embedding for Case 1.1.2.

Case 1.1.2 i and $a = j - |S| - |A^*| + 1$ are the only neighbors of b in B. We claim that in this case A^* extends all the way up to $i+1$, that is, $A^* = B[i+1, j-|S|]$. To prove this claim, suppose to the contrary that $a \geq i+2$. Then there is another subtree of i to the left of a and, in particular, $\{i, a-1\} \in \mathrm{E}(B)$. By LSFR this closer subtree is at least as large as A^*. Using (1) and (2) we get $|[i+1, a-1]| + |A^*| \geq 2|A^*| > 2(|R^-| - \deg_{R^-}(r)) > 3|R^-|/2 > |R^-|$, in contradiction to $|[i+1, a-1]| + |A^*| < |R^-|$. Therefore $a = i+1$, as claimed (Figure 7c).

The vertex a has high degree in B but it is not adjacent to i. Therefore, we can embed R^- as follows: put r at $j - |S|$ and embed an arbitrary subtree Y of r onto $[i, i+|Y|-1]$ recursively or, if it is a star, explicitly, using the locally isolated vertex at i for the center (and $i+|Y|-1$ for the root in case of a dangling star). As i is isolated on $[i, i+|Y|-1]$ there is no conflict between $[i, i+|Y|-1]$ and Y. As $|Y| \geq |S| \geq 4$, the remaining graph $B[i+|Y|, j-|S|-1]$ consists of isolated vertices only, on which we can explicitly embed any remaining subtrees of r using the algorithm from Section 3.

Case 1.2 $\{i, b\} \notin \mathrm{E}(B)$ and $b = \mathrm{p}_B(j - |S|)$. Then $j - |S|$ is a locally isolated vertex in $B[i, j-|S|]$, whose only neighbor in B is at $b \notin B[j-|S|+1, j]\langle j \rangle$. Therefore, we can provisionally place s at j so that $[j-|S|, i]$ is not in conflict with R^-. By the assumption of Case 1 $[j, j-|S|+1]$ is not in degree-conflict with S. Therefore, we obtain the claimed packing by first embedding R^- onto $[j-|S|, i]$ recursively and then S onto $[j, j-|S|+1]$.

Case 1.3 $\{i, b\} \notin \mathrm{E}(B)$ and $b \neq \mathrm{p}_B(j - |S|)$. As $\{i, b\} \notin \mathrm{E}(B)$ and s is provisionally placed at b, the interval $[i, j-|S|]$ is not in edge-conflict with R^-. Thus, Option 2 (Figure 5b) succeeds unless $[j-|S|+1, j]$ is in degree-conflict with S. Hence suppose

$$\deg_S(s) + \deg_{B^*}(b) \geq |S|. \tag{3}$$

The following lemma holds in general. Its easy proof is omitted due to space constraints.

▶ **Lemma 7.** *If an interval $[i, j]$ is in degree-conflict with a nonstar subtree R of T_2, then $B\langle i \rangle$ is a central-star on at least three vertices.*

Hence, $|B^*| \geq 3$ by Lemma 7. As $b \neq \mathrm{p}_B(j - |S|)$, by LSFR b has exactly one neighbor in B outside of B^*: its parent $\mathrm{p}_B(b) \in [i+1, j-|S|]$ (Figure 8a). Let $B^+ = B^* \cup \{\mathrm{p}_B(b)\}$. The plan is to explicitly embed S such that the children of s that do not fit on $[b+1, j]$ – there is no room for all by (3) – are put on $[i+1, j] \setminus B^+$, and the rest of S on B^*. For this to work, we need

$$|S| - \deg_S(s) \leq |B^*| \text{ and } \deg_S(s) \leq |I| - 1 - |B^+|. \tag{4}$$

The first inequality follows from (3) and the second from $|B^*| + 1 + \deg_S(s) \leq (|S|-1) + 1 + (|S|-1) \leq |S| + (|R^-|-1) - 1 = |I| - 2$. We embed S as follows: put s at b and embed a child of s onto every vertex in $[b+1, j]$. By (3), at least one child of s remains. Let y be such that the remaining children of s fit exactly in $[y, j-|S|] \setminus \{\mathrm{p}_B(b)\}$ and embed them there.

Suppose $\mathrm{p}_B(b) < y$. Denote the $d := \deg_S(s)$ children of s in S by $c_1, \ldots, c_d$ from right to left (c_1 is embedded at j and c_d at y). For each c_k, $1 \leq k \leq d$, we need an additional

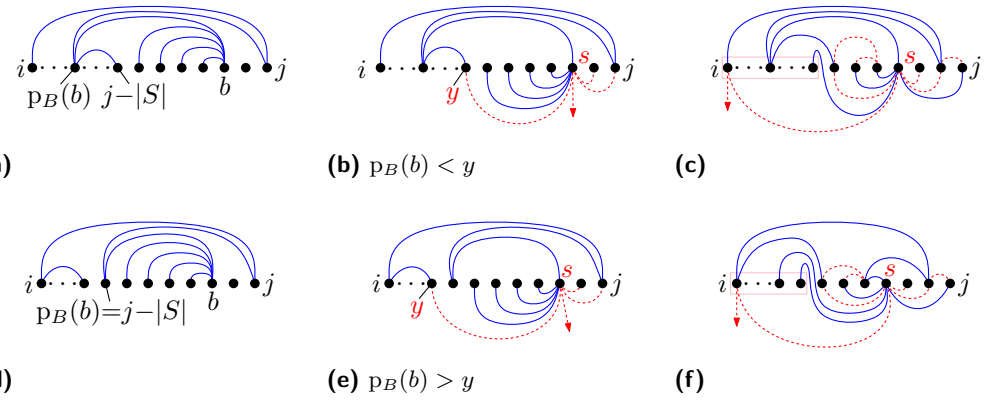


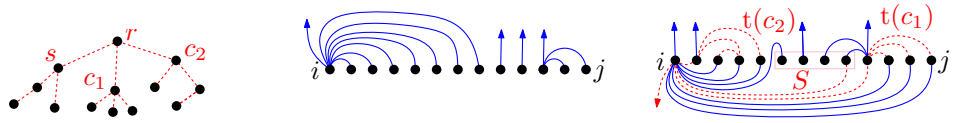
Figure 8 Explicit embedding of S in Case 1.3. The edge $\{p_B(b), j\}$ need not be present in B.

$|t_S(c_k)| - 1$ vertices to embed $t_S(c_k)$. For this, we use the leaves of B^*. Mirror B^* vertically into the bottom halfplane (Figure 8b). For each c_k, move $|t_S(c_k)|-1$ leaves of B^* immediately to the right of where c_k is embedded. This requires $|S| - \deg_S(s) - 1$ leaves in B^*, which is ensured by the first inequality from (4). In this way, each c_k has $|t_S(c_k)| - 1$ locally isolated vertices to its right, on which we embed $t_S(c_k) \setminus \{c_k\}$ explicitly in the *upper* halfplane (Figure 8c), using the algorithm from Section 3. This explicit embedding works also if $t_S(c_k)$ is a star. Any remaining leaves of B^* are placed immediately to the left of c_d. For every such leaf, the blue edge to s is routed so that it starts upwards and crosses the spine immediately to the left of $y = c_d$. In this way the remaining blue vertices in I form an interval.

Otherwise $p_B(b) > y$, which in this case by LSFR for $p_B(b)$ and (4) implies $p_B(b) = j-|S|$ (Figure 8d–8e). If we proceed as above, then the remaining blue vertices do not form an interval because $p_B(b)$ is separated from the rest. We address this problem as follows. Since $p_B(b) > y$, the procedure above embeds a child of s at $p_B(b) - 1$. As S is not a star, s has a non-leaf child v. We modify the procedure to embed v at $p_B(b) - 1$ and let $p_B(b)$ take the role of a leaf of B^* for the purpose of embedding $t_S(v)$ (Figure 8f).

After embedding S we complete the packing by recursively embedding R^- onto the interval $[i, j - |S|]$. This interval differs from the interval $[i, j - |S|]$ before embedding S only in that some (possibly empty) suffix of vertices has been replaced by locally isolated vertices. These vertices correspond to leaves of the star B^*, and since we embedded s at the center of B^*, they are all in edge-conflict with r. The "new" interval $[i, j - |S|]$ is not in edge-conflict with R^- by 3, $\{i, j\} \in E(B)$, and $\{i, b\} \notin E(B)$. Suppose towards a contradiction that $[i, j - |S|]$ is in degree-conflict with R^-. Then due to $\{i, j\} \in E(B)$ there is a central-star $D^* = B[i, j - |S|]\langle i\rangle$ rooted at i. By LSFR $B[i, j - |S|]\langle i\rangle$ was a central-star of size at least $|D^*|$ before embedding S. But then $[i, j - |S|]$ is in degree-conflict with R^- before embedding S, contrary to the assumption of Case 1. This completes the proof of Case 1.

Blue-star embedding. The procedure we used to embed S explicitly in Case 1.3 is useful in general. Hence we define this procedure, which we call *blue-star embedding*, in a more general form. Let $A = t_R(a)$ be a subtree rooted at a vertex a of R, and let $\sigma \in [i, j]$ be the center of a star $B^* = t_B(\sigma)$. Either σ is the root of $B\langle\sigma\rangle$ or $\tau := p_B(\sigma) \in [i, j]$. Denote by B^+ the subgraph of B induced by σ and all its neighbors (parent and children). Note that either $B^+ = B^*$ or $B^+ = B^* \cup \{\tau\}$. Put $d = \deg_A(a)$ and let $\varphi = (v_1, \ldots, v_d)$ be a sequence of elements from $B \setminus B^+$. Furthermore, suppose the following four conditions hold:



(a) R **(b)** B **(c)** blue-star

Figure 9 Handling a degree-conflict for R^- in Case 2.1.

BS1. a is not in edge-conflict with σ,
BS2. $|A| \leq |B^*| + \deg_A(a)$ and $|B^+| + \deg_A(a) \leq |R| - 1$,
BS3. at least one of $B \setminus (B^* \cup \varphi)$ or $B \setminus (B^+ \cup \varphi)$ forms an interval, and
BS4. if $B \setminus (B^* \cup \varphi)$ does not form an interval, then A is not a central-star, $v_1 = \tau \pm 1$ and $\{v_1, \tau\} \notin \mathrm{E}(B)$.

Then we can *blue-star embed A starting from σ with φ* such that a is embedded at σ, the children of a in A are embedded onto φ, the remaining vertices of A are embedded on $B^* \setminus \{\sigma\}$, and the remaining blue vertices form an interval $[i', j']$. A precise formulation with proof can be found in the full paper; here we provide some intuition only. 1 is necessary to put a at σ. 2 corresponds to (4). 3 ensures that the remaining blue vertices form an interval on which we can continue to work and 4 allows us to pull the trick of using $\mathrm{p}_B(\sigma)$ in place of a leaf of B^*, if applicable.

Case 2 $[i, j - |S|]$ is in degree-conflict with R^-. Then $B[i, j - |S|]\langle i\rangle$ is a central-star $B[i, x]$

$$\text{with} \quad \deg_{R^-}(r) + (x - i) \geq |R^-| \tag{5}$$

and $|B[i, x]| = x - i + 1 \geq 3$ by Lemma 7. We distinguish two cases.

Case 2.1 $B\langle i\rangle = B[i, x]$. Then $B\langle i\rangle \neq B\langle j\rangle$. If necessary, flip $B\langle i\rangle$ to put its center at i. If $B\langle j\rangle$ is a central-star on ≥ 3 vertices, then – if necessary – flip $B\langle j\rangle$ to put its root at j. We use a blue-star embedding for R^- starting from $\sigma = i$ with $\varphi = (x + 1, \ldots)$. As φ consists of $d := \deg_{R^-}(r)$ vertices, we have $[i, j] \setminus (B[i, x] \cup \varphi) = [x + d + 1, j]$. If $B[x + d + 1, j]\langle j\rangle$ is a central-star on ≥ 3 vertices, then use $\varphi = (j, x + 1, \ldots)$ instead (and note that $\uparrow(B[x + d + 1, j]\langle j\rangle) = j$).

In the notation of the blue-star embedding we have $B^* = B^+ = B[i, x]$. We need to show that the conditions for this embedding hold. 1 holds by 1 (for embedding R onto $[i, j]$). For 2 we have to show $|R^-| \leq |B^*| + \deg_{R^-}(r) \leq |R| - 1$. The first inequality holds by (5) and $|B^*| \geq x - i$. The second inequality holds due to 1 (for embedding R onto $[i, j]$), which implies $\deg_R(r) + (x - i) \leq |R| - 1$. As $|B[i, x]| = x - i + 1$ and $\deg_R(r) = \deg_{R^-}(r) + 1$, 2 follows. 3 is obvious by the choice of φ and 4 is trivial for $B^* = B^+$ due to 3. That leaves us with an interval $[i', j']$, where $j' \in \{j, j - 1\}$. We claim that $[j', i']$ is not in conflict with S.

To prove the claim we consider two cases. If $j' = j - 1$, then initally $B[x + d + 1, j]\langle j\rangle$ was a central-star on ≥ 3 vertices rooted at j. By the choice of φ a leaf of this star is at j' whose only neighbor in B is at $j \neq i = r$. Therefore $[j', i']$ is not in edge-conflict with S. As $B[j', i']\langle j'\rangle$ is an isolated vertex, by Lemma 7 there is no degree-conflict between $[j', i']$ and S, either, which proves the claim.

Otherwise, $j' = j$ and $B[j', i']\langle j'\rangle = B[x + d + 1, j]\langle j\rangle$ is not a central-star on ≥ 3 vertices. Therefore by Lemma 7 there is no degree-conflict between $[j', i']$ and S. In order to show that there is no edge-conflict, either, it is enough to show that $\uparrow(B[j', i']\langle j'\rangle)$ is not adjacent to $i = r$ in B. If $\uparrow(B[j', i']\langle j'\rangle) \neq j'$ this follows from 1SR. Otherwise $\uparrow(B[j', i']\langle j'\rangle) = j' = j$, and $\{i, j\} \notin \mathrm{E}(B)$ because $B\langle i\rangle$ is a star but B is not. Therefore the claim holds and we can complete the packing by recursively embedding S onto $[j', i']$.

Case 2.2 $B\langle i\rangle \neq B[i,x]$. By 1SR this means that $i = \mathrm{p}_B(x)$ and i has at least one more neighbor in $[i,j]\setminus B[i,x]$. Since by assumption $B[i,j-|S|]$ is not a star, we have $x \leq j-|S|-1$. Since $B[i,x]$ is a central-star and $x \leq j-|S|-1$, by LSFR for i the only neighbor of i in B outside of $B[i,x]$ is its parent $\mathrm{p}_B(i) \in [j-|S|+1, j]$. We claim that such a configuration is impossible. To prove the claim, note that $\mathrm{p}_B(i)$ has at least two children in $B[i,j-|S|]$ because $x \leq j-|S|-1$ and $\mathrm{p}_B(i) \geq j-|S|+1$. By LSFR, the corresponding subtrees have size at least $|B[i,x]| = x-i+1$, and so $|R| \geq |B[i,\mathrm{p}_B(i)]| \geq 2|B[i,x]|+1 \geq 2(|R^-| - \deg_{R^-}(r)+1)+1$, where the last inequality uses (5). Rewriting and using (1) yields

$$|R^-| \leq \frac{|R|-1}{2} + \deg_{R^-}(r) - 1 < \frac{|R|-1}{2} + \frac{|R^-|}{4}.$$

It follows that $|R^-| < \frac{2}{3}(|R|-1)$ and hence that $|S| > \frac{1}{3}(|R|-1)$. Since S is a smallest subtree of r in R, this means that r is binary in R and thus unary in R^-. This, finally, contradicts the degree-conflict for $[i,j-|S|]$ with R^- because $x < j-|S|$ and hence $\deg_{R^-}(r) + \deg_{B[i,x]}(i) = 1 + (x-i) < 1 + (j-|S|) - i = |R^-|$.

Case 3 $[j, j-|S|+1]$ is in degree-conflict with S and $[i,j-|S|]$ is not in degree-conflict with R^-. Then $B[j-|S|+1,j]\langle j\rangle$ is a central-star $Z = \mathrm{t}_B(z)$ with $|Z| \geq 3$ by Lemma 7 and

$$\deg_S(s) + \deg_Z(z) \geq |S|. \tag{6}$$

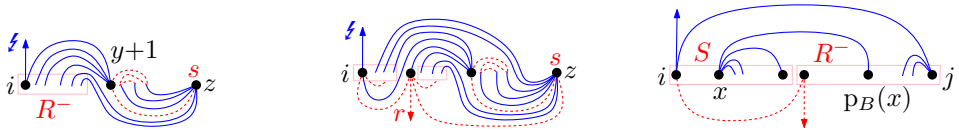
Case 3.1 $\{i,j\} \notin \mathrm{E}(B)$. Then we claim that we may assume $z = j$ and $Z = B\langle j\rangle$.

Let us prove this claim. If $z = j-|Z|+1$, then by 1SR it does not have any neighbor in $B\setminus Z$. Flipping $Z = B\langle j\rangle$ establishes the claim. Otherwise, $z = j$. Suppose that z has a neighbor $y \in B\setminus Z$. As z is the root of $Z = B[j-|S|+1,j]\langle j\rangle$, it does not have a neighbor in $[j-|S|+1, j-|Z|]$ and therefore $y \in [i+1, j-|S|]$. By LSFR and because $B[j-|S|+1,j]$ is not a star, $y = \mathrm{p}_B(z)$. In particular, since $|Z| \geq 3$, LSFR for y implies $\{y, y+1\} \notin \mathrm{E}(B)$. It follows that after flipping $B\langle j\rangle$ the resulting subtree $B[j-|S|+1,j]\langle j\rangle$ is not a central-star anymore and so there is no conflict for embedding S onto $[j, j-|S|+1]$ anymore. Therefore we can proceed as above in Case 1 (the conflict situation for R^- did not change because $B\langle i\rangle$ remains unchanged). Hence we may suppose that there is no such neighbor y of z, which establishes the claim.

We blue-star embed S starting from $\sigma = j = z$ with $\varphi = (j-|Z|, j-|Z|-1, \ldots)$. In the terminology of the blue-star embedding we have $B^* = B^+ = Z$. Let us argue that the conditions for the embedding hold. 1 is trivial because no neighbor of s is embedded yet. For 2 we have to show $|S| \leq |Z| + \deg_S(s) \leq |R|-1$. The first inequality holds by (6) and the second by $|S| \leq (|R|-1)/\deg_R(r) \leq (|R|-1)/2$, which implies $|Z| + \deg_S(s) \leq 2(|S|-1) \leq |R|-3$. 3 is obvious by the choice of φ and given $B^* = B^+$, 4 is trivial. That leaves us with an interval $[i', j']$, where $i' = i$.

The plan is to recursively embed R^- onto $[i,j']$. This works fine, unless $[i,j']$ and R^- are in conflict. So suppose that they are in conflict. Then there is a central-star $Y = B[i,j']\langle i\rangle$. Considering how φ consumes the vertices in I from right to left, Y appears as a part of some component of B, that is, $Y = B[i,y]$, for some $y \in [i, j-|S|]$.

We claim that $\uparrow(Y) = i$. To prove the claim, suppose to the contrary that $\uparrow Y = y = \mathrm{p}_B(i)$. Then by 1SR Y *is* a component of B. Thus, a degree-conflict contradicts the assumption of Case 3 that $[i, j-|S|]$ is not in degree-conflict with R^-, and an edge-conflict contradicts 1 for embedding R onto $[i,j]$ together with the fact that by 1SR y is not adjacent to any vertex outside of Y in B – in particular not to j, where s was placed. This proves the claim and, furthermore, that $\mathrm{p}_B(i) \in [y+1, j-|S|]$ and $\mathrm{p}_B(i)$ appears in φ.



(a) (b) (c)

Figure 10 (a)–(b): Relocating some leaves of the stars $Y = \mathrm{t}_B(y+1)$ and $Z = \mathrm{t}_B(z)$ in Case 3.1. One subtree of $\mathrm{t}_R(c)$ of R^- is embedded at i and the leaves to the right of i; all other subtrees of R^- are embedded to the right of r. Both from $y+1$ and from z we can route as many blue edges as desired to either of these "pockets". (c): Evading a degree-conflict for S in Case 3.2.1.

By 3 for embedding R onto $[i,j]$ and $\{i,j\} \notin \mathrm{E}(B)$ we know that $[i,j']$ and R^- are not in edge-conflict and so they are in degree-conflict. In particular, $\deg_Y(i) + \deg_{R^-}(r) \geq |R^-|$.

Undo the blue-star embedding. We claim $\{i, y+1\} \in \mathrm{E}(B)$. To prove the claim, suppose to the contrary that $\{i, y+1\} \notin \mathrm{E}(B)$. Then $\{y+1, \mathrm{p}_B(i)\} \in \mathrm{E}(B)$ because in B the vertex $y+1$ lies below the edge $\{i, \mathrm{p}_B(i)\}$. By LSFR the subtree of $\mathrm{p}_B(i)$ rooted at $y+1$ is at least as large as Y. Therefore,

$$|\mathrm{t}_B(p_B(i))| \geq 2|Y| + 1 = 2\deg_Y(i) + 3 \geq 2(|R^-| - \deg_{R^-}(r)) \geq \frac{3}{2}|R^-|,$$

where the last inequality uses (1). This is in contradiction to $p_B(i) \leq j - |S|$, which implies $|\mathrm{t}_B(p_B(i))| \leq |R^-|$. Therefore, the claim holds and $\{i, y+1\} \in \mathrm{E}(B)$.

Flip $B[i, y+1]$ and perform the blue-star embedding again. Although 1SR may be violated at $y+1$, this is of no consequence for the blue-star embedding. As $Y = B[i,j']\langle i\rangle = B[i,y]$, we know that $y+1$ appears in φ and so the offending vertex is not part of $[i,j']$ after the blue-star embedding. Furthermore, in this way we also get rid of the high-degree vertex of B that was at i initially so that the vertices in $[i,y] \subset [i,j']$ are isolated. In particular, i is isolated in $[i,j']$ and its only neighbor in B is at $y+1 \neq j$. Therefore, $[i,j']$ and R^- are not in conflict, unless $y+1 = \mathrm{p}_B(i)$ initially and $\mathrm{p}_B(i)$ is in edge-conflict with r.

In other words, it remains to consider the case $B\langle i\rangle = B[i, y+1] = \mathrm{t}_B(y+1)$ is a dangling star whose root i (at $y+1$ before flipping) is in edge-conflict with r (Figure 10a). Then $[i,j']$ is an independent set in B that consists of leaves of the two stars Y and Z plus the isolated vertex at i. Yet we cannot simply embed R^- using the algorithm from Section 3 because i is and j' may be in edge-conflict with r. Given that $|Y| \geq 3$ and φ gets to $y+1$ only, at least two leaves of Y remain in $[i,j']$ and so, in particular, $i+1$ is not in conflict with r. We explicitly embed R^- as follows (Figure 10b): place r at $i+1$ and a child c of r in R^- at i. Then collect $|\mathrm{t}_R(c)|$ leaves from Z and/or Y and put them right in between i and $i+1$. First – from left to right – the leaves of Z whose blue edges leave them upwards to bend down and cross the spine immediately to the right of the vertices of the red subtree rooted at $y+1$ (the leftmost subtree of S) and then reach z from below. Next come the leaves of Y whose blue edges to $y+1$ are drawn as arcs in the upper halfplane. In order to make room for those leaves, the blue edge $\{i, y+1\}$ is re-routed to leave i downwards to bend up and cross the spine immediately to the left of $i+1$ in order to reach $y+1$ from above. Using the algorithm from Section 3 we can now embed $\mathrm{t}_R(c)$ onto these leaves and any remaining subtrees of r can be embedded explicitly on the vertices $i+2, \ldots$ (ignoring the change of numbering caused by the just discussed repositioning of leaves).

Case 3.2 $\{i,j\} \in \mathrm{E}(B)$. Then $z = j$ because $j - |Z| + 1$ is enclosed by $\{i,j\}$ and therefore cannot be the root of Z. Moreover, $\uparrow(B) = i$ by LSFR and since B is not a star. By LSFR j does not have any child in $B \setminus Z$ and as $B[j - |S| + 1, j]$ is not a star, $Z \subseteq B[j - |S| + 2, j]$.

In particular, j is not adjacent to any vertex in $B[i+1, j-|S|+1]$. We provisionally place s at any vertex in $[i+1, j-|R^-|]$, say, at $j-|R^-|$. Then $[j, j-|R^-|+1]$ is not in edge-conflict with R^-. We claim that it is not in degree-conflict, either. As Z is a star on $|Z| \leq |S|-1$ vertices, by Lemma 6 we have $\deg_{R^-}(r) + \deg_Z(z) \leq \deg_{R^-}(r) + |S| - 2 \leq |R^-| - 2$ and the claim follows. We recursively embed R^- onto $[j, j-|R^-|+1]$, treating all local roots of B other than j as in conflict with r. It remains to recursively embed S onto $[j-|R^-|, i]$.

Suppose towards a contradiction that $[j-|R^-|, i]$ is in conflict with S. Then there is a central-star $X = B[x, j-|R^-|] = B[i, j-|R^-|]\langle j-|R^-|\rangle$. Due to $\{i,j\} \in \mathrm{E}(B)$ and 1SR we have $\uparrow(X) = x$ and $\mathrm{p}_B(x) > j-|R^-|$. Together with LSFR for j it follows that $\mathrm{p}_B(x) \in [j-|R^-|+1, j-|Z|]$. Due to the conflict setting for embedding R^-, i is the only vertex in $[j-|R^-|, i]$ that may be in edge-conflict with s. As X is a central-star and $\uparrow(B) = i$, we cannot have $x = i$ because then B would be a star. It follows that $x > i$ and so $[j-|R^-|, i]$ is not in edge-conflict with S. Therefore $[j-|R^-|, i]$ and S are in degree-conflict. Then $|X| \geq 3$ by Lemma 7 and

$$\deg_S(s) + \deg_X(x) \geq |S|. \tag{7}$$

Depending on $\mathrm{p}_B(x)$ we consider two final subcases.

Case 3.2.1 $\mathrm{p}_B(x) \in [j-|R^-|+2, j-|Z|]$ (Figure 10c). Then the edge $\{x, \mathrm{p}_B(x)\}$ encloses $j-|R^-|+1$ so that, in particular, $\{i, j-|R^-|+1\} \notin \mathrm{E}(B)$. We provisionally place s at $i = \uparrow(B[i, j-|R^-|]\langle i\rangle)$ and claim that $[j-|R^-|+1, j]$ and R^- are not in conflict.

To prove the claim, consider $W^* := B[j-|R^-|+1, j]\langle j-|R^-|+1\rangle$ and suppose it is a central-star. (If it is not, then we are done.) If $\uparrow(W^*) > j-|R^-|+1$, then by 1SR and $\{x, \mathrm{p}_B(x)\} \in \mathrm{E}(B)$ we have $\mathrm{p}_B(\uparrow(W^*)) = x$, in contradiction to LSFR for x. Therefore $\uparrow(W^*) = j-|R^-|+1$. In order for $j-|R^-|+1$ to be the local root for W^* in the presence of $\{x, \mathrm{p}_B(x)\} \in \mathrm{E}(B)$, it follows that $\mathrm{p}_B(j-|R^-|+1) = x$ and so by 1SR $|W^*| = 1$. Therefore by Lemma 7 there is no degree-conflict between $[j-|R^-|+1, j]$ and R^-. As $\{x, \mathrm{p}_B(x)\} \in \mathrm{E}(B)$ prevents any connection in B from $j-|R^-|+1$ to i and to vertices outside of $[i, j]$, there is no edge-conflict between $[j-|R^-|+1, j]$ and R^-, either. This proves the claim. Recursively embed R^- onto $[j-|R^-|+1, j]$. Recall that $\uparrow(B) = i$. There is no conflict for embedding S onto $[i, j-|R^-|]$ since $\{i, r\} \notin \mathrm{E}(B)$ and $B[i, j-|R^-|]\langle i\rangle$ is not a central-star of size at least 2 by LSFR at i. Finish the packing by recursively embedding S onto $[i, j-|R^-|]$.

Case 3.2.2 $\mathrm{p}_B(x) = j-|R^-|+1$. Then by 1SR $\mathrm{p}_B(x)$ is the only neighbor of x outside of X in B. We provisionally place r at j and employ a blue-star embedding for S, starting from $\sigma = x$ with $\varphi = (i, \ldots)$, that is, φ takes vertices from left to right, skipping over $[x, \mathrm{p}_B(x)]$. Let us argue that the conditions for the blue-star embedding hold.

In the terminology of the blue-star embedding we have $B^* = X$ and $B^+ = X \cup \{\mathrm{p}_B(x)\}$. 1 holds because $\{x, j\} \notin \mathrm{E}(B)$. For the first inequality of 2 we have to show $|S| \leq |X| + \deg_S(s)$, which is immediate from (7). For the second inequality of 2 we have to show $|X|+1+\deg_S(s) \leq |I|-1$. This follows from $|X|+1+\deg_S(s) \leq |S| + (|S|-1) \leq |R^-| + |S| - 1 = |I| - 1$. Regarding 3 note that in φ we take the vertices of $B \setminus B^+$ from left to right. As there are not enough vertices in $[i, x-1]$ to embed the neighbors of s (which causes the degree-conflict), φ reaches beyond $\mathrm{p}_B(x)$ and so $B \setminus (B^+ \cup \varphi)$ forms an interval. In particular, φ includes $\mathrm{p}_B(x)+1$ and we may simply move $\mathrm{p}_B(x)+1$ to the front of φ, establishing the second condition in 4. Regarding the remaining two conditions in 4 note that S is not a star by assumption and that $\mathrm{p}_B(x)+1$ is not a neighbor of x in B because $\mathrm{p}_B(x)$ is the only neighbor of x outside of X.

Therefore, we can blue-star embed S as claimed, which leaves us with an interval $[i', j']$, where $j = j'$. As $\{x, j\} \notin \mathrm{E}(B)$ and j is not the local root of B (i is), there is no edge-conflict

between $[j', i']$ and R^-. As there is no degree-conflict between $[j, j - |R^-| + 1]$ and R^- and the number of neighbors of j in $B[i', j']$ can only decrease compared to $B[j - |R^-| + 1, j]$ (if they appear in φ), there is no degree-conflict between $[j', i']$ and R^-, either. Therefore, we can complete the packing by embedding R^- onto $[j', i']$ recursively. ◀

References

1 J. Akiyama and V. Chvátal. Packing paths perfectly. *Discrete Math.*, 85(3):247–255, 1990.
2 P. Braß, E. Cenek, C. A. Duncan, A. Efrat, C. Erten, D. Ismailescu, S. G. Kobourov, A. Lubiw, and J. S. B. Mitchell. On simultaneous planar graph embeddings. *Comput. Geom.*, 36(2):117–130, 2007.
3 Y. Caro and R. Yuster. Packing graphs: The packing problem solved. *Electr. J. Combin.*, 4(1), 1997.
4 D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Inform. Process. Lett.*, 51(4):207–211, 1994.
5 P. Erdős. Extremal problems in graph theory. In M. Fiedler, editor, *Theory of Graphs and its Applications*, pages 29–36. Academic Press, 1965.
6 A. Frank and Z. Szigeti. A note on packing paths in planar graphs. *Math. Program.*, 70(2):201–209, 1995.
7 F. Frati. Planar packing of diameter-four trees. In *Proc. 21st Canad. Conf. Comput. Geom.*, pages 95–98, 2009.
8 F. Frati, M. Geyer, and M. Kaufmann. Planar packings of trees and spider trees. *Inform. Process. Lett.*, 109(6):301–307, 2009.
9 A. García, C. Hernando, F. Hurtado, M. Noy, and J. Tejel. Packing trees into planar graphs. *J. Graph Theory*, pages 172–181, 2002.
10 M. Geyer, M. Hoffmann, M. Kaufmann, V. Kusters, and Cs. D. Tóth. Planar packing of binary trees. In *Proc. 13th Algorithms and Data Struct. Sympos.*, volume 8037 of *Lecture Notes Comput. Sci.*, pages 353–364. Springer, 2013.
11 D. Gonçalves. Edge partition of planar graphs into two outerplanar graphs. In *Proc. 37th Annu. ACM Sympos. Theory Comput.*, pages 504–512. ACM Press, 2005.
12 A. Gyárfás and J. Lehel. Packing trees of different order into K_n. In *Combinatorics*, volume 18 of *Colloq. Math. Soc. János Bolyai*, pages 463–469. North Holland, 1978.
13 S. M. Hedetniemi, S. T. Hedetniemi, and P. J. Slater. A note on packing two trees into K_N. *Ars Combin.*, 11:149–153, 1981.
14 M. Maheo, J.-F. Saclé, and M. Woźniak. Edge-disjoint placement of three trees. *European J. Combin.*, 17(6):543–563, 1996.
15 P. Mutzel, T. Odenthal, and M. Scharbrodt. The thickness of graphs: A survey. *Graphs and Combinatorics*, 14(1):59–73, 1998.
16 C. St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of The London Mathematical Society*, s1-36:445–450, 1961.
17 Y. Oda and K. Ota. Tight planar packings of two trees. In *European Workshop on Computational Geometry*, pages 215–216, 2006.
18 W. Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.
19 S.K. Teo and H.P. Yap. Packing two graphs of order n having total size at most $2n - 2$. *Graphs and Combinatorics*, 6(2):197–205, 1990.
20 W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of the London Mathematical Society*, s1-36(1):221–230, 1961.

Crossing Number is Hard for Kernelization*

Petr Hliněný[1] and Marek Derňár[2]

1 **Faculty of Informatics, Masaryk University Brno, Brno, Czech Republic**
`hlineny@fi.muni.cz`
2 **Faculty of Informatics, Masaryk University Brno, Brno, Czech Republic**
`m.dernar@gmail.com`

Abstract

The graph crossing number problem, $\mathrm{cr}(G) \leq k$, asks for a drawing of a graph G in the plane with at most k edge crossings. Although this problem is in general notoriously difficult, it is fixed-parameter tractable for the parameter k [Grohe]. This suggests a closely related question of whether this problem has a *polynomial kernel*, meaning whether every instance of $\mathrm{cr}(G) \leq k$ can be in polynomial time reduced to an equivalent instance of size polynomial in k (and independent of $|G|$). We answer this question in the negative. Along the proof we show that the tile crossing number problem of twisted planar tiles is NP-hard, which has been an open problem for some time, too, and then employ the complexity technique of cross-composition. Our result holds already for the special case of graphs obtained from planar graphs by adding one edge.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations, F.1.3 [Complexity Measures and Classes] Reducibility and completeness

Keywords and phrases crossing number, tile crossing number, parameterized complexity, polynomial kernel, cross-composition

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.42

1 Introduction

We refer to Sections 2,3 for detailed formal definitions. Briefly, the *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of pairwise edge crossings in a drawing of G in the plane. Finding the crossing number of a graph is one of the most prominent hard optimization problems in geometric graph theory [10] and is NP-hard already in very restricted cases, e.g., for cubic graphs [12], and for graphs with prescribed edge rotations [16]. Concerning approximations, there exists $c > 1$ such that the crossing number cannot be approximated within the factor c in polynomial time [5]. Moreover, the following very special case of the problem is still hard – a result that greatly inspired our paper:

▶ **Theorem 1** (Cabello and Mohar [6]). *Let G be an almost-planar graph, i.e., G having an edge $e \in E(G)$ such that $G \setminus e$ is planar (called also* near-planar *in [6]). Let $k \geq 1$ be an integer. Then it is NP-complete to decide whether $\mathrm{cr}(G) \leq k$.*

On the other hand, it has been shown that the problem is *fixed-parameter tractable* when parameterized by itself: one can decide whether $\mathrm{cr}(G) \leq k$ in quadratic (Grohe [11]) and even linear (Kawarabayashi–Reed [13]) time while having k fixed. Fixed-parameter tractability (FPT) is closely related to the concept of so called *kernelization*. In fact, one can easily show

* This research was supported by the Czech Science Foundation project No. 14-03501S.


© Petr Hliněný and Marek Derňár;
licensed under Creative Commons License CC-BY

that a (decidable) problem $\mathcal{A}$ parameterized by an integer k is FPT if, and only if, every instance of $\mathcal{A}$ can be in polynomial time reduced to an equivalent instance (the *kernel*) of size bounded only by some function of k. This function of k, bounding the kernel size, may in general be arbitrarily huge. Though, the really interesting case is when the kernel size may be bounded by a polynomial function of k (a *polynomial kernel*).

The nature of the methods used in [11, 13], together with the recent great advances in algorithmic graph minors theory, might suggest that the crossing number problem $\mathrm{cr}(G) \leq k$ should have a polynomial kernel in k, as many related FPT problems do. This question was raised as open, e.g., at WorKer 2015 [unpublished]. Polynomial kernels for some special crossing number problem instances were constructed before, e.g., in [1]. The general result is, however, very unlikely to hold as our main result claims:

▶ **Theorem 2.** *Let G be an* almost-planar *graph, i.e., G having an edge $e \in E(G)$ such that $G \setminus e$ is planar. Let $k \geq 1$ be an integer. The crossing number problem, asking if $\mathrm{cr}(G) \leq k$ while parameterized by k, does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

In order to prove Theorem 2, we use the technique of *cross-composition* [2]. While its formal description is postponed till Section 3, here we very informally outline the underlying idea of cross-composition. Imagine we have an NP-hard language $\mathcal{L}$ such that we can "OR-cross-compose" an arbitrary collection of instances $x_1, x_2, \ldots, x_t$ of $\mathcal{L}$ into the crossing number problem $\mathrm{cr}(G_0) \leq k_0$ for suitable G_0 and k_0 efficiently depending on $x_1, x_2, \ldots, x_t$. By the words "OR-cross-compose" we mean that $\mathrm{cr}(G_0) \leq k_0$ holds if and only if $x_i \in \mathcal{L}$ for some $1 \leq i \leq t$ (informally, $x_1 \in \mathcal{L}$ OR $x_2 \in \mathcal{L}$ OR $\ldots$). Now assume we could always reduce a crossing number instance $\langle G, k \rangle$ into an equivalent instance of size $p(k)$ where p is a polynomial. Then, for the instance $\langle G_0, k_0 \rangle$ and suitable t such that $p(k_0) << t \approx |G_0| << 2^{|x_i|}$, such a reduction effectively means that we should somehow decide many of the t instances $x_i \in \mathcal{L}$ in time polynomial in $|G_0|$ (which is $<< 2^{|x_i|}$). The latter sounds highly unlikely [9] in the complexity theory.

The task is to find a suitable NP-hard language $\mathcal{L}$ for the aforementioned construction. While the ordinary crossing number problem is not suitable for cross-composition (roughly, since the crossing numbers of disjoint instances sum up together), a helping hand is given by the concept of the *tile crossing number* [17], defined in detail in Section 2.

Informally, a *tile* is a graph T with two disjoint sequences of vertices defining the left and right walls of T. A *tile drawing* is a drawing of T inside a rectangle such that the walls of T lie respectively on the left and right sides of this rectangle. A tile T is planar if T admits a tile drawing without crossings, and T is *twisted planar* if T becomes a planar tile after inverting (upside-down) one of the walls. As observed by Schaefer [20], the tile crossing number problem is NP-hard by a trivial reduction from ordinary crossing number, but we need much more. In order to embed the tile crossing number problem in a cross-composition construction, which will be realized as a concatenation of the tile instances across their respective walls, we shall use only twisted planar tiles. See Figure 1. The underlying idea which makes the cross-composition work, is that only one of the tile instances is drawn twisted in the concatenation and all the other contribute no crossings.

Hence the proof of Theorem 2 would be finished, modulo technical details, if we show that the tile crossing number problem of twisted planar tiles is NP-hard. This particular question seems to have been latently considered in the crossing number community for the past several years, and it is still open nowadays to our best knowledge. We provide the following affirmative answer by adapting a construction from the proof [6] of Theorem 1:

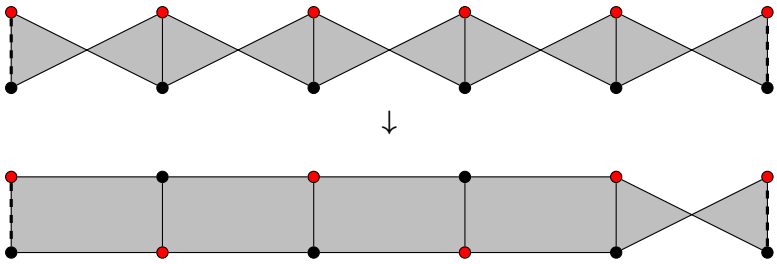


Figure 1 Schematic concatenation of an odd number of twisted planar tiles; in fact, only one (and an arbitrary one) of the tiles needs to be drawn twisted in this case.

▶ **Theorem 3** (Corollary 12). *Let T be a twisted planar tile and $k \geq 1$ an integer. Then it is NP-complete to decide whether there exists a tile drawing of T with at most k edge crossings. Furthermore, the same holds if both the walls of T are of size two and there exists an edge $e \in E(T)$ such that $T \setminus e$ is a planar tile.*

Paper organization. We provide the necessary formal definitions of the aforementioned concepts from crossing numbers and parameterized complexity in Sections 2,3. Then we prove Theorem 3 in Section 4, and provide technical claims useful for the next cross-composition construction in Section 5. Finally, we summarize the paper and present some additional ideas in Section 6.

2 Crossing numbers

We consider multigraphs by default, even though we could always subdivide parallel edges in order to make the 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 curves joining their endvertices. It is required that no edge passes through a vertex, and no three edges cross in a common point.

▶ **Definition 4** (crossing number). The *crossing number* $\mathrm{cr}(G)$ of a graph G is the minimum number of crossing points of edges in a drawing of G in the plane.

Hence, a graph G is planar if and only if $\mathrm{cr}(G) = 0$. Note that the crossing number is invariant under subdividing edges of G.

A useful concept in crossing numbers research are tiles. They were used already by Kochol [14] and Richter–Thomassen [19], although they were formalized only later in the work of Pinnontoan and Richter [17, 18]. So far, primary use of the tile concept in crossing numbers research concerned study of so called crossing-critical graphs, as can be seen also in recent papers such as [3, 4]. Here we will use tiles in a rather different way. We briefly sketch the necessary terms as follows.

A *tile* is a triple $T = (G, \lambda, \rho)$ where $\lambda, \rho \in V(G)^*$ are two disjoint sequences of distinct vertices of G, called the *left and right wall* of T, respectively. A *tile drawing* of T is a drawing of the underlying graph G in the unit square such that the vertices of λ occur in this order on the left side of the square and those of ρ in this order on the right side of it. The *tile crossing number* $\mathrm{tcr}(T)$ of a tile T is the minimum number of crossing points of edges over all tile drawings of T. The *right-inverted* tile $T^{\updownarrow}$ is the tile $(G, \lambda, \bar{\rho})$ and the *left-inverted* tile ${}^{\updownarrow}T$ is $(G, \bar{\lambda}, \rho)$, where $\bar{\lambda}$ and $\bar{\rho}$ denote the inverted sequences of λ, ρ.

For simplicity, in this brief exposition, we shall assume that all tiles involved in one construction satisfy $|\lambda| = |\rho| = w$ for suitable $w \geq 2$ (though, a more general treatment is obviously possible). The *join of two tiles* $T = (G, \lambda, \rho)$ and $T' = (G', \lambda', \rho')$ is defined as the tile $T \otimes T' := (G'', \lambda, \rho')$, where G'' is the graph obtained from the disjoint union of G and G', by identifying $\rho(i)$ with $\lambda'(i)$ for $i = 1, \ldots, w$. Since the operation $\otimes$ is associative, we can safely define the join of a sequence of tiles $\mathcal{T} = (T_1, T_2, \ldots, T_m)$ as the tile given by $\otimes\mathcal{T} = T_1 \otimes T_2 \otimes \ldots \otimes T_m$.

A tile $T = (G, \lambda, \rho)$ is *planar* if $\mathrm{tcr}(T) = 0$, and T is *twisted planar* if $\mathrm{tcr}(T^{\updownarrow}) = 0$ (which is clearly equivalent to $\mathrm{tcr}({}^{\updownarrow}T) = 0$). We briefly illustrate these definitions (also Figure 1):

▶ **Example 5.** Let $\mathcal{T} = (T_1, T_2, \ldots, T_m)$ be a sequence of twisted planar tiles T_i, $i = 1, \ldots, m$. Then $\mathrm{tcr}(\otimes\mathcal{T}) = 0$ if m is even, and $\mathrm{tcr}(\otimes\mathcal{T}) \leq \min_{i\in\{1,\ldots,m\}} \mathrm{tcr}(T_i)$ otherwise.

Finally, the following is a useful artifice in crossing numbers research. In a *weighted* graph, each edge is assigned a positive number (the *weight, or thickness* of the edge). Now the crossing number is defined as in the ordinary case, but a crossing point between edges e_1 and e_2, say of weights t_1 and t_2, contributes $t_1 \cdot t_2$ to the result. In the case of integer weights, this extension can be easily seen equivalent to the unweighted setting as follows:

▶ **Proposition 6** (folklore). *Let G be an integer-weighted graph, $F \subseteq E(G)$, and G^+ be constructed from G via replacing each edge $e \in F$ of weight t with a bunch of t parallel edges of weight 1. Then $\mathrm{cr}(G) = \mathrm{cr}(G^+)$. Moreover, if G is the graph of a tile T and T^+ is the corresponding tile based on G^+, then $\mathrm{tcr}(T) = \mathrm{tcr}(T^+)$.*

3 Parameterized complexity and kernelization

Here we introduce the relevant concepts of parameterized complexity theory. For more details, we refer to textbooks [7, 8]. Let Σ be a finite alphabet. A parameterized problem over Σ is a language $\mathcal{A} \subseteq \Sigma^* \times \mathbb{N}$. An instance of $\mathcal{A}$ is thus a pair $\langle x, k\rangle$ where x is the input and $k \geq 0$ an (integer) parameter. In our case, e.g., $\langle G, k\rangle$ is the crossing number instance "$\mathrm{cr}(G) \leq k$". A parameterized problem is *fixed-parameter tractable* (FPT) if every instance $\langle x, k\rangle$ can be solved in time $f(k) \cdot |x|^c$, where f is a computable function and c is a constant.

A hot research direction in the area of parameterized complexity of the past decade is that of kernelization. A *kernelization* for a parameterized problem $\mathcal{A}$ is an algorithm that takes an instance $\langle x, k\rangle$ of $\mathcal{A}$ and, in time polynomial in $|x| + k$, maps $\langle x, k\rangle$ to an equivalent instance $\langle x', k'\rangle$ of $\mathcal{A}$ such that $|x'| + k' \leq f(k)$ where f is a computable function. The output $\langle x', k'\rangle$ is called the *kernel*. We say that $\mathcal{A}$ has a *polynomial kernel* if there is a kernelization for $\mathcal{A}$ such that f is a polynomial. Every fixed-parameter tractable problem admits a kernel, but not necessarily a polynomial kernel.

We now describe the basic OR-cross-composition framework of [2]. An equivalence relation $\sim$ on Σ^* is called a polynomial equivalence if, for any $x, y \in \Sigma^*$, we can decide in polynomial time whether $x \sim y$ and, moreover, on any finite $S \subseteq \Sigma^*$ the relation $\sim$ defines a number of equivalence classes which is polynomially bounded in the size of a largest element of S. For our purpose, $\sim$ will group together the tile crossing number instances of the same objective value k.

▶ **Definition 7** (OR-cross-composition). Let $\mathcal{L} \subseteq \Sigma^*$ be a language, $\sim$ be a polynomial equivalence relation on Σ^*, and let $\mathcal{A} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. An OR-*cross-composition* of $\mathcal{L}$ into $\mathcal{A}$ is an algorithm that, given t instances $x_1, x_2, \ldots, x_t \in \Sigma^*$ of $\mathcal{L}$ such that $x_1 \sim x_2 \sim \cdots \sim x_t$;

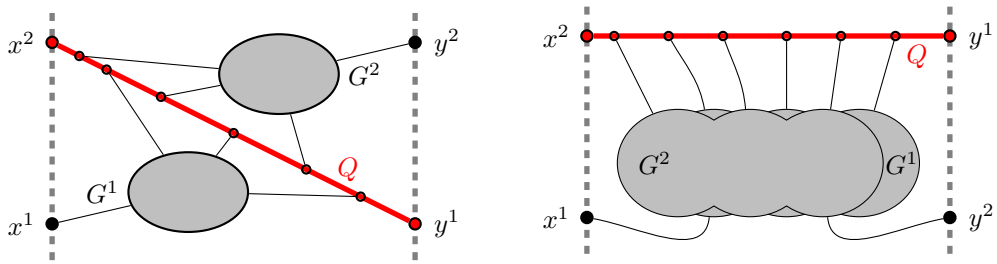


Figure 2 A diagonally separated tile and a possible drawing of the corresponding right-inverted tile. The underlying graph of this tile contains two vertex-disjoint subgraphs G^1, G^2 such that $V(G^1) \cup V(G^2) = V(G) \setminus \{x^2, y^1\}$, and their drawings "overlay" each other on the right.

- in time polynomial in $|x_1| + \ldots |x_t|$ it outputs an instance $\langle y_0, k_0 \rangle \in \Sigma^* \times \mathbb{N}$ such that k_0 is polynomially bounded in $\max_i |x_i| + \log t$, and
- $\langle y_0, k_0 \rangle \in \mathcal{A}$ if and only if $x_i \in \mathcal{L}$ for some $1 \leq i \leq t$.

▶ **Theorem 8** (Bodlaender, Jansen and Kratsch [2]). *If an NP-hard language $\mathcal{L}$ has an* OR*-cross-composition into the parameterized problem $\mathcal{A}$, then $\mathcal{A}$ does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

We remark in passing that the full claim of [2] is even stronger than stated Theorem 8, and in particular it also excludes the existence of a so-called polynomial compression of $\mathcal{A}$.

4 Twisted planar tiles

For the purpose of our proof, we are especially interested in the following kind of integer-weighted planar tiles. See Figure 2 for an illustration.

▶ **Definition 9** (diagonally separated tile). Consider an integer-weighted planar tile $T = (G, \lambda, \rho)$ where the walls are $\lambda = (x^1, x^2)$ and $\rho = (y^1, y^2)$ for some distinct $x^1, x^2, y^1, y^2 \in V(G)$. We say that T is *diagonally separated* if we can write $G = G^1 \cup G^2 \cup Q$ such that

- G^1, G^2 are vertex-disjoint subgraphs of G such that $V(G^1) \cup V(G^2) = V(G) \setminus \{x^2, y^1\}$,
- $E(Q) = E(G) \setminus (E(G^1) \cup E(G^2))$, $x^1, y^2 \notin V(Q)$, and Q is a "thick" path from x^2 to y^1 having each edge of weight $t \geq w_1 \cdot w_2 + 1$ where w_i is the sum of weights of all the edges of the subgraph $G^i \setminus V(Q)$,
- G is connected and both $G^1 \setminus V(Q)$ and $G^2 \setminus V(Q)$ are connected subgraphs, and no edge of $G^1 \cup G^2$ has both ends in $V(Q) \cup \{x^1, y^2\}$,
- $x^1 \in V(G^1) \setminus V(Q)$ and $y^2 \in V(G^2) \setminus V(Q)$, both the vertices x^1, y^2 are of degree one in G and the two incident edges have weight 1.

Twisted diagonally separated planar tiles have the suitable "or-composability" property:

▶ **Lemma 10.** *Let $\mathcal{T} = (T_1^{\updownarrow}, T_2^{\updownarrow}, \ldots, T_m^{\updownarrow})$ be a sequence of tiles such that, for $i = 1, \ldots, m$, T_i is a diagonally separated planar tile. Let $U := \otimes\mathcal{T}$ if m is odd, and $U := (\otimes\mathcal{T})^{\updownarrow}$ otherwise. Then $\mathrm{tcr}(U) = \min_{i \in \{1,\ldots,m\}} \mathrm{tcr}(T_i^{\updownarrow})$.*

Proof. Let the underlying graph of T_i be $G_i^1 \cup G_i^2 \cup Q_i$, as anticipated by Definition 9, and let t_i be the weight of $E(Q_i)$. Let (x_i^1, x_i^2), (y_i^2, y_i^1) be the left and right walls, respectively, of $T_i^{\updownarrow}$. By the definition of join $\otimes$, $y_i^1 = x_{i+1}^2$ and hence $Q := Q_1 \cup \cdots \cup Q_m$ is a path from x_1^2

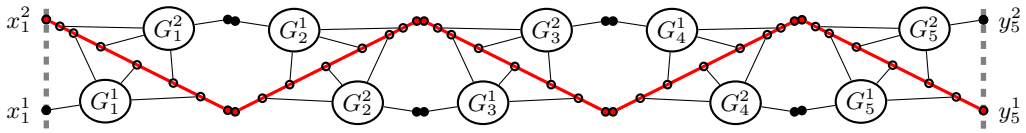


Figure 3 The planar tile $U^{\updownarrow}$ where U for $m = 5$ is from the statement of Lemma 10.

to y_m^1. Similarly, $y_i^2 = x_{i+1}^1$, and so $H_i := G_i^2 \cup G_{i+1}^1$ is a connected component of $U \setminus V(Q)$ for $i = 1, \ldots, m-1$. See Figure 3. For simplicity, we let $H_0 := G_1^1$ and $H_m := G_m^2$ which are also components of $U \setminus V(Q)$.

It clearly holds $\mathrm{tcr}(U) \leq \min_{i \in \{1,\ldots,m\}} \mathrm{tcr}(T_i^{\updownarrow})$. Furthermore, we claim that $t_i > \mathrm{tcr}(T_i^{\updownarrow})$ and so $t_i > \mathrm{tcr}(U)$ for each $i = 1, \ldots, m$. Since T_i is planar, each of G_i^1, G_i^2 has a plane embedding in which the vertices adjacent to $V(Q) \cup \{x_i^1, y_i^2\}$ lie on the outer face. Consequently, there is a tile drawing of $T_i^{\updownarrow}$ with each of G_i^1, G_i^2 plane and crossings only between the edges of G_i^1 and of G_i^2 that are not incident to Q. See Figure 2 right. By standard arguments, we may assume that no two edges cross more than once in this drawing and so $\mathrm{tcr}(T_i^{\updownarrow}) \leq w_i^1 \cdot w_i^2 \leq t_i - 1$ where w_i^j is the sum of weights of all the edges of $G_i^j \setminus V(Q)$.

From $t_i > \mathrm{tcr}(U)$ for $i = 1, \ldots, m$ we get that no edge of Q is ever crossed in an optimal tile drawing of U. We may hence properly define, in any optimal tile drawing of U and for each subgraph H_i, whether whole H_i lies (is drawn) *above* or *below* Q. We aim to show that there always exists $i \in \{1, \ldots, m\}$ such that H_{i-1}, H_i are drawn on the same side of Q, either both above or both below Q.

Assume the contrary. Then H_0 is drawn below Q by the left wall (x_1^1, x_1^2) of U. Next, H_1 is drawn above, H_2 below, ..., and finally, H_m should be drawn above U if m is odd and below U otherwise. That is exactly the opposite position to what is requested by the right wall of U which is (y_m^2, y_m^1) if m is odd and (y_m^1, y_m^2) otherwise, a contradiction.

So, H_{i-1} and H_i are drawn on the same side of Q for some $i \in \{1, \ldots, m\}$. First assume that $1 < i < m$. By supposed connectivity of G_{i-1} there is a path $P^1 \subseteq G_{i-1}^2$ from $x_i^1 = y_{i-1}^2$ to an internal vertex of Q_{i-1}, and similarly, there is a path $P^2 \subseteq G_{i+1}^1$ from $y_i^2 = x_{i+1}^1$ to an internal vertex of Q_{i+1} by connectivity of G_{i+1}. Let D be the drawing obtained from a considered optimal tile drawing of U restricted to T_i, by prolonging the single weight-1 edge incident with x_i^1 along P^1 and the single edge incident with y_i^2 along P^2. Since whole Q is uncrossed, the paths Q_{i-1}, Q_{i+1} can play the role of the left and right wall of D, and hence D is a valid tile drawing of T_i having no more crossings than $\mathrm{tcr}(U)$.

If $i = 1$ or $i = m$, then we directly use the left or the right wall of U in the previous argument. Consequently, $\min_{i \in \{1,\ldots,m\}} \mathrm{tcr}(T_i^{\updownarrow}) \leq \mathrm{tcr}(U)$ and the proof is finished. ◀

The last step of this section is to prove that the tile crossing number problem is NP-hard for twisted diagonally separated planar tiles. Due to their similarity to intermediate steps in the paper [6], it is no surprise that we can easily derive hardness using the same means; from NP-hardness of the so called anchored crossing number.

An *anchored graph* [6] is a triple (G, A, σ), where G is a graph, $A \subseteq V(G)$ are the anchor vertices and σ is a cyclic ordering (sequence) of A. An *anchored drawing* of (G, A, σ) is a drawing of G in a closed disc Δ such that the vertices of A are placed on the boundary of Δ in the order specified by σ, and the rest of the drawing lies in the interior of D. The *anchored crossing number* $\mathrm{acr}(G, A, \sigma)$, or shortly $\mathrm{acr}(G)$, is the minimum number of pairwise edge crossings in an anchored drawing of (G, A, σ). A *planar anchored graph* is an anchored graph that has an anchored drawing without crossings. Any subgraph $H \subseteq G$ naturally defines the corresponding anchored subgraph $\big(H, A \cap V(H), \sigma \restriction V(H)\big)$.

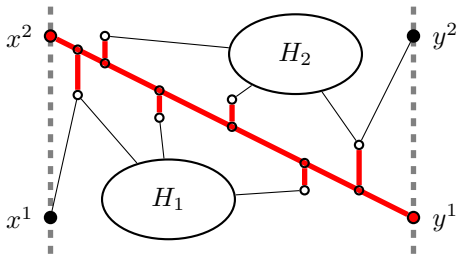


Figure 4 Constructing a twisted diagonally separated planar tile; proof of Corollary 12.

▶ **Theorem 11** (Cabello and Mohar [6][1]). *Let G be an anchored graph that can be decomposed into two vertex-disjoint connected planar anchored subgraphs. Let $k \geq 1$ be an integer. Then it is NP-complete to decide whether* $\mathrm{acr}(G) \leq k$.

▶ **Corollary 12.** *Let T be a diagonally separated planar tile, and $k \geq 1$ be an integer. Then it is NP-complete to decide whether* $\mathrm{tcr}(T^{\updownarrow}) \leq k$.

Note that twisted diagonally separated planar tiles satisfy all the assumptions of Theorem 3. In particular, if f denotes the edge incident to x^1 in T then both T and $T^{\updownarrow} \setminus f$ are planar tiles. Since the edge weights in the reduction are polynomial, the unweighted version in Theorem 3 follows immediately via Proposition 6.

Proof. Membership of the "$\mathrm{tcr}(T^{\updownarrow}) \leq k$" problem in NP is trivial by a folklore argument; we may guess the at most k crossings of an optimal drawing, replace those by new vertices and test planarity of the new tile. We provide the hardness reduction from Theorem 11. Let (G, A, σ) be an anchored graph anticipated in Theorem 11. Then G is a disjoint union of two connected components H_1 and H_2 where each of the corresponding anchored subgraphs H_1, H_2 is a planar anchored graph. Let $a = |\sigma|$ and σ' be an ordinary (non-cyclic) sequence obtained from σ by "opening it" at any position such that $\sigma'(1) \in V(H_1)$ and $\sigma'(a) \in V(H_2)$.

Let Q be a path on the vertex set $(x^2, s_1, s_2, \ldots, s_a, y^1)$ in this order and let x^1, y^2 be isolated vertices. We construct a graph G_0 from the disjoint union $G \cup Q \cup \{x^1, y^2\}$ by adding the following edges: the edges $\{x^1, \sigma'(1)\}$ and $\{y^2, \sigma'(a)\}$, and the edges $\{s_i, \sigma'(i)\}$ for $i = 1, 2, \ldots, a$. All the edges incident with $V(Q)$ get weight $t = (|E(H_1)|+1)\cdot(|E(H_2)|+1)+1$, while the remaining edges have weight 1. Observe (Figure 4) that $T_0 := \big(G_0, (x^1, x^2), (y^1, y^2)\big)$ is a diagonally separated planar tile by Definition 9.

We claim that $\mathrm{acr}(G) \leq k$ if and only if $\mathrm{tcr}(T_0^{\updownarrow}) \leq k$. In the forward direction, we take an anchored drawing of (G, A, σ) achieving $\mathrm{acr}(G)$ crossings. This drawing immediately gives (see also Figure 2 right) a tile drawing of $T_0^{\updownarrow}$ in which "thick" Q and its incident edges, and the vertices x^1 and y^2, are all drawn along the boundary of the anchored drawing without additional crossings. So, indeed, $\mathrm{tcr}(T_0^{\updownarrow}) \leq \mathrm{acr}(G) \leq k$.

In the backward direction, we observe that there is a valid tile drawing of $T_0^{\updownarrow}$ in which the only crossings are between the edges from $E(H_1) \cup \{\{x^1, \sigma'(1)\}\}$ and the edges from $E(H_2) \cup \{\{y^2, \sigma'(a)\}\}$. Consequently, $\mathrm{tcr}(T_0^{\updownarrow}) \leq (|E(H_1)|+1)\cdot(|E(H_2)|+1) = t-1$. Assume

[1] Note that [6] in general deals with weighted crossing number, in the same way as we do e.g. in Proposition 6. However, since their weights are always polynomial in the graph size, Theorem 11 holds also for unweighted graphs.

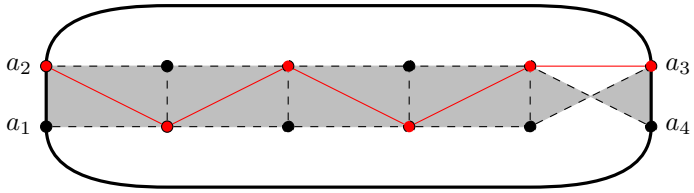


Figure 5 A sketch of the construction of G in the proof of Lemma 13; the cycle C_0 is in bold and the tiles of U are shaded gray.

a tile drawing D_0 (recall, D_0 is contained in a unit square Σ with its walls on the left and right sides of Σ) of $T_0^{\updownarrow}$ with $\mathrm{tcr}(T_0^{\updownarrow})$ crossings. By the previous, no "thick" edge incident with Q is crossed in D_0. Since each of the subgraphs $H_1 + \{x^1, \sigma'(1)\}$ and $H_2 + \{y^2, \sigma'(a)\}$ is connected, and both x^1, y^2 are positioned to the same side of the ends x^2, y^1 of Q on the boundary of Σ, both subgraphs H_1 and H_2 of G are drawn in the same region of Σ separated by the drawing of Q. Contracting the uncrossed ("thick") edges $\{s_i, \sigma'(i)\}$ for $i = 1, 2, \ldots, a$ hence results in an anchored drawing of G with at most $\mathrm{tcr}(T_0^{\updownarrow})$ crossings. The proof is finished. ◀

5 Cross-composing

We now prove the main result, Theorem 2. By Theorem 8 we know that it is enough to construct an OR-cross-composition, that is an algorithm satisfying the requirements of Definition 7.

▶ **Lemma 13.** *Let $\mathcal{L}$ be the language of instances $\langle T^{\updownarrow}, k\rangle$ where T is a diagonally separated planar tile and k an integer polynomially bounded in $|T|$, such that $\mathrm{tcr}(T^{\updownarrow}) \leq k$. Let an equivalence relation $\sim$ be given as $\langle T_1^{\updownarrow}, k_1\rangle \sim \langle T_2^{\updownarrow}, k_2\rangle$ iff $k_1 = k_2$.*

Then $\mathcal{L}$ admits an OR*-cross-composition, with respect to $\sim$, into the graph crossing number problem "$\mathrm{cr}(G) \leq k$" parameterized by k. Moreover, this is true even if we restrict G to be an almost-planar graph.*

Proof. Assume we are given t equivalent instances $\langle T_i^{\updownarrow}, k\rangle$, $i = 1, 2, \ldots, t$, of the tile crossing number problem $\mathcal{L}$; "$\mathrm{tcr}(T_i^{\updownarrow}) \leq k$". Each T_i is a diagonally separated planar tile. We construct a weighted graph G as follows (see also Figure 5 and Lemma 10):

- Let C_0 be a cycle on four vertices a_1, a_2, a_3, a_4 in this cyclic order, and all edges of C_0 having weight $k + 1$.
- Let $\mathcal{T} = (T_1^{\updownarrow}, T_2^{\updownarrow}, \ldots, T_t^{\updownarrow})$. Let $U := \otimes\mathcal{T}$ if m is odd, and $U := (\otimes\mathcal{T})^{\updownarrow}$ otherwise.
- G results from the union of C_0 and U by identifying, in the prescribed order, the left wall of U with (a_1, a_2) and the right wall of U with (a_4, a_3).

We show that $\mathrm{cr}(G) \leq k$ iff $\mathrm{tcr}(U) \leq k$. In the backward direction, any tile drawing of U with ℓ crossings gives a drawing of G with ℓ crossings simply by embedding C_0 "around" the tile U. Conversely, assume a drawing D of G with $\ell \leq k$ crossings, and observe that no edge of C_0 (weighted $k + 1$) is crossed in D. Since $G \setminus C_0$ is connected, it is drawn in one of the two faces of C_0 and this clearly gives a tile drawing of U with ℓ crossings.

Now, by Lemma 10, $\mathrm{tcr}(U) \leq k$ iff there exists $i \in \{1, \ldots, t\}$ such that $\mathrm{tcr}(T_i^{\updownarrow}) \leq k$, as required by Definition 7. The construction of G is easily finished in polynomial time,

and since the edge weights $k+1$ in G are polynomially bounded, there is a polynomial reduction to an unweighted crossing number instance by Proposition 6. It remains to verify that G is almost-planar. Let e_1 be the unique edge of T_1 incident with a_1 in G. Then $\mathrm{tcr}(T_1^{\updownarrow} \setminus e_1) = \mathrm{tcr}(T_1 \setminus e_1) = 0$ and hence $\mathrm{cr}(G \setminus e_1) = 0$. ◀

Theorem 2 follows from Corollary 12 and Lemma 13 via Theorem 8 (note that $\sim$ trivially is a polynomial equivalence).

6 Conclusion

We have proved that the graph crossing number problem parameterized by the number of crossings, which is known to be fixed parameter tractable, is highly unlikely to admit a polynomial kernelization. The complexity of the crossing number problem has been commonly studied under various additional restrictions on the input graph. Our negative result extends even to the instances in which the input graph G is one edge away from planarity (i.e., almost-planar G).

On the other hand, the ordinary crossing number problem remains NP-hard for cubic graphs and for the so-called minor crossing number [12], and for graphs with a prescribed edge rotation system [16]. For a drawing of a graph, the *rotation* of a vertex is the clockwise order of its incident edges (in a local neighbourhood). A *rotation system* is the list of rotations of every vertex. As proved in [16], there is a polynomial equivalence between the problems of computing the crossing number of cubic graphs and that of computing the crossing number under prescribed rotation systems.

The construction we use to show hardness in the paper, produces instances which are "very far" from having small vertex degrees or a fixed rotation system, and there does not seem to be any easy modification for that. Nevertheless, we have an indication that the following strengthening might also be true:

▶ **Conjecture 14.** *Let G be a graph with a given rotation system. Let $k \geq 1$ be an integer. The problem of whether there is a drawing of G respecting the prescribed rotation system and having at most k crossings, parameterized by k, does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly.*

Consequently, the crossing number problem $\mathrm{cr}(G) \leq k$ restricted to cubic graphs G, and the analogous minor crossing number problem, do not admit a polynomial kernel w.r.t. k unless NP $\subseteq$ coNP/poly.

Acknowledgements. We would like to thank the anonymous referees for helpful comments, and the organizers of the Workshop on Kernelization 2015, at the University of Bergen, Norway, where the idea of this paper was born.

References

1 Michael J. Bannister, David Eppstein, and Joseph A. Simons. Fixed parameter tractability of crossing minimization of almost-trees. In *Graph Drawing – GD 2013*, volume 8242 of *Lecture Notes in Computer Science*, pages 340–351. Springer, 2013. doi:10.1007/978-3-319-03841-4_30.

2 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM J. Discrete Math.*, 28(1):277–305, 2014. doi:10.1137/120880240.

3 Drago Bokal, Mojca Bracic, Marek Derňár, and Petr Hliněný. On degree properties of crossing-critical families of graphs. In *Graph Drawing and Network Visualization – GD 2015*, volume 9411 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2015.

4 Drago Bokal, Bogdan Oporowski, R. Bruce Richter, and Gelasio Salazar. Characterizing 2-crossing-critical graphs. Manuscript, 171 pages. http://arxiv.org/abs/1312.3712, 2013.

5 Sergio Cabello. Hardness of approximation for crossing number. *Discrete & Computational Geometry*, 49(2):348–358, 2013. doi:10.1007/s00454-012-9440-6.

6 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. doi:10.1137/120872310.

7 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

8 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

9 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

10 M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM J. Alg. Discr. Meth.*, 4:312–316, 1983.

11 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. ACM, 2001. doi:10.1145/380752.380805.

12 Petr Hliněný. Crossing number is hard for cubic graphs. *J. Comb. Theory, Ser. B*, 96(4):455–471, 2006. doi:10.1016/j.jctb.2005.09.009.

13 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. ACM, 2007. doi:10.1145/1250790.1250848.

14 Martin Kochol. Construction of crossing-critical graphs. *Discrete Mathematics*, 66(3):311–313, 1987. doi:10.1016/0012-365X(87)90108-7.

15 B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins University Press, Baltimore, 2001.

16 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Stefankovic. Crossing numbers of graphs with rotation systems. *Algorithmica*, 60(3):679–702, 2011. doi:10.1007/s00453-009-9343-y.

17 Benny Pinontoan and R. Bruce Richter. Crossing numbers of sequences of graphs II: planar tiles. *Journal of Graph Theory*, 42(4):332–341, 2003. doi:10.1002/jgt.10097.

18 Benny Pinontoan and R. Bruce Richter. Crossing numbers of sequences of graphs I: General tiles. *The Australasian Journal of Combinatorics*, 30:197–206, 2004.

19 R. Bruce Richter and Carsten Thomassen. Minimal graphs with crossing number at least k. *J. Comb. Theory, Ser. B*, 58(2):217–224, 1993. doi:10.1006/jctb.1993.1038.

20 Marcus Schaefer. The graph crossing number and its variants: A survey. *Electronic Journal of Combinatorics*, #DS21, May 15, 2014.

Shortest Path Embeddings of Graphs on Surfaces

Alfredo Hubard[*1], **Vojtěch Kaluža**[†2], **Arnaud de Mesmay**[‡3], **and Martin Tancer**[§4]

1 **INRIA Sophia-Antipolis and Université Paris-Est, Marne-la-Vallée, France**
alfredo.hubard@inria.fr

2 **Department of Applied Mathematics, Charles University, Prague, Czech Republic**
kaluza@kam.mff.cuni.cz

3 **CNRS, Gipsa-Lab, Grenoble, France**
arnaud.de-mesmay@gipsa-lab.fr

4 **Department of Applied Mathematics and Institute of Theoretical Computer Science, Charles University, Prague, Czech Republic**
tancer@kam.mff.cuni.cz

Abstract

The classical theorem of Fáry states that every planar graph can be represented by an embedding in which every edge is represented by a straight line segment. We consider generalizations of Fáry's theorem to surfaces equipped with Riemannian metrics. In this setting, we require that every edge is drawn as a shortest path between its two endpoints and we call an embedding with this property a *shortest path embedding*. The main question addressed in this paper is whether given a closed surface S, there exists a Riemannian metric for which every topologically embeddable graph admits a shortest path embedding. This question is also motivated by various problems regarding crossing numbers on surfaces.

We observe that the round metrics on the sphere and the projective plane have this property. We provide flat metrics on the torus and the Klein bottle which also have this property.

Then we show that for the unit square flat metric on the Klein bottle there exists a graph without shortest path embeddings. We show, moreover, that for large g, there exist graphs G embeddable into the orientable surface of genus g, such that with large probability a random hyperbolic metric does not admit a shortest path embedding of G, where the probability measure is proportional to the Weil-Petersson volume on moduli space.

Finally, we construct a hyperbolic metric on every orientable surface S of genus g, such that every graph embeddable into S can be embedded so that every edge is a concatenation of at most $O(g)$ shortest paths.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Graph embedding, surface, shortest path, crossing number, hyperbolic geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.43

* The research of A. H. was funded by GUDHI, geometric understanding in higher dimensions.
† V. K. was partially supported by the project CE-ITI (GAČR P202/12/G061).
‡ The research of A. dM. leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no. 291734.
§ M. T. was partially supported by the project CE-ITI (GAČR P202/12/G061).

© Alfredo Hubard, Vojtěch Kaluža, Arnaud de Mesmay, and Martin Tancer;
licensed under Creative Commons License CC-BY

1 Introduction

Fáry's theorem and joint crossing numbers. A famous theorem of Fáry [10] (originally showed by K. Wagner [33]) states that any simple planar graph can be embedded with straight line segments representing the edges. In this article, we investigate a generalization of Fáry's theorem to the setting of surface-embedded graphs. We focus on the following question: Given a surface S, is there a metric on S such that every graph embeddable into S admits a *shortest path embedding*, i.e., can be embedded so that the edges are represented by shortest paths? We call such a metric a *universal shortest path metric*. (We do not require that these shortest paths are unique but as we will see later on, in the case of our positive results, i.e., Theorem 1 and 4, the uniqueness of the shortest paths can be obtained as well.)

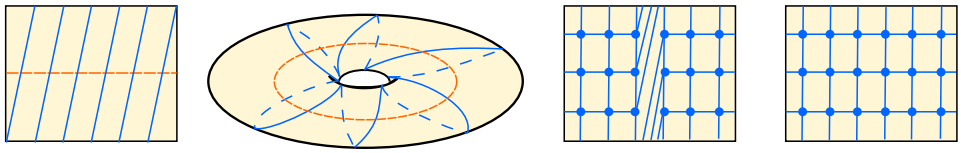
Before being enticed by this question, there were a number of problems involving joint embeddings of curves or graphs on surfaces, but arising from seemingly disparate settings, that motivated us to consider it. The literature on the subject goes back at least 15 years with Negami's work related to diagonal flips in triangulations [25]. He conjectured that there exists a universal constant c such that for any pair of graphs G_1 and G_2 embedded in a surface S, there exists a homeomorphism $h : S \to S$ such that $h(G_1)$ and G_2 intersect transversely at their edges and the number of edge crossings satisfies $cr(h(G_1), G_2) \leq c|E(G_1)| \cdot |E(G_2)|$. Recently, on one hand, Matoušek, Sedgwick, Tancer, and U. Wagner [20, 21], working on decidability of embeddability of 2-complexes into $\mathbb{R}^3$ and on the other hand, Geelen, Huynh, and Richter [12], in a quest for explicit bounds for graph minors, were faced with a similar question and provided bounds for related problems. Dually, this problem is equivalent to the one of finding a graph with a specific pattern within an embedded graph while bounding the multiplicity of the edges used: this is a fundamental question in computational topology of surfaces, where one is interested in finding objects with a fixed topology and minimal combinatorial complexity, e.g., short canonical systems of loops [19], short pants decompositions [5] or short octagonal decompositions [3]; see also [4].

Negami provided an upper bound on the number of crossings which grows linearly with g, and despite subsequent discoveries [1, 27], his conjecture is still open. In a paper that refines Negami's work [27], Richter and Salazar wrote *"this [conjecture] seems eminently reasonable: why should two edges be forced to cross more than once?"*. The connection with our work is that if two graphs are embedded transversally by shortest path embeddings, then indeed no two edges cross more than once, since otherwise one of them could be shortcut.

Beyond these crossing numbers, the existence of shortest path universal metrics might be of interest to Riemannian geometers working on curvature free estimates and extremal metrics.

Related work. Generalizing Fáry's theorem is one of the drives of the field of graph drawing [31], where a lot of effort has been devoted towards embedding planar graphs with additional constraints such that the edges are straight lines, or polylines with few bends. Yet only few extensions to graphs embedded in surfaces are known. Two classical avatars of Fáry's theorem in the plane are of relevance to our work: Tutte's barycentric embedding theorem [32] and the Koebe-Andreev-Thurston circle packing theorem (see, for example, the book of Stephenson [28]). Both have been generalized to surfaces, providing positive answers to the following questions:

1. Given a surface S, a metric m, and a graph G embeddable into S, can we embed the graph G so that every edge is represented by a geodesic with respect to m?
2. Given a graph G embeddable into S, does there exist a metric m on S so that G embeds into S with shortest paths?



a. b. c. d.

Figure 1 a. and b. Two geodesics crossing many times. c. A grid embedded in a torus with geodesics. d. A reembedding of this grid with shortest paths.

Regarding the first question, Y. Colin de Verdière [6] generalized Tutte's barycentric embedding approach using a variational principle. The idea is to start with a topological embedding of the graph, replace the edges by springs, and let the system reach an equilibrium. Y. Colin de Verdière proved that for any metric of non-positive curvature, the edges become geodesics with disjoint interiors when the system reaches stability; moreover, this embedding is essentially unique within its homotopy class. However, geodesics need not be shortest paths, and two geodesics can intersect an arbitrarily large number of times, see Figure 1. Yet, these examples do not provide a negative answer to the second question, or to our main question, since we could change the embedding by a homeomorphism of the torus (thus even preserving the combinatorial map) to obtain a shortest path embedding.

The second question also has a positive answer, which can be proved via a generalization of the circle packing theorem to closed surfaces [28]. Namely, for every triangulation T of a surface, there exists a metric of constant curvature so that T can be represented as the contact graph of a family of circles. The representation of the triangulation that places a vertex at the center of its corresponding circle is an embedding with shortest paths. Such a representation can be computed efficiently and can be used as a tool for representing graphs on surfaces [23]. However, the metric is "chosen" by the triangulation, which makes this approach ill-suited for our purpose.

Our results. Our objective here is a mix of these last two results. On the one hand, we require shortest paths and not geodesics, on the other hand, we want a single metric for each surface and not one which depends on the triangulation. We will also consider the relaxation of our problem where we are allowed to use concatenations of shortest paths: we say that a metric is a *k-universal shortest path metric* if every topologically embeddable graph can be represented by an embedding in which edges are drawn as concatenations of k shortest paths. This is akin to various problems in graph drawing where graphs are embedded with polylines with a bounded number of bends instead of straight lines [8, 30].

Our results and techniques are organized by curvature. We first observe that for the sphere and the projective plane, since there is a unique Riemannian metric of curvature 1, the circle packing approach applies to all graphs. Then, with the aid of *irreducible triangulations*, we provide flat metrics (i.e., of zero curvature) on the torus and the Klein bottle for which every graph admits a shortest path embedding.

▶ **Theorem 1.** *The sphere S^2, the projective plane $\mathbb{R}P^2$, the torus T^2, and the Klein bottle K can be endowed with a universal shortest path metric.*

This result could lead to the idea that shortest path embeddings can be achieved for any metric, i.e., that every metric is a universal shortest path metric. We prove that this is not the case already for the unit square flat metric on the Klein bottle (arguably the first example to consider).

▶ **Theorem 2.** *Let K denote the Klein bottle endowed with the unit square flat metric on the polygonal scheme $aba^{-1}b$. Then there exists a graph embeddable into K which cannot be embedded into K so that the edges are shortest paths.*

In higher genus, the number of irreducible triangulations is too large to check all cases by hand. Hyperbolic surfaces of large genus are hard to comprehend, but the probabilistic method allows us to show that if there exists universal shortest path metrics at all, their fraction tends to 0 as the genus tends to infinity.

▶ **Theorem 3.** *For any $\varepsilon > 0$, with probability tending to 1 as g goes to infinity, a random hyperbolic metric is not a $O(g^{1/3-\varepsilon})$-universal shortest path metric. In particular, with probability tending to 1 as g goes to infinity, a random hyperbolic metric is not a universal shortest path metric.*

We refer to Section 5 for an introduction to the probability measure used on the space of hyperbolic metrics, called the Weil-Petersson volume form. Our proof is an application of deep results on this volume form by Mirzakhani [22] and Guth, Parlier, and Young [14].

For genus $g > 1$ we do not know if there exist shortest path universal metrics. But relaxing the question to concatenations of shortest paths and combining ideas from hyperbolic geometry and computational topology, we provide for every orientable surface of genus g an $O(g)$-universal shortest paths metric. The proof relies on the octagonal decompositions of É. Colin de Verdière and Erickson [3] and a variant of the aforementioned theorem of Y. Colin de Verdière [6].

▶ **Theorem 4.** *For every $g > 1$, there exists an $O(g)$-universal shortest path hyperbolic metric m on the orientable surface S of genus g.*

In this article we choose to focus on Riemannian metrics of constant curvature, but we remark that both of our last results also hold in the setting of piecewise-Euclidean metrics as well. For the upper bound, it suffices to replace hyperbolic hexagons with Euclidean ones, and the rest of the proof works similarly. For the lower bound it follows from the strong parallels between the Weil-Petersson volume form on moduli space and the counting measure on the space of $N = 4g$ Euclidean triangles randomly glued together. In particular the results that we use have analogs in this latter space: see Brooks and Makover [2] and the second half of the article of Guth, Parlier, and Young [14].

We have stated our results for graphs in this introduction. We note that one could consider the problem of shortest path embeddings for a graph with a fixed embedding up to a homeomorphism of the surface (i.e., for a combinatorial map), which is more in the spirit of Negami's conjecture. Our positive results can be stated in this stronger version; i.e., in our proofs the map is preserved. Our negative results would be weaker if the map had to be preserved, and in fact the proofs deal firstly with the statements for maps and then we derive the analog for graphs with some extra work.

The main open question is the existence of universal shortest path metrics, or $O(1)$-universal shortest path metrics. Natural candidates for these are given by certain celebrated extremal metrics, for instance the one minimizing $\int \log \det(\Delta) ds$, (where Δ denotes the Laplacian) that appeared in the work of Osgood, Phillips, and Sarnak [26], or the extremal ones in Gromov's systolic inequality [13].

After introducing the main definitions in Section 2, we will prove Theorems 1, 2, 3, and 4 in Sections 3, 4, 5, and 6, respectively.

2 Preliminaries

In this article we only deal with compact surfaces without boundaries. By the classification theorem, these are characterized by their *orientability* and their *genus*, generally denoted by g. Orientable surfaces of genus 0 and 1 are respectively the *sphere* S^2 and the *torus* T^2, while non-orientable surfaces of genus 1 and 2 are the *projective plane* $\mathbb{R}P^2$ and the *Klein bottle* K. The orientable surface of genus g is denoted by S_g. The *Euler genus* is equal to the genus for non-orientable surfaces and equals twice the genus for orientable surfaces.

By a *path* on a surface S we mean a continuous map $p : [0, 1] \to S$, and a *closed curve* denotes a continuous map $\gamma : S^1 \to S$. These are *simple* if they are injective. We will be using occasionally the notions of *homotopy*, *homology*, and *universal cover*, we refer to Hatcher [15] for an introduction to these concepts. All the graphs that we consider in this paper are **simple** graphs unless specified otherwise, i.e., loops and multiple edges are disallowed. An *embedding* of a graph G into a surface S is, informally, a crossing-free drawing of G on S. We refer to Mohar and Thomassen [24] for a thorough reference on graphs on surfaces, and only recall the main definitions. A graph embedding is *cellular* if its faces are homeomorphic to open disks. Euler's formula states that $v - e + f = 2 - g$ for any graph with v vertices, e edges, and f faces cellularly embedded in a surface S of Euler genus g. When the graph is not cellularly embedded, this becomes an inequality: $v - e + f \geq 2 - g$. A *triangulation* of a surface is a cellular graph embedding such that all the faces are adjacent to three edges. By a slight abuse of language, we will sometimes refer to an embedding of a triangulation, by which we mean an embedding of its underlying graph which is homeomorphic to the given triangulation. A *pants decomposition* of a surface S is a family of disjoint curves Γ such that cutting S along all of the curves of Γ gives a disjoint union of pairs of pants, i.e., spheres with three boundaries. Every surface except the sphere, the projective plane, the torus, and the Klein bottle admits a pants decomposition with $3g-3$ closed curves and $2g-2$ pairs of pants. Note that all the pants decompositions are not topologically the same, i.e., are not related by a self-homeomorphism of the surface. A class of pants decompositions equivalent under such homeomorphisms will be called the (topological) *type* of the pants decomposition. We say that an embedding $f : G \to S$ *contains a pants decomposition* if there exists a subgraph $H \subseteq G$ such that $f : H \to S$ is a pants decomposition of S.

In this article, we will also be dealing with notions coming from Riemannian geometry, we refer to the book of do Carmo for more background [7]. By a *metric* we always mean a Riemannian metric, which associates to every point of a surface the *curvature* at this point. The Gauss-Bonnet theorem ties geometry and topology; it implies that the sign of a metric of constant curvature that a topological surface accepts is determined solely by its Euler genus.

A Riemannian metric induces a length functional on paths and closed curves. A path or a closed curve is a *geodesic* if the functional is locally minimal. Shortest paths between two points are global minima of the length functional. Unlike in the plane, geodesics are not, in general, shortest paths; in addition, neither geodesics nor shortest paths are unique in general. If we have a shortest path embedding of a graph where every edge is drawn as the unique shortest path between its endpoints, we speak of *shortest paths embedding with uniqueness*.

3 Shortest path embeddings through minimal triangulations.

▶ **Theorem 1.** *The sphere S^2, the projective plane $\mathbb{R}P^2$, the torus T^2, and the Klein bottle K can be endowed with a universal shortest path metric.*

In the theorem above, for S^2 and $\mathbb{R}P^2$ we use the round metric of positive constant curvature scaled to 1. In the case of torus we use the flat metric obtained by the identification of the opposite edges of the square. In the case of the Klein bottle we can show that an analogous result fails with the flat square metric on the polygonal scheme $aba^{-1}b$, as we will see in Section 4. But we can get the result for the metric obtained by the identification of the edges of a rectangle of dimensions $1 \times b$ where $b = \sqrt{4/3} + \varepsilon$ for some small $\varepsilon > 0$. (The edges of length 1 are identified coherently, whereas the edges of length b are identified in opposite directions.)

In all cases we can get shortest path embeddings with uniqueness. Actually, for the torus and the Klein bottle, uniqueness will be a convenient assumption for inductive proofs.

The sphere and the projective plane. The circle packing theorem states that any triangulation of the sphere can be represented as the contact graph of circles on the sphere [28, Theorem 4.3], endowed with the usual metric. Since any graph can be extended into a triangulation (adding new vertices if needed), and paths between centers of circles are shortest paths, this proves Theorem 1 for S^2. For $\mathbb{R}P^2$ endowed with the spherical metric, a similar circle packing theorem holds, yet we could not find a satisfying reference for it, so we provide a proof of it in the full version of the paper [16], based on building the circle-packing in the universal cover of $\mathbb{R}P^2$.

Minimal triangulations. Let S be a surface and T be a triangulation of it. The triangulation T is called *reducible*, if it contains an edge e such that the contraction of e yields again a triangulation, which we denote by T/e. We refer to e as a *contractible* edge (we do not mean contractibility in a topological sense). On the other hand, a triangulation is *minimal* (or *irreducible*), if no edge can be contracted this way. For every surface there is a finite list of minimal triangulations. In particular, for the torus T^2 this list consists of 21 triangulations found by Lawrencenko [17] and for the Klein bottle K there are 29 minimal triangulations found by Sulanke [29].

The strategy of the proof of Theorem 1 for T^2 and K is to show that it is sufficient to check Theorem 1 for minimal triangulations with appropriate fixed metric; see Lemma 5. Then, since every embedded graph can be extended to a triangulation (possibly with adding new vertices), we finish the proof by providing the list of shortest path embeddings of the minimal triangulations.

▶ **Lemma 5.** *Let S be a surface equipped with a flat metric. Let T be a reducible triangulation with contractible edge e. Let us assume that T/e admits a shortest path embedding with uniqueness into S. Then T admits a shortest path embedding with uniqueness into S as well.*

The restriction on flat metrics in the lemma above does not seem essential, but this is all we need and this way the proof is quite simple.

Proof. Let v be the vertex of T/e obtained by the contraction of e. The idea is to consider the shortest path drawing of T/e. Then we perform the appropriate vertex splitting of v (the inverse operation to the contraction) in a close neighborhood of v so that we get a shortest path embedding of T.

In order to see that this is indeed possible, let us consider an edge $f = uv$ of T/e. Since the shortest paths are unique, a simple compactness argument shows that there is an ε-neighborhood N_f of f in S which is isometric to an ε-neighborhood of a segment of the same length in $\mathbb{R}^2$ and such that for every v' in the ε-neighborhood of v, the straight line segment connecting v and v' inside N_f is the unique shortest path between v and v' in S.

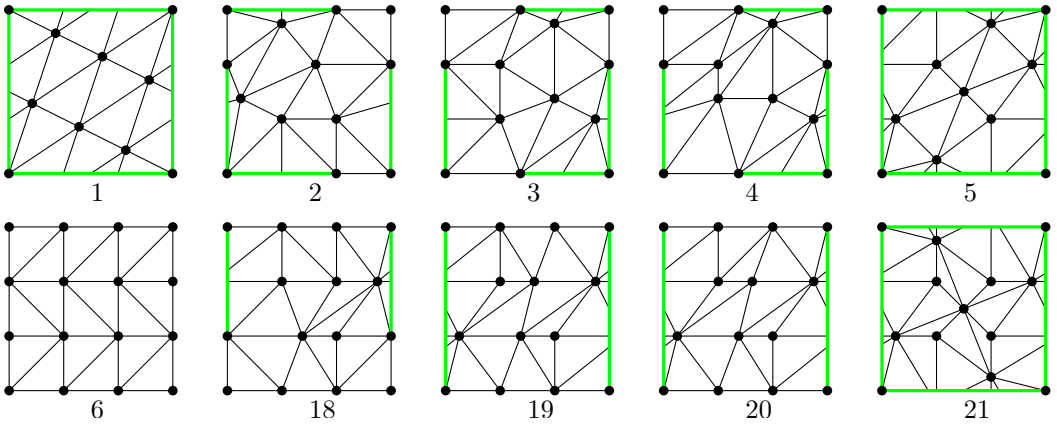
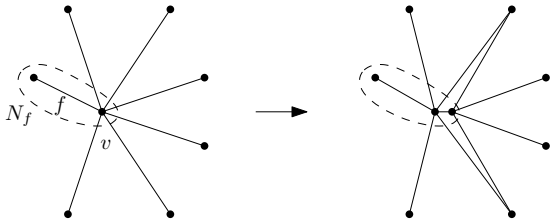


Figure 2 Minimal triangulations of the torus.

Therefore, considering such a neighborhood for every edge, it is sufficient to perform the vertex splitting of v in sufficiently small neighborhood of v so that we do not introduce new intersections as on the picture.



◀

The minimal triangulations of T^2 and K. In Figure 2 we provide a list of shortest path embeddings with uniqueness of minimal triangulations of the torus with a flat metric obtained by identifying the opposite edges of the unit square. They are in the same order as in the book of Mohar and Thomassen [24, Figure 5.3]. The black (thin) edges are the edges of the triangulation whereas the green (thick) edges are the identified boundaries of the unit square which are not parts of the edges of the triangulations. We just skip drawings of the triangulations 7 to 17, because they are all analogous to the triangulation 6, they only have different patterns of diagonals. It is clear that every edge is a geodesic. In order to check that each of them is drawn as a shortest path, it is sufficient to verify that each edge projects vertically and horizontally to a segment of length less than $\frac{1}{2}$.

For the Klein bottle K, we also provide a metric such that all the minimal triangulations admit shortest path embeddings with uniqueness. We obtain this metric as the identification of the edges of the rectangle $R = [0, a] \times [0, b]$, where $a = 1$ and $b = \sqrt{4/3} + \varepsilon$ for sufficiently small ε. The edges of length 1 are identified in coherent directions. The edges of length b are identified in the opposite directions. Minimal triangulations of the Klein bottle were obtained by Lawrencenko and Negami [18], and Sulanke [29], and we show how to embed them with shortest paths in Figure 3. The details regarding these embeddings are explained in the full version [16], but for a cursory glance the Figures 3 should be self-explanatory with a couple of remarks. In some cases an additional shift is necessary by a small value $\frac{1}{1000}$ (but this value is large compared to ε): this is indicated by arrows next to the vertices. (The pair of arrows in Kh25 indicates a shift by $\frac{2}{1000}$.) The triangulations split into two

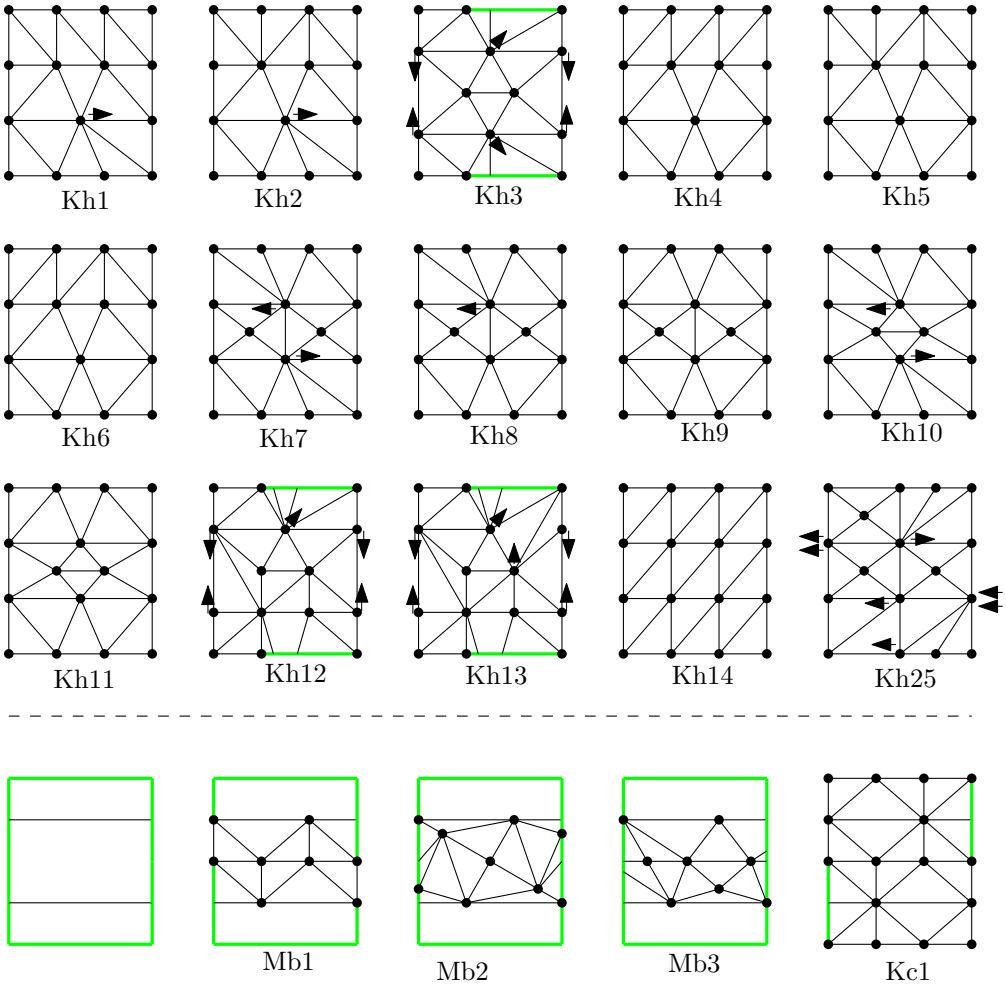


Figure 3 Minimal triangulations of the Klein bottle.

types—so called Kh cases and Kc cases. For the Kh cases we omit few pictures analogous to Kh14. For the Kc cases, the triangulations of the Klein bottle are obtained by gluing together two triangulations of the Möbius band pictured in the bottom line; Kc1, obtained from two copies of Mb1, is provided as an example.

4 Square flat metric on the Klein bottle

The task of this section is to prove the following theorem.

▶ **Theorem 2.** *Let K denote the Klein bottle endowed with the unit square flat metric on the polygonal scheme $aba^{-1}b$. Then there exists a graph embeddable into K which cannot be embedded into K so that the edges are shortest paths.*

We consider the minimal triangulation Kc1 (see Figure 3) and we denote by G the underlying graph for this triangulation. We will prove that G does not admit a shortest path embedding into K with the square metric. First, we observe that the triangulation Kc1 is the only embedding of G into K. The proof is given in the full version [16].

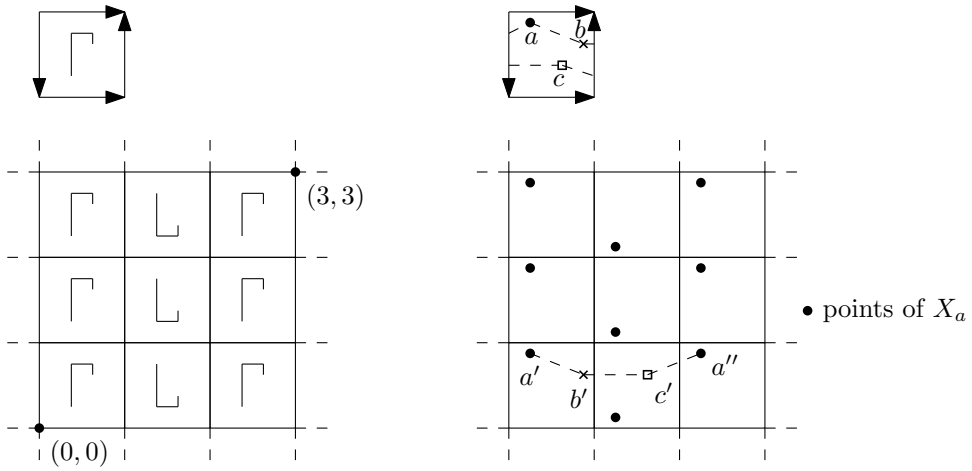


Figure 4 The Klein bottle with a letter 'Γ', and its universal cover (left). A lift of the cycle *abc* (right).

▶ **Proposition 6.** *G has a unique embedding into the Klein bottle.*

For contradiction, let us assume that G admits a shortest paths embedding into K. We know that Kc1 is obtained by gluing two triangulations of a Möbius band along a cycle of length 3 (the triangle corresponding to this cycle is not part of the triangulation). Let abc be this cycle. With a slight abuse of notation we identify this cycle with its image in the (hypothetical) shortest path embedding into K. Our strategy is to show that already abc cannot be embedded into K with shortest path edges, which will give the required contradiction. By Proposition 6, we know that abc splits K to two Möbius bands.

Let $X = \mathbb{R}^2$ be the universal cover of K (with standard Euclidean metric). Let $\pi\colon X \to K$ be the isometric projection corresponding to the cover. We will represent the Klein bottle with the flat-square metric as the unit square $[0,1]^2$ with suitable identification of the edges ($aba^{-1}b$, as in the previous section). We will use the convention that $\pi((0,1)^2) = (0,1)^2$; that is, the projection is the identity on the interior of this square. See Figure 4.

Given a point $p \in K$ we set $X_p := \pi^{-1}(p)$. Finally, let $\mathcal{V}_p$ be the Voronoi diagram in X corresponding to the set X_p.

▶ **Lemma 7.** *Let p and q be two points in K and γ be an arc (edge) connecting them, considered as a subset of K. Then γ is the unique shortest path between p and q if and only if there are $p' \in X_p$, $q' \in X_q$ such that $\gamma = \pi(\overline{p'q'})$ where $\overline{p'q'}$ denotes the straight edge connecting p' and q' in X and q' belongs to the open Voronoi cell for p' in $\mathcal{V}_p$.*

The proof directly follows the correspondence between K and X. It is given in the full version [16].

Now let us lift the cycle abc to a path $a'b'c'a''$ in X; see Figure 4. Given a curve in X, we call the length of its projection to the x-axis, the "horizontal length" of the curve; similarly we speak about the horizontal distance and the vertical distance of two points in X.

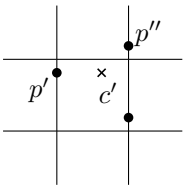
▶ **Lemma 8.** *The horizontal distance between a' and a'' is at least 2.*

Sketch of a proof. If we consider the point a' fixed, then the position of a'' in X_a determines the homotopy class of the cycle abc in the fundamental group $\pi_1(K)$.

The horizontal distance between a' and a'' must be a non-negative integer. However, by checking the homology class of abc in the $\mathbb{Z}$ and $\mathbb{Z}_2$-homology, it is possible to rule out the cases when this distance is 0. By the fact that abc is a double-sided cycle, it is possible to rule out the distance 1. The full proof is given in the full version of this paper [16]. ◀

▶ **Lemma 9.** *Let γ be a unique shortest path in K connecting points p and q. Let γ' be a lift of γ with endpoints p' and q'. Then the horizontal distance in X between p' and q' is less than $\frac{5}{8}$.*

Proof. Let C be the open Voronoi cell for p' in $\mathcal{V}_p$. By Lemma 7, q' belongs to C. Therefore, it is sufficient to check that every point c' of C has horizontal distance less than $\frac{5}{8}$ from p'. Without loss of generality, we may assume that the x-coordinate of p' equals 0 since shifting p' in horizontal direction only shifts X_p and $\mathcal{V}_p$ (note that this is not true for the vertical direction). For contradiction, there is a c' in C at distance at least $\frac{5}{8}$ and without loss of generality the x-coordinate of c' is positive. Let p'' be the point of X_p with x-coordinate equal 1 which is vertically closest to c' (pick any suitable point in case of draw); see the picture on the left. The vertical distance between c' and p'' is at most $\frac{1}{2}$. A simple calculation, using the Pythagoras theorem, gives that p'' is at most as far from c' as p'. A contradiction.



◀

Finally, we summarize how the previous lemmas yield a contradiction. By Lemma 8, the horizontal distance between a' and a'' is at least 2. On the other hand, Lemma 9 gives that the horizontal length of each of the edges $a'b'$, $b'c'$, and $c'a''$ is at most $\frac{5}{8}$, altogether at most $\frac{15}{8}$. This gives the required contradiction, which finishes the proof of Theorem 2.

5 Asymptotically almost all hyperbolic metrics are not universal

Before stating the main theorem of this section, we will give some very quick background on the geometry of surfaces, we refer to Farb and Margalit [9] for a proper introduction. The Teichmüller space $\mathcal{T}_g$ of a surface S of genus g denotes the set of hyperbolic metrics on S, such that two metrics are equivalent if they are related by an isometry isotopic to the identity. In some contexts, like ours, one might also want to identify metrics related by an isometry (not necessarily isotopic to the identity). The corresponding space is called the *moduli space* $\mathcal{M}_g$ of the surface, and is obtained by quotienting $\mathcal{T}_g$ by the *mapping class group* of S, i.e., its group of homeomorphisms. This moduli space can be endowed with multiple structures, here we will be interested in a particular one, called the Weil-Petersson metric. This metric provides $\mathcal{M}_g$ with a Riemannian structure of finite volume, and therefore by renormalizing, we obtain a probability space, allowing to choose a random metric. We can now state the main theorem of this section.

▶ **Theorem 3.** *For any $\varepsilon > 0$, with probability tending to 1 as g goes to infinity, a random hyperbolic metric is not a $O(g^{1/3-\varepsilon})$-universal shortest path metric. In particular, with probability tending to 1 as g goes to infinity, a random hyperbolic metric is not a universal shortest path metric.*

The proof is a consequence of two important results on random hyperbolic metrics. The first is a small variant of a theorem of Guth, Parlier, and Young [14, Theorem 1]. Before stating it, we need some definitions.

Given a hyperbolic metric m on a surface S, we say that m has total pants length at least ℓ if in any pants decomposition Γ of S, the lengths of the closed curves of Γ sum up to at least ℓ. We say that m has total pants length of type ξ at least ℓ if in any pants decomposition Γ of S of type ξ, the lengths of the closed curves of Γ sum up to at least ℓ.

▶ **Theorem 10.** *For any $\varepsilon > 0$, a random metric on $\mathcal{M}_g$ has total pants length at least $g^{7/6-\varepsilon}$ with probability tending to 1 as $g \to \infty$. For any $\varepsilon > 0$ and any family of types of pants decomposition ξ_g, a random metric on $\mathcal{M}_g$ has total pants length of fixed type ξ_g at least $g^{4/3-\varepsilon}$ with probability tending to 1 as $g \to \infty$.*

The proof is almost identical to the one in Guth, Parlier, and Young [14]; it is summarized in the full version of this paper [16].

The following is an immediate corollary of this theorem.

▶ **Corollary 11.** *Let T_g be a family of triangulations of S_g, such that every member of T_g contains a pants decomposition of fixed type ξ_g. For any $\varepsilon > 0$, with probability tending to 1 as $g \to \infty$, a shortest embedding of T_g into a random hyperbolic surface of genus g has length at least $\Omega(g^{4/3-\varepsilon})$.*

The next theorem was proved by Mirzakhani [22, Theorem 4.10].

▶ **Theorem 12.** *With probability tending to 1, the diameter of a random hyperbolic surface of genus g is $O(\log g)$.*

Theorem 3 is proved by providing an explicit family of graphs G_g which will embed badly. It is defined in the following way for $g \geq 2$. Let ξ_g be a type of pants decompositions for every value of g.

- We start with a pants decomposition of type ξ_g of a surface S_g
- We place four vertices on every boundary curve.
- We triangulate each pair of pants with a bounded size triangulation so that each cycle of length 3 bounds a triangle in the triangulation, and any path connecting two boundary components of the pair of pants has length at least 4 (in particular G_g is a simple graph and each cycle of length 3 in the graph G_g bounds a triangle in the triangulation).

The following proposition controls the issues related to the flexibility of embeddings of graphs into surfaces, it is proved in the full version [16].

▶ **Proposition 13.** *There is a unique embedding of G_g into S_g, up to a homeomorphism; in particular every embedding contains a pants decomposition of type ξ_g.*

With these three results at hand we are ready to provide a proof of the theorem.

Proof of Theorem 3. We use the family of graphs G_g previously defined. Since there are $O(g)$ curves in a pants decomposition, it contains $O(g)$ edges, and every embedding of G_g into S_g contains a pants decomposition of type ξ_g by Proposition 13.

Now, by Corollary 11, for every $\varepsilon > 0$, and for g large enough, the probability that the shortest possible embedding of G_g into a random metric has length at least $O(g^{4/3-\varepsilon})$ is at least $1 - \varepsilon/2$. In particular, since there are $O(g)$ edges in G_g, some edge e_g in this embedding must have length at least $\Omega(g^{1/3-\varepsilon})$. By Theorem 12, we can choose g large enough so that with probability at least $1 - \frac{\varepsilon}{2}$, the random hyperbolic metric has diameter $O(\log g)$. Hence,

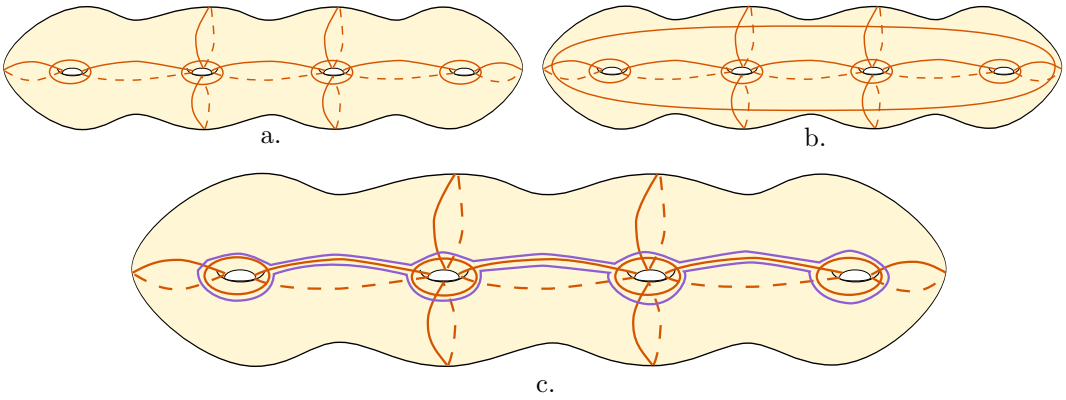


Figure 5 a. An octagonal decomposition b. A hexagonal decomposition c. How to add one closed curve to upgrade an octagonal decomposition to a hexagonal decomposition.

by the union bound, with probability $1 - \varepsilon$ both properties hold. Therefore, for every $\varepsilon > 0$, there exists some value g_0 such that for any $g \geq g_0$, in any embedding of G_g, there exists an edge $e_g = (x, y)$ such that $\ell_m(e_g) = \Omega(g^{1/3-\varepsilon})$, but $d_m(x, y) \leq diam(m) \leq O(\log g)$. This implies that e is not drawn by a shortest path. Similarly, subdividing each edge $O(g^{1/3-\varepsilon})$ times will run into the same issue. This concludes the proof. ◀

6 Higher genus: positive results

▶ **Theorem 4.** *For every $g > 1$, there exists an $O(g)$-universal shortest path hyperbolic metric m on the orientable surface S of genus g.*

Our approach to prove Theorem 4 is to cut the surface S_g with a *hexagonal decomposition* Δ, so that every edge of G is cut $O(g)$ times by this decomposition Δ. The construction to do this is a slight modification of the *octagonal decompositions* provided by É. Colin de Verdière and Erickson [3, Theorem 3.1]. Each of the hexagons is then endowed with a specific hyperbolic metric m_H, and pasting these together yields the hyperbolic metric m on S_g. The hyperbolic metric m_H is chosen so that the hexagons are *convex*, i.e., the shortest paths between points of a hexagon stay within this hexagon. Therefore, there only remains to embed the graph G cut along Δ, separately in every hexagon with shortest paths. To do this, we use a variant of a theorem of Y. Colin de Verdière [6] which generalizes Tutte's barycentric method to metrics of nonpositive curvature.

Hexagonal decompositions. A *hexagonal decomposition*, respectively an *octagonal decomposition* of S_g is an arrangement of closed curves on S_g that is homeomorphic to the one pictured in Figure 5.a., respectively Figure 5.b. In particular, every vertex has degree four and every face has six sides, respectively eight sides.

Octagonal decompositions were introduced by É. Colin de Verdière and Erickson [3] where they showed how to compute one that does not cross the edges of an embedded graph too many times. We restate their theorem in our language.

▶ **Theorem 14** ([3, Theorem 3.1])**.** *Let G be a graph embedded in a surface S_g for $g \geq 2$. There exists an octagonal decomposition Γ of S_g such that each edge of G crosses each closed curve of Γ a constant number of times.*

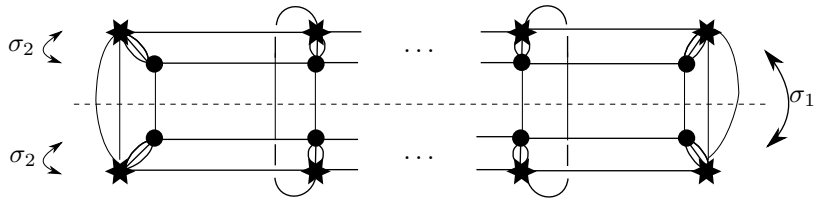


Figure 6 The intersection graph I and the two involutions: σ_1 is the symmetry about the dashed horizontal line, and σ_2 swaps every disk and its adjacent star.

We observe that this octagonal decomposition can be upgraded to a hexagonal decomposition that still does not cross G too much:

▶ **Corollary 15.** *Let G be a graph embedded in a surface S_g. There exists a hexagonal decomposition Δ of S_g such that each edge of G crosses each closed curve of Δ a constant number of times, except for maybe one closed curve which is allowed to cross each edge of G at most $O(g)$ times. In particular, the number of crossing between every edge of G and Δ is $O(g)$.*

Proof. The decomposition Δ is simply obtained by taking the decomposition Γ and adding a single curve that follows closely a concatenation of $O(g)$ subpaths of curves of Γ, see Figure 5c. The resulting arrangement of curves has the topology of a hexagonal decomposition, and the bounds on the number of crossings results directly from the construction. ◀

The hyperbolic metric. We first endow each hexagon of the hexagonal decomposition with the hyperbolic metric m_H of an equilateral right-angled hyperbolic hexagon. Since the hexagons have right angles and the vertices of a hexagonal decomposition have degree 4, this metric can be safely pasted between hexagons to endow S_g with a hyperbolic metric m. The main property of this metric that we will use is the following one:

▶ **Proposition 16.** *Every hexagon H, viewed as a subset of S_g endowed with m, is convex, i.e., every path between $x, y \in H$ that is a shortest path in H is also a shortest path in S_g.*

Proof. The proof relies on an exchange argument based on the symmetries of the hexagonal decomposition.

The intersection graph I of the hexagonal decomposition is defined by taking one vertex for each hexagon and edges between adjacent hexagons (we allow multiple edges). We are interested in two graph automorphisms which are also involutions. These are pictured in Figure 6:

- The symmetry σ_1 about the horizontal axis, corresponding to the so-called hyperelliptic involution of the surface S.
- The automorphism σ_2 swapping every hexagon with its neighbor in the octagonal decomposition.

Since all the hexagons are isometric, these involutions correspond naturally to isometric involutions of S.

Now, let γ be a shortest path between two vertices x and y in a hexagon H_1, let us assume without loss of generality that H_1 is in the upper part of I. This path γ naturally induces a walk in I obtained by taking each hexagon of which interior is met by γ. This walk does not backtrack at some hexagon H: otherwise one could shortcut γ by staying on the boundary of H.

From γ, one can build a path γ' between x and y which stays in the upper half of the graph: for every maximal subpath of the graph in the lower half, one applies the isometry σ_1, effectively mirroring these paths in the upper half. Similarly, by applying σ_2, one obtains a path γ'', which only uses half of the hexagons. The walk in I corresponding to γ'' lies now in a path. Since it does not backtrack, it is necessarily trivial and never leaves hexagon H_1. We have thus found a shortest path in H_1 connecting x and y. ◀

Finishing the proof. We prove in this paragraph how to reembed a graph embedded in a hexagon so that its edges are shortest paths. This allows us to finish the proof.

▶ **Theorem 17.** *Let G be a graph embedded as a triangulation in a hyperbolic hexagon H endowed with the metric m_H. If there are no dividing edges in G, i.e., edges between two non-adjacent vertices on the boundary of H, then G can be embedded with geodesics, with the vertices on the boundary of H in the same positions as in the initial embedding.*

Given this theorem, we can now conclude the proof of Theorem 4: the intersections of our input graph with the hexagonal decompositions subdivide it, and in each hexagon one can subdivide the dividing edges if needed, then upgrade the subgraph to a triangulation, and finally embed it with this theorem. The details are in the full version [16]. We note that by subdividing each edge once more, the shortest paths we obtain are unique.

The proof of Theorem 17 is obtained in a spirit similar to the proof of the one of the celebrated *spring theorem* of Tutte [32]. However, there are two main differences which prevent us from directly appealing to the literature: on the one hand the metric is not Euclidean but hyperbolic, and on the other hand the boundary of the input polygon is not *strictly convex*, since there may be multiple vertices of G on a geodesic boundary of H. The hypothesis on dividing edges is tailored to circumvent the second issue, and in a Euclidean setting it was proved by Floater [11] that the correspond embedding theorem holds. Regarding the first issue, Y. Colin de Verdière stated a Tutte embedding theorem [6, Theorem 3] for the hyperbolic setting with strictly convex boundary, yet he actually did not provide a proof for it. In Appendix A of the full version [16] we show how to prove Theorem 17 in the generality that we need following the ideas laid out (in French) by Y. Colin de Verdière in the rest of his article [6].

Acknowledgements. We are grateful to Éric Colin de Verdière for his involvement in the early stages of this research, to Sergio Cabello, Francis Lazarus, Bojan Mohar, and Eric Sedgwick for helpful discussions and to Xavier Goaoc for organizing the workshop that led to this work.

References

1 Dan Archdeacon and C. Paul Bonnington. Two maps on one surface. *Journal of Graph Theory*, 36(4):198–216, 2001.

2 Robert Brooks and Eran Makover. Random construction of Riemann surfaces. *J. Differential Geom.*, 68(1):121–157, 2004.

3 Éric Colin de Verdière and Jeff Erickson. Tightening nonsimple paths and cycles on surfaces. *SIAM Journal on Computing*, 39(8):3784–3813, 2010.

4 Éric Colin de Verdière, Alfredo Hubard, and Arnaud de Mesmay. Discrete systolic inequalities and decompositions of triangulated surfaces. *Discrete & Computational Geometry*, 53(3):587–620, 2015.

5 Éric Colin de Verdière and Francis Lazarus. Optimal pants decompositions and shortest homotopic cycles on an orientable surface. *Journal of the ACM*, 54(4):Article 18, 2007.

6 Yves Colin de Verdière. Comment rendre géodésique une triangulation d'une surface ? *L'Enseignement Mathématique*, 37:201–212, 1991.

7 Manfredo P. do Carmo. *Riemannian geometry*. Birkhäuser, 1992.

8 Cesim Erten and Stephen G. Kobourov. Simultaneous embedding of planar graphs with few bends. *Journal of Graph Algorithms and Applications*, 9(3):347–364, 2005.

9 Benson Farb and Dan Margalit. *A primer on mapping class groups*. Princeton University Press, 2011.

10 István Fáry. On straight line representations of planar graphs. *Acta scientiarum mathematicarum (Szeged)*, 11:229–233, 1948.

11 Michael Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696, 2003.

12 Jim Geelen, Tony Huynh, and R. Bruce Richter. Explicit bounds for graph minors. arXiv:1305.1451, 2013.

13 Mikhael Gromov. Filling Riemannian manifolds. *Journal of Differential Geometry*, 1983.

14 Larry Guth, Hugo Parlier, and Robert Young. Pants decompositions of random surfaces. *Geometric and Functional Analysis*, 21:1069–1090, 2011.

15 Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002. Available at `http://www.math.cornell.edu/~hatcher/`.

16 Alfredo Hubard, Vojtěch Kaluža, Arnaud de Mesmay, and Martin Tancer. Shortest path embeddings of graphs on surfaces. *arXiv preprint: http://arxiv.org/abs/1602.06778*, 2016.

17 Serge Lawrencenko. The irreducible triangulations of the torus. *Ukrainskiĭ Geometricheskiĭ Sbornik*, 30:52–62, 1987.

18 Serge Lawrencenko and Seiya Negami. Irreducible triangulations of the Klein bottle. *Journal of Combinatorial Theory, Series B*, 70(2):265–291, 1997.

19 Francis Lazarus, Michel Pocchiola, Gert Vegter, and Anne Verroust. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the 17th Annual Symposium on Computational Geometry (SOCG)*, pages 80–89. ACM, 2001.

20 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Untangling two systems of noncrossing curves. In Stephen Wismath and Alexander Wolff, editors, *Graph Drawing*, volume 8242 of *Lecture Notes in Computer Science*, pages 472–483. Springer International Publishing, 2013.

21 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Embeddability in the 3-sphere is decidable. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 78–84. ACM, 2014.

22 Maryam Mirzakhani. Growth of Weil-Petersson volumes and random hyperbolic surface of large genus. *J. Differential Geom.*, 94(2):267–300, 06 2013.

23 Bojan Mohar. Drawing graphs in the hyperbolic plane. In *Graph Drawing*, pages 127–136. Springer, 1999.

24 Bojan Mohar and Carsten Thomassen. *Graphs on surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001.

25 Seiya Negami. Crossing numbers of graph embedding pairs on closed surfaces. *Journal of Graph Theory*, 36(1):8–23, 2001.

26 B. Osgood, R. Phillips, and P. Sarnak. Extremals of determinants of Laplacians. *Journal of Functional Analysis*, 80(1):148–211, 1988.

27 R. Bruce Richter and Gelasio Salazar. Two maps with large representativity on one surface. *Journal of Graph Theory*, 50(3):234–245, 2005.

28 Kenneth Stephenson. *Introduction to circle packing: The theory of discrete analytic functions*. Cambridge University Press, 2005.

29 Thom Sulanke. Note on the irreducible triangulations of the Klein bottle. *Journal of Combinatorial Theory, Series B*, 96:964–972, 2006.

30 Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

31 Roberto Tamassia. *Handbook of graph drawing and visualization*. CRC press, 2013.

32 William T. Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 13:743–768, 1963.

33 K. Wagner. Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

Simultaneous Nearest Neighbor Search*

Piotr Indyk[1], Robert Kleinberg[2], Sepideh Mahabadi[3], and Yang Yuan[4]

1 **MIT, CSAIL, Cambridge, USA**
indyk@mit.edu

2 **Cornell University, MSR, Ithaca, USA**
rdk@cs.cornell.edu

3 **MIT, CSAIL, Cambridge, USA**
mahabadi@mit.edu

4 **Cornell University, Ithaca, USA**
yy528@cornell.edu

Abstract

Motivated by applications in computer vision and databases, we introduce and study the Simultaneous Nearest Neighbor Search (SNN) problem. Given a set of data points, the goal of SNN is to design a data structure that, given a *collection* of queries, finds a *collection* of close points that are "compatible" with each other. Formally, we are given k query points $Q = q_1, \cdots, q_k$, and a compatibility graph G with vertices in Q, and the goal is to return data points $p_1, \cdots, p_k$ that minimize (i) the weighted sum of the distances from q_i to p_i and (ii) the weighted sum, over all edges (i, j) in the compatibility graph G, of the distances between p_i and p_j. The problem has several applications in computer vision and databases, where one wants to return a set of *consistent* answers to multiple related queries. Furthermore, it generalizes several well-studied computational problems, including Nearest Neighbor Search, Aggregate Nearest Neighbor Search and the 0-extension problem.

In this paper we propose and analyze the following general two-step method for designing efficient data structures for SNN. In the first step, for each query point q_i we find its (approximate) nearest neighbor point $\hat{p}_i$; this can be done efficiently using existing approximate nearest neighbor structures. In the second step, we solve an off-line optimization problem over sets $q_1, \cdots, q_k$ and $\hat{p}_1, \cdots, \hat{p}_k$; this can be done efficiently given that k is much smaller than n. Even though $\hat{p}_1, \cdots, \hat{p}_k$ might not constitute the optimal answers to queries $q_1, \cdots, q_k$, we show that, for the unweighted case, the resulting algorithm satisfies a $O(\log k / \log \log k)$-approximation guarantee. Furthermore, we show that the approximation factor can be in fact reduced to a constant for compatibility graphs frequently occurring in practice, e.g., 2D grids, 3D grids or planar graphs.

Finally, we validate our theoretical results by preliminary experiments. In particular, we show that the "empirical approximation factor" provided by the above approach is very close to 1.

1998 ACM Subject Classification F.2.2 Geometrical Problems and Computations

Keywords and phrases Approximate Nearest Neighbor, Metric Labeling, 0-extension, Simultaneous Nearest Neighbor, Group Nearest Neighbor

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.44

* This work was in part supported by NSF grant CCF 1447476 [11] and the Simons Foundation.

© Piotr Indyk, Robert Kleinberg , Sepideh Mahabadi, and Yang Yuan;
licensed under Creative Commons License CC-BY

1 Introduction

The nearest neighbor search (NN) problem is defined as follows: given a collection P of n points, build a data structure that, given any query point from some set Q, reports the data point closest to the query. The problem is of key importance in many applied areas, including computer vision, databases, information retrieval, data mining, machine learning, and signal processing. The nearest neighbor search problem, as well as its approximate variants, have been a subject of extensive studies over the last few decades, see, e.g., [6, 4, 16, 22, 21, 2] and the references therein.

Despite their success, however, the current algorithms suffer from significant theoretical and practical limitations. One of their major drawbacks is their inability to support and exploit *structure* in query sets that is often present in applications. Specifically, in many applications (notably in computer vision), queries issued to the data structure are not unrelated but instead correspond to samples taken from the same object. For example, queries can correspond to pixels or small patches taken from the same image. To ensure consistency, one needs to impose "compatibility constraints" that ensure that related queries return similar answers. Unfortunately, standard nearest neighbor data structures do not provide a clear way to enforce such constraints, as all queries are processed independently of each other.

To address this issue, we introduce the *Simultaneous Nearest Neighbor Search* (SNN) problem. Given k simultaneous query points $q_1, q_2, \cdots, q_k$, the goal of a SNN data structure is to find k points (also called *labels*) $p_1, p_2, \cdots, p_k$ in P such that (i) p_i is close to q_i, and (ii) $p_1, \cdots, p_k$ are "compatible". Formally, the compatibility is defined by a graph $G = (Q, E)$ with k vertices which is given to the data structure, along with the query points $Q = q_1, \cdots, q_k$. Furthermore, we assume that the data set P is a subset of some space X equipped with a distance function dist_X, and that we are given another metric dist_Y defined over $P \cup Q$. Given the graph G and the queries $q_1, \cdots, q_k$, the goal of the SNN data structure is to return points $p_1, \cdots, p_k$ from P that minimize the following function:

$$\sum_{i=1}^{k} \kappa_i \mathrm{dist}_Y(p_i, q_i) + \sum_{(i,j)\in E} \lambda_{i,j} \mathrm{dist}_X(p_i, p_j) \tag{1}$$

where κ_i and $\lambda_{i,j}$ are parameters defined in advance.

The above formulation captures a wide variety of applications that are not well modeled by traditional NN search. For example, many applications in computer vision involve computing nearest neighbors of pixels or image patches from the same image [14, 8, 5]. In particular, algorithms for tasks such as de-noising (removing noise from an image), restoration (replacing a deleted or occluded part of an image) or super-resolution (enhancing the resolution of an image) involve assigning "labels" to each image patch[1]. The labels could correspond to the pixel color, the enhanced image patch, etc. The label assignment should have the property that the labels are similar to the image patches they are assigned to, while at the same time the labels assigned to nearby image patches should be similar to each other. The objective function in Equation 1 directly captures these constraints.

From a theoretical perspective, Simultaneous Nearest Neighbor Search generalizes several well-studied computational problems, notably the Aggregate Nearest Neighbor problem [27,

[1] This problem has been formalized in the algorithms literature as the *metric labeling* problem [19]. The problem considered in this paper can thus be viewed as a variant of metric labeling with a very large number of labels.

25, 24, 1, 20] and the 0-extension problem [18, 10, 9, 3]. The first problem is quite similar to the basic nearest neighbor search problem over a metric dist, except that the data structure is given k queries $q_1 \cdots q_k$, and the goal is to find a data point p that minimizes the sum[2] $\sum_i \text{dist}(q_i, p)$. This objective can be easily simulated in SNN by setting $\text{dist}_Y = \text{dist}$ and $\text{dist}_X = L \cdot \text{uniform}$, where L is a very large number and $\text{uniform}(p, q)$ is the uniform metric. The 0-extension problem is a combinatorial optimization problem where the goal is to minimize an objective function quite similar to that in Equation 1. The exact definition of 0-extension as well as its connections to SNN are discussed in detail in Section 2.1.

1.1 Our results

In this paper we consider the basic case where $\text{dist}_X = \text{dist}_Y$ and $\lambda_{i,j} = \kappa_i = 1$; we refer to this variant as the *unweighted* case. Our main contribution is a general reduction that enables us to design and analyze efficient data structures for unweighted SNN. The algorithm (called *Independent Nearest Neighbors* or *INN*) consists of two steps. In the first (pruning) step, for each query point q_i we find its nearest neighbor[3] point $\hat{p}_i$; this can be done efficiently using existing nearest neighbor search data structures. In the second (optimization) step, we run an appropriate (approximation) algorithm for the SNN problem over sets $q_1, \cdots, q_k$ and $\hat{p}_1, \cdots, \hat{p}_k$; this can be done efficiently given that k is much smaller than n. We show that the resulting algorithm satisfies a $O(b \log k / \log \log k)$-approximation guarantee, where b is the approximation factor of the algorithm used in the second step. This can be further improved to $O(b\delta)$, if the metric space dist admits a *δ-padding decomposition* (see Preliminaries for more detail). The running time incurred by this algorithm is bounded by the cost of k nearest neighbor search queries in a data set of size n plus the cost of the approximation algorithm for the 0-extension problem over an input of size k. By plugging in the best nearest neighbor algorithms for dist we obtain significant running time savings if $k \ll n$.

We note that INN is somewhat similar to the belief propagation algorithm for super-resolution described in [14]. Specifically, that algorithm selects 16 closest labels for each q_i, and then chooses one of them by running a belief propagation algorithm that optimizes an objective function similar to Equation 1. However, we note that the algorithm in [14] is heuristic and is not supported by approximation guarantees.

We complement our upper bound by showing that the aforementioned reduction inherently yields super-constant approximation guarantee. Specifically, we show in the full version that, for an appropriate distance function dist, queries $q_1, \cdots, q_k$, and a label set P, the best solution to SNN with the label set restricted to $\hat{p}_1, \cdots, \hat{p}_k$ can be $\Theta(\sqrt{\log k})$ times larger than the best solution with label set equal to P. This means that even if the second step problem is solved to optimality, reducing the set of labels from P to $\hat{P}$ inherently increases the cost by a super-constant factor.

However, we further show that the aforementioned limitation can be overcome if the compatibility graph G has pseudoarboricity r (which means that each edge can be mapped to one of its endpoint vertices such that at most r edges are mapped to each vertex). Specifically, we show that if G has pseudoarboricity r, then the gap between the best solution using labels in P, and the best solution using labels in $\hat{P}$, is at most $O(r)$. Since many graphs used in practice do in fact satisfy $r = O(1)$ (e.g., 2D grids, 3D grids or planar graphs), this means that the gap is indeed constant for a wide collection of common compatibility graphs.

[2] Other aggregate functions, such as the maximum, are considered as well.
[3] Our analysis immediately extends to the case where the we compute approximate, not exact, nearest neighbors. For simplicity we focus only on the exact case in the following discussion.

In Appendix A we also present an alternative algorithm for the r-pseudoarboricity case. Similarly to INN, the algorithm computes the nearest label to each query q_i. However, the distance function used to compute the nearest neighbor involves not only the distance between q_i and a label p, but also the distances between the *neighbors* of q_i in G and p. This nearest neighbor operation can be implemented using any data structure for the Aggregate Nearest Neighbor problem [27, 25, 24, 1, 20]. Although this results in a more expensive query time, the labeling computed by this algorithm is final, i.e., there is no need for any additional postprocessing. Furthermore, the pruning gap (and therefore the final approximation ratio) of the algorithm is only $2r + 1$, which is better than our bound for INN.

Finally, we validate our theoretical results by preliminary experiments comparing our SNN data structure with an alternative (less efficient) algorithm that solves the same optimization problem using the full label set P. In our experiments we apply both algorithms to an image denoising task and measure their performance using the objective function (1). In particular, we show that the "empirical gap" incurred by the above approach, i.e, the ratio of objective function values observed in our experiments, is very close to 1.

1.2 Our techniques

We start by pointing out that SNN can be reduced to 0-extension in a "black-box" manner. Unfortunately, this reduction yields an SNN algorithm whose running time depends on the size of labels n, which could be very large; essentially this approach defeats the goal of having a data structure solving the problem. The INN algorithm overcomes this issue by reducing the number of labels from n to k. However the pruning step can increase the cost of the best solution. The ratio between the optimum cost after pruning to the optimum cost before pruning is called the *pruning gap*.

To bound the pruning gap, we again resort to existing 0-extension algorithms, albeit in a "grey box" manner. Specifically, we observe that many algorithms, such as those in [9, 3, 10, 23], proceed by first creating a label assignment in an "extended" metric space (using a LP relaxation of 0-extension), and then apply a rounding algorithm to find an actual solution. The key observation is that the correctness of the rounding step does *not* rely on the fact that the initial label assignment is optimal, but instead it works for any label assignment. We use this fact to translate the known upper bounds for the integrality gap of linear programming relaxations of 0-extension into upper bounds for the pruning gap. On the flip side, we show a lower bound (which will appear in the full version) for the pruning gap by mimicking the arguments used in [9] to lower bound the integrality gap of a 0-extension relaxation.

To overcome the lower bound, we consider the case where the compatibility graph G has pseudoarboricity r. Many graphs used in applications, such as 2D grids, 3D grids or planar graphs, have pseudoarboricity r for some constant r. We show that for such graphs the pruning gap is only $O(r)$. The proof proceeds by directly assigning labels in $\hat{P}$ to the nodes in Q and bounding the resulting cost increase. It is worth noting that the "grey box" approach outlined in the preceding paragraph, combined with Theorem 11 of [9], yields an $O(r^3)$ pruning gap for the class of $K_{r,r}$-minor-free graphs, whose pseudoarboricity is $\tilde{O}(r)$. Our $O(r)$ pruning gap not only improves this $O(r^3)$ bound in a quantitative sense, but it also applies to a much broader class of graphs. For example, three-dimensional grid graphs have pseudoarboricity 6, but the class of three-dimensional grid graphs includes graphs with $K_{r,r}$ minors for every positive integer r.

Finally, we validate our theoretical results by experiments. We focus on a simple denoising scenario where X is the pixel color space, i.e., the discrete three-dimensional space

space $\{0 \dots 255\}^3$. Each pixel in this space is parametrized by the intensity of the red, green and blue colors. We use the Euclidean norm to measure the distance between two pixels. We also let $P = X$. We consider three test images: a cartoon with an MIT logo and two natural images. For each image we add some noise and then solve the SNN problems for both the full color space P and the pruned color space $\hat{P}$. Note that since $P = X$, the set of pruned labels $\hat{P}$ simply contains all pixels present in the image.

Unfortunately, we cannot solve the problems optimally, since the best known exact algorithm takes exponential time. Instead, we run the same approximation algorithm on both instances and compare the solutions. We find that the values of the objective function for the solutions obtained using pruned labels and the full label space are equal up to a small multiplicative factor. This suggests that the empirical value of the pruning gap is very small, at least for the simple data sets that we considered.

2 Definitions and Preliminaries

We define the *Unweighted Simultaneous Nearest Neighbor* problem as follows. Let (X, dist) be a metric space and let $P \subseteq X$ be a set of n points from the space.

▶ **Definition 1.** In the *Unweighted Simultaneous Nearest Neighbor* problem, the goal is to build a data structure over a given point set P that supports the following operation. Given a set of k points $Q = \{q_1, \cdots, q_k\}$ in the metric space X, along with a graph $G = (Q, E)$ of k nodes, the goal is to report k (not necessarily unique) points from the database $p_1, \cdots, p_k \in P$ which minimize the following cost function:

$$\sum_{i=1}^{k} \text{dist}(p_i, q_i) + \sum_{(q_i,q_j)\in E} \text{dist}(p_i, p_j) \tag{2}$$

We refer to the first term in sum as the *nearest neighbor (NN)* cost, and to the second sum as the *pairwise (PW)* cost. We denote the cost of the optimal assignment from the point set P by $\text{Cost}(Q, G, P)$.

In the rest of this paper, simultaneous nearest neighbor (SNN) refers to the unweighted version of the problem (unless stated otherwise). Next, we define the *pseudoarboricity* of a graph and *r-sparse* graphs.

▶ **Definition 2.** *Pseudoarboricity* of a graph G is defined to be the minimum number r, such that the edges of the graph can be oriented to form a directed graph with out-degree at most r. In this paper, we call such graphs as *r-sparse*.

Note that given an r-sparse graph, one can map the edges to one of its endpoint vertices such that there are at most r edges mapped to each vertex. The doubling dimension of a metric space is defined as follows.

▶ **Definition 3.** The *doubling dimension* of a metric space (X, dist) is defined to be the smallest δ such that every ball in X can be covered by 2^δ balls of half the radius.

It is known that the doubling dimension of any finite metric space is $O(\log |X|)$. We then define padding decompositions.

▶ **Definition 4.** A metric space (X, dist) is δ-padded decomposable if for every r, there is a randomized partitioning of X into clusters $\mathcal{C} = \{C_i\}$ such that, each C_i has diameter at most r, and that for every $x_1, x_2 \in X$, the probability that x_1 and x_2 are in different clusters is at most $\delta \text{dist}(x_1, x_2)/r$.

It is known that any finite metric with doubling dimension δ admits an $O(\delta)$-padding decomposition [15].

2.1 0-Extension Problem

The 0-*extension* problem, first defined by Karzanov [18] is closely related to the Simultaneous Nearest Neighbor problem. In the 0-extension problem, the input is a graph $G(V, E)$ with a weight function $w(e)$, and a set of terminals $T \subseteq V$ with a metric d defined on T. The goal is to find a mapping from the vertices to the terminals $f : V \to T$ such that each terminal is mapped to itself and that the following cost function is minimized:

$$\sum_{(u,v)\in E} w(u, v) \cdot d(f(u), f(v))$$

It can be seen that this is a special case of the metric labeling problem [19] and thus a special case of the general version of the SNN problem defined by Equation 1. To see this, it is enough to let $Q = V$ and $P = T$, and let $\kappa_i = \infty$ for $q_i \in T$, $\kappa_i = 0$ for $q_i \notin T$, and $\lambda_{i,j} = w(i, j)$ in Equation 1.

Calinescu et al. [9] considered the semimetric relaxation of the LP for the 0-extension problem and gave an $O(\log |T|)$ algorithm using randomized rounding of the LP solution. They also proved an integrality ratio of $O(\sqrt{\log |T|})$ for the semimetric LP relaxation. Later Fakcharoenphol et al. [10] improved the upper-bound to $O(\log |T|/ \log \log |T|)$, and Lee and Naor [23] proved that if the metric d admits a δ-padded decomposition, then there is an $O(\delta)$-approximation algorithm for the 0-extension problem. For the finite metric spaces, this gives an $O(\delta)$ algorithm where δ is the doubling dimension of the metric space. Furthermore, the same results can be achieved using another metric relaxation (earth-mover relaxation), see [3]. Later Karloff et al. [17] proved that there is no polynomial time algorithm for 0-extension problem with approximation factor $O((\log n)^{1/4-\epsilon})$ unless $NP \subseteq DTIME(n^{poly(\log n)})$.

SNN can be reduced to 0-extension in a "black-box" manner via the following lemma whose proof will appear in the full version.

▶ **Lemma 5.** *Any b-approximate algorithm for the 0-extension problem yields an $O(b)$-approximate algorithm for the SNN problem.*

By plugging in the known 0-extension algorithms cited earlier we obtain the following:

▶ **Corollary 6.** *There exists an $O(\log n/ \log \log n)$ approximation algorithm for the SNN problem with running time $n^{O(1)}$, where n is the size of the label set.*

▶ **Corollary 7.** *If the metric space (X, dist) is δ-padded decomposable, then there exists an $O(\delta)$ approximation algorithm for the SNN problem with running time $n^{O(1)}$. For finite metric spaces X, δ could represent the doubling dimension of the metric space (or equivalently the doubling dimension of $P \cup Q$).*

Unfortunately, this reduction yields a SNN algorithm with running time depending on the size of labels n, which could be very large. In the next section we show how to improve the running time by reducing the labels set size from n to k. However, unlike the reduction in this section, our new reduction will no longer be "black-box". Instead, its analysis will use *particular properties* of the 0-extension algorithms. Fortunately those properties are satisfied by the known approximation algorithms for this problem.

Algorithm 1 Independent Nearest Neighbors (INN) Algorithm

Input $Q = \{q_1, \cdots, q_k\}$, and input graph $G = (Q, E)$

1: **for** $i = 1$ **to** k **do**
2: Query the NN data structure to extract a nearest neighbor (or approximate nearest neighbor) $\hat{p}_i$ for q_i
3: **end for**
4: Find the optimal (or approximately optimal) solution among the set $\hat{P} = \{\hat{p_1}, \cdots, \hat{p_k}\}$.

3 Independent Nearest Neighbors Algorithm

In this section, we consider a natural and general algorithm for the SNN problem, which we call *Independent Nearest Neighbors (INN)*. The algorithm proceeds as follows. Given the query points $Q = \{q_1, \cdots, q_k\}$, for each q_i the algorithm picks its (approximate) nearest neighbor $\hat{p}_i$. Then it solves the problem over the set $\hat{P} = \{\hat{p_1}, \cdots, \hat{p_k}\}$ instead of P. This simple approach reduces the size of search space from n down to k.

The details of the algorithm are shown in Algorithm 1.

In the rest of the section we analyze the quality of this pruning step. More specifically, we define the *pruning gap* of the algorithm as the ratio of the optimal cost function using the points in $\hat{P}$ over its value using the original point set P.

▶ **Definition 8.** The *pruning gap* of an instance of SNN is defined as $\alpha(Q, G, P) = \frac{\text{Cost}(Q,G,\hat{P})}{\text{Cost}(Q,G,P)}$. We define the pruning gap of the INN algorithm, α, as the largest value of $\alpha(Q, G, P)$ over all instances.

First, in Section 3.1, by proving a reduction from algorithms for rounding the LP solution of the 0-extension problem, we show that for arbitrary graphs G, we have $\alpha = O(\log k / \log\log k)$, and if the metric (X, dist) is δ-padded decomposable, we have $\alpha = O(\delta)$ (for example, for finite metric spaces X, δ can represent the doubling dimension of the metric space). Then, in Section 3.2, we prove that $\alpha = O(r)$ where r is the pseudoarboricity of the graph G. This would show that for the sparse graphs, the pruning gap remains constant. Finally, in the full version, we present a lower bound showing that the pruning gap could be as large as $\Omega(\sqrt{\log k})$ and as large as $\Omega(r)$ for $(r \leq \sqrt{\log k})$. Therefore, we get the following theorem.

▶ **Theorem 9.** *The following bounds hold for the pruning gap of the INN algorithm. First we have* $\alpha = O(\frac{\log k}{\log\log k})$*, and that if metric* (X, dist) *is* δ*-padded decomposable, we have* $\alpha = O(\delta)$*. Second,* $\alpha = O(r)$ *where* r *is the pseudoarboricity of the graph* G*. Finally, we have that* $\alpha = \Omega(\sqrt{\log k})$ *and* $\alpha = \Omega(r)$ *for* $r \leq \sqrt{\log k}$*.*

Note that the above theorem results in an $O(b \cdot \alpha)$ time algorithm for the SNN problem where b is the approximation factor of the algorithm used to solve the metric labeling problem for the set $\hat{P}$, as noted in line 4 of the INN algorithm. For example in a general graph b would be $O(\log k / \log\log k)$ that is added on top of $O(\alpha)$ approximation of the pruning step.

3.1 Bounding the pruning gap using 0-extension

In this section we show upper bounds for the pruning gap (α) of the INN algorithm. The proofs use specific properties of existing algorithms for the 0-extension problem.

▶ **Definition 10.** We say an algorithm A for the 0-extension problem is a β*-natural rounding* algorithm if, given a graph $G = (V, E)$, a set of terminals $T \subseteq V$, a metric space (X, d_X), and a mapping $\mu : V \to X$, it outputs another mapping $\nu : V \to X$ with the following properties:

- $\forall t \in T : \nu(t) = \mu(t)$
- $\forall v \in V : \exists t \in T$ s.t. $\nu(v) = \mu(t)$
- $\text{Cost}(\nu) \leq \beta \text{Cost}(\mu)$, i.e., $\sum_{(u,v)\in E} d_X(\nu(u), \nu(v)) \leq \beta \cdot \sum_{(u,v)\in E} d_X(\mu(u), \mu(v))$

Many previous algorithms for the 0-extension problem, such as [9, 3, 10, 23], first create the mapping μ using some LP relaxation of 0-extension (such as semimetric relaxation or earth-mover relaxation), and then apply a β-natural rounding algorithm for the 0-extension to find the mapping ν which yields the solution to the 0-extension problem. Below we give a formal connection between guarantees of these rounding algorithms, and the quality of the output of the INN algorithm (the pruning gap of INN).

▶ **Lemma 11.** *Let A be a β-natural rounding algorithm for the 0-extension problem. Then we can infer that the pruning gap of the INN algorithm is $O(\beta)$, that is, $\alpha = O(\beta)$.*

Proof. Fix any SNN instance (Q, G_S, P), where $G_S = (Q, E_{PW})$, and its corresponding INN invocation.

We construct the inputs to the algorithm A from the INN instance as follows. Let the metric space of A be the same as (X, dist) defined in the SNN instance. Also, let V be a set of $2k$ vertices corresponding to $\hat{P} \cup P^*$ with T corresponding to $\hat{P}$. Here $P^* = \{p_1^*, \cdots, p_k^*\}$ is the set of the optimal solutions of SNN, and $\hat{P}$ is the set of nearest neighbors as defined by INN. The mapping μ simply maps each vertex from $V = \hat{P} \cup P^*$ to itself in the metric X defined in SNN. Moreover, the graph $G = (V, E)$ is defined such that $E = \{(\hat{p}_i, p_i^*) | 1 \leq i \leq k\} \cup \{(p_i^*, p_j^*) | (q_i, q_j) \in E_{PW}\}$.

First we claim the following (note that $\text{Cost}(\mu)$ is defined in Definition 10, and that by definition $\text{Cost}(Q, G_S, P) = \text{Cost}(Q, G_S, P^*)$)

$$\text{Cost}(\mu) \leq 2\text{Cost}(Q, G_S, P^*) = 2\text{Cost}(Q, G_S, P)$$

We know that $\text{Cost}(Q, G_S, P^*)$ can be split into NN cost and PW cost. We can also split $\text{Cost}(\mu)$ into NN cost (corresponding to edge set $\{(\hat{p}_i, p_i^*) | 1 \leq i \leq k\}$) and PW cost (corresponding to edge set $\{(p_i^*, p_j^*) | (q_i, q_j) \in E_{PW}\}$). By definition we know the PW costs of $\text{Cost}(Q, G_S, P)$ and $\text{Cost}(\mu)$ are equal. For NN cost, by triangle inequality, we know $\text{dist}(\hat{p}_i, p_i^*) \leq \text{dist}(\hat{p}_i, q_i) + \text{dist}(q_i, p_i^*) \leq 2 \cdot \text{dist}(q_i, p_i^*)$. Here we use the fact that $\hat{p}_i$ is the nearest database point of q_i. Thus, the claim follows.

We then apply algorithm A to get the mapping ν. By the assumption on A, we know that $\text{Cost}(\nu) \leq \beta \text{Cost}(\mu)$. Given the mapping ν by the algorithm A, consider the assignment in the SNN instance where each query q_i is mapped to $\nu(p_i^*)$, and note that since $\nu(p_i^*) \in T$, this would map all points q_i to points in $\hat{P}$. Thus, by definition, we have that

$$\begin{aligned}
\text{Cost}(Q, G_S, \hat{P}) &\leq \sum_{i=1}^{k} \text{dist}(q_i, \nu(p_i^*)) + \sum_{(q_i,q_j)\in E_{PW}} \text{dist}(\nu(p_i^*), \nu(p_j^*)) \\
&\leq \sum_{i=1}^{k} \text{dist}(q_i, \hat{p}_i) + \sum_{i=1}^{k} \text{dist}(\hat{p}_i, \nu(p_i^*)) + \sum_{(q_i,q_j)\in E_{PW}} \text{dist}(\nu(p_i^*), \nu(p_j^*)) \\
&\leq \sum_{i=1}^{k} \text{dist}(q_i, \hat{p}_i) + \text{Cost}(\nu) \\
&\leq \text{Cost}(Q, G_S, P) + \beta \text{Cost}(\mu) \\
&\leq (2\beta + 1)\text{Cost}(Q, G_S, P)
\end{aligned}$$

where we have used the triangle inequality. Therefore, we have that the pruning gap α of the INN algorithm is $O(\beta)$, as claimed. ◀

Algorithm 2 r-Sparse Graph Assignment Algorithm

Input Query points $q_1, \cdots, q_k$, Optimal assignment $p^*_1, \cdots, p^*_k$, Nearest Neighbors $\hat{p_1}, \cdots, \hat{p_k}$, and the input graph $G = (Q, E)$
Output An Assignment $p_1, \cdots, p_k \in \hat{P}$

1: **for** $i = 1$ **to** k **do**
2: Let $j_0 = i$ and let $q_{j_1}, \cdots, q_{j_t}$ be all the neighbors of q_i in the graph G
3: $m \leftarrow \arg\min_{\ell=0}^{t} \text{dist}(p^*_i, p^*_{j_\ell}) + \text{dist}(p^*_{j_\ell}, q_{j_\ell})$
4: Assign $p_i \leftarrow \hat{p}_{j_m}$
5: **end for**

Using the previously cited results, and noting that in the above instance $|V| = O(k)$, we get the following corollaries.

▶ **Corollary 12.** *The INN algorithm has pruning gap $\alpha = O(\log k / \log\log k)$.*

▶ **Corollary 13.** *If the metric space (X, dist) admits a δ-padding decomposition, then the INN algorithm has pruning gap $\alpha = O(\delta)$. For finite metric spaces (X, dist), δ is at most the doubling dimension of the metric space.*

3.2 Sparse Graphs

In this section, we prove that the INN algorithm performs well on sparse graphs. More specifically, here we prove that when the graph G is r-sparse, then $\alpha(Q, G, P) = O(r)$. To this end, we show that there exists an assignment using the points in $\hat{P}$ whose cost function is within $O(r)$ of the optimal solution using the points in the original data set P.

Given a graph G of pseudoarboricity r, we know that we can map each edge to one of its end points such that the number of edges mapped to each vertex is at most r. For each edge e, we call the vertex that e is mapped to as the *corresponding vertex* of e. This would mean that each vertex is the corresponding vertex of at most r edges.

Let $p^*_1, \cdots, p^*_k \in P$ denote the optimal solution of SNN. Algorithm 2 shows how to find an assignment $p_1, \cdots, p_k \in \hat{P}$. We show that the cost of this assignment is within a factor $O(r)$ from the optimum.

▶ **Lemma 14.** *The assignment defined by Algorithm 2, has $O(r)$ approximation factor.*

Proof. For each $q_i \in Q$, let $y_i = \text{dist}(p^*_i, q_i)$ and for each edge $e = (q_i, q_j) \in E$ let $x_e = \text{dist}(p^*_i, p^*_j)$. Also let $Y = \sum_{i=1}^{k} y_i$ and $X = \sum_{e \in E} x_e$. Note that Y is the NN cost and X is the PW cost of the optimal assignment and that $OPT = \text{Cost}(Q, G, P) = X + Y$. Define the variables y'_i, x'_e, Y' , X' in the same way but for the assignment $p_1, \cdots, p_k$ produced by the algorithm. That is, for each $q_i \in Q$, $y'_i = \text{dist}(p_i, q_i)$, and for each edge $e = (q_i, q_j) \in E$, $x'_e = \text{dist}(p_i, p_j)$. Moreover, for a vertex q_i, we define the *designated neighbor* of q_i to be q_{j_m} for the value of m defined in the line 3 of Algorithm 2 (note that the designated neighbor might be the vertex itself). Fix a vertex q_i and let q_c be the designated neighbor of q_i. We can bound the value of y'_i as follows.

$$
\begin{aligned}
y'_i &= \text{dist}(q_i, p_i) \\
&= \text{dist}(q_i, \hat{p}_c) \\
&\le \text{dist}(q_i, p^*_i) + \text{dist}(p^*_i, p^*_c) + \text{dist}(p^*_c, q_c) + \text{dist}(q_c, \hat{p}_c) \quad \text{(by triangle inequality)} \\
&\le y_i + \text{dist}(p^*_i, p^*_c) + 2\text{dist}(p^*_c, q_c) \quad \text{(since } \hat{p_c} \text{ is the nearest neighbor of } q_c\text{)} \\
&\le y_i + 2[\text{dist}(p^*_i, p^*_c) + \text{dist}(p^*_c, q_c)] \\
&\le 3y_i \quad \text{(by definition of designated neighbor and the value } m \text{ in line 3 of Algorithm 2)}
\end{aligned}
$$

Thus summing over all vertices, we get that $Y' \leq 3Y$. Now for any fixed edge $e = (q_i, q_s)$ (with q_i being its corresponding vertex), let q_c be the designated neighbor of q_i, and q_z be the designated neighbor of q_s. Then we bound the value of x'_e as follows.

$$\begin{aligned}
x'_e &= \mathrm{dist}(p_i, p_s) \\
&= \mathrm{dist}(\hat{p}_c, \hat{p}_z) \quad \text{(by definition of designated neighbor and line 4 of Algorithm 2)} \\
&\leq \mathrm{dist}(\hat{p}_c, q_c) + \mathrm{dist}(q_c, p^*_c) + \mathrm{dist}(p^*_c, p^*_i) + \mathrm{dist}(p^*_i, p^*_s) \\
&+ \mathrm{dist}(p^*_s, p^*_z) + \mathrm{dist}(p^*_z, q_z) + \mathrm{dist}(q_z, \hat{p}_z) \quad \text{(by triangle inequality)} \\
&\leq 2\mathrm{dist}(q_c, p^*_c) + \mathrm{dist}(p^*_c, p^*_i) + \mathrm{dist}(p^*_i, p^*_s) \\
&+ \mathrm{dist}(p^*_s, p^*_z) + 2\mathrm{dist}(p^*_z, q_z) \quad \text{(since } \hat{p}_c(\hat{p}_z \text{ respectively) is a NN of } q_c(q_z \text{ respectively))} \\
&\leq 2[\mathrm{dist}(q_c, p^*_c) + \mathrm{dist}(p^*_c, p^*_i)] + \mathrm{dist}(p^*_i, p^*_s) + 2[\mathrm{dist}(p^*_s, p^*_z) + \mathrm{dist}(p^*_z, q_z)] \\
&\leq 2y_i + x_e + 2[x_e + y_i] \\
&\qquad \text{(since } q_c(q_z \text{ respectively) is designated neighbor of } q_i(q_s \text{ respectively))} \\
&\leq 4(x_e + y_i)
\end{aligned}$$

Hence, summing over all the edges, since each vertex q_i is the corresponding vertex of at most r edges, we get that $X' \leq 4X + 4rY$. Therefore we have the following.

$$\mathrm{Cost}(Q, G, \hat{P}) \leq X' + Y' \leq 3Y + 4X + 4rY \leq (4r + 3) \cdot \mathrm{Cost}(Q, G, P)$$

and thus $\alpha(Q, G, P) = O(r)$. ◀

4 Experiments

We consider image denoising as an application of our algorithm. A popular approach to denoising (see e.g. [13]) is to minimize the following objective function:

$$\sum_{i \in V} \kappa_i d(q_i, p_i) + \sum_{(i,j) \in E} \lambda_{i,j} d(p_i, p_j)$$

Here q_i is the color of pixel i in the noisy image, and p_i is the color of pixel i in the output. We use the standard 4-connected neighborhood system for the edge set E, and use Euclidean distance as the distance function $d(\cdot, \cdot)$. We also set all weights κ_i and $\lambda_{i,j}$ to 1.

When the image is in grey scale, this objective function can be optimized approximately and efficiently using message passing algorithm, see e.g. [12]. However, when the image pixels are points in RGB color space, the label set becomes huge ($n = 256^3 = 16,777,216$), and most techniques for metric labeling are not feasible.

Recall that our algorithm proceeds by considering only the nearest neighbor labels of the query points, i.e., only the colors that appeared in the image. In what follows we refer to this reduced set of labels as the *image color* space, as opposed to the *full color* space where no pruning is performed.

In order to optimize the objective function efficiently, we use the technique of [13]. We first embed the original (color) metric space into a tree metric (with $O(\log n)$ distortion), and then apply a top-down divide and conquer algorithm on the tree metric, by calling the alpha-beta swap subroutine [7]. We use the random-split kd-tree for both the full color space and the image color space. When constructing the kd-tree, split each interval $[a, b]$ by selecting a random number chosen uniformly at random from the interval $[0.6a + 0.4b, 0.4a + 0.6b]$.

To evaluate the performance of the two algorithms, we use one cartoon image with MIT logo and two images from the Berkeley segmentation dataset [26] which was previously used

Table 1 The empirical values of objective functions for the respective images and algorithms.

	Avg cost for full color	Avg cost for image color	Empirical pruning gap
MIT	$341878 \pm 3.1\%$	$340477 \pm 1.1\%$	0.996
Snow	$9338604 \pm 4.5\%$	$9564288 \pm 6.2\%$	1.024
Surf	$8304184 \pm 6.6\%$	$7588244 \pm 5.1\%$	0.914

in other computer vision papers [13]. We use Matlab imnoise function to create noisy images from the original images. We run each instance 20 times, and compute both the average and the variance of the objective function (the variance is due to the random generating process of kd tree).

The results are presented in Figure 1 and Table 1. In Figure 1, one can see that the images produced by the two algorithms are comparable. The full color version seems to preserve a few more details than the image color version, but it also "hallucinates" non-existing colors to minimize the value of the objective function. The visual quality of the de-noised images can be improved by fine-tuning various parameters of the algorithms. We do not report these results here, as our goal was to compare the values of the objective function produced by the two algorithms, as opposed to developing the state of the art de-noising system.

Note that, as per Table 1, for some images the value of the objective function is sometimes *lower* for the image color space compared to the full color space. This is because we cannot solve the optimization problem exactly. In particular, using the kd tree to embed the original metric space into a tree metric is an approximate process.

4.1 De-noising with patches

To improve the quality of the de-noised images, we run the experiment for *patches* of the image, instead of pixels. Moreover, we use Algorithm 3 which implements not only a pruning step, but also computes the solution directly. In this experiment (see Figure 2 for a sample of the results), each patch (a grid of pixels) from the noisy image is a query point, and the dataset consists of available patches which we use as a substitute for a noisy patch.

In our experiment, to build the dataset, we take one image from the Berkeley segmentation data set, then add noise to the right half of the image, and try to use the patches from the left half to denoise the right half. Each patch is of size 5×5 pixels. We obtain 317×236 patches from the left half of the image and use it as the patch database. Then we apply Algorithm 3 to denoise the image. In particular, for each noisy patch q_n (out of 317×237 patches) in the right half of the image, we perform a linear scan to find the closest patch p_i from the patch database, based on the following cost function:

$$dist(q_n, p_i) + \sum_{p_j \in neighbor(q_n)} \frac{dist(p_j, p_i)}{5}$$

where $dist(p, q)$ is defined to be the sum of squares of the l_2 distances between the colors of corresponding pixels in the two patches.

After that, for each noisy patch we retrieve the closest patch from the patch database. Then for each noisy pixel x, we first identify all the noisy patches (there are at most 25 of them) that cover it. The denoised color of this pixel x is simply the average of all the corresponding pixels in those noisy patches which cover x.

Since the nearest neighbor algorithm is implemented using a linear scan, it takes around 1 hour to denoise one image. One could also apply some more advanced techniques like locality sensitive hashing to find the closest patches with much faster running time.

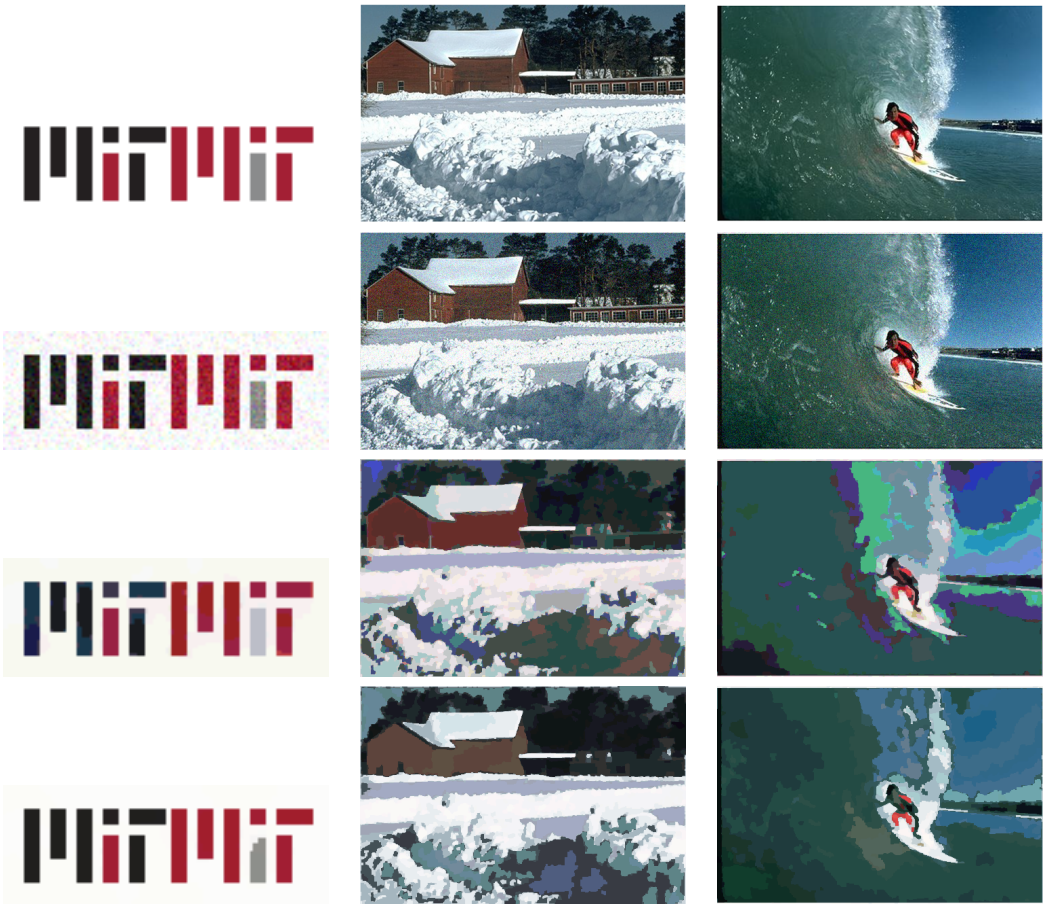


Figure 1 MIT logo (first column, size $45 * 124$), and two images from the Berkeley segmentation dataset [26] (second & third columns, size $321 * 481$). The first row shows the original image; the second row shows the noisy image; the third row shows the denoised image using full color space; the fourth row shows the denoised image using image space (our algorithm).

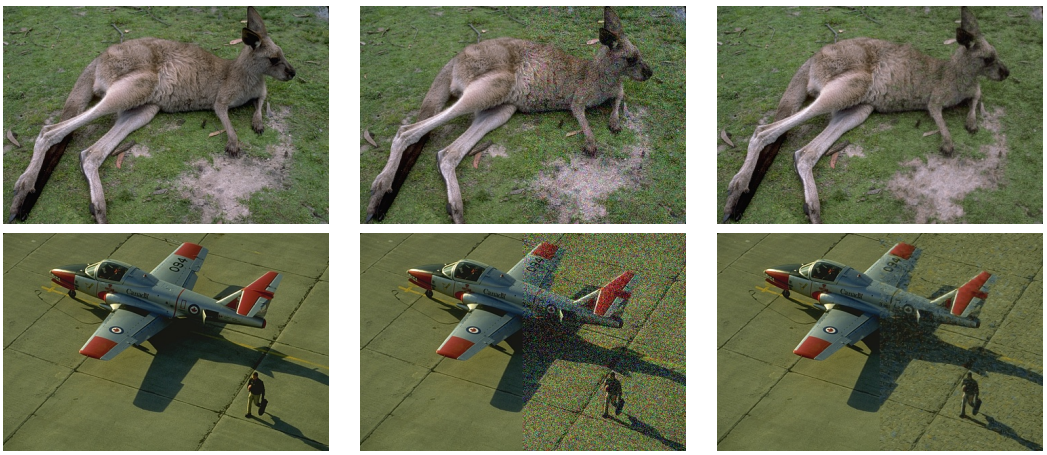


Figure 2 Two images from the Berkeley segmentation dataset [26] (size $321 * 481$). The first column shows the original image; the second column shows the half noisy image; the third column shows the de-noised image using our algorithm for the patches.

Acknowledgements. The authors would like to thank Pedro Felzenszwalb for formulating the Simultaneous Nearest Neighbor problem, as well as many helpful discussions about the experimental setup.

References

1 Pankaj K Agarwal, Alon Efrat, and Wuzhou Zhang. Nearest-neighbor searching under uncertainty. In *Proceedings of the 32nd symposium on Principles of database systems*. ACM, 2012.

2 Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. SIAM, 2014.

3 Aaron Archer, Jittat Fakcharoenphol, Chris Harrelson, Robert Krauthgamer, Kunal Talwar, and Éva Tardos. Approximate classification via earthmover metrics. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1079–1087. Society for Industrial and Applied Mathematics, 2004.

4 Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM (JACM)*, 45(6):891–923, 1998.

5 Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 28(3):24, 2009.

6 Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

7 Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

8 Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

9 Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2005.

10 Jittat Fakcharoenphol, Chris Harrelson, Satish Rao, and Kunal Talwar. An improved approximation algorithm for the 0-extension problem. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 257–265. Society for Industrial and Applied Mathematics, 2003.

11 Pedro Felzenszwalb, William Freeman, Piotr Indyk, Robert Kleinberg, and Ramin Zabih. Bigdata: F: Dka: Collaborative research: Structured nearest neighbor search in high dimensions, 2015. URL: `http://cs.brown.edu/~pff/SNN/`.

12 Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

13 Pedro F Felzenszwalb, Gyula Pap, Eva Tardos, and Ramin Zabih. Globally optimal pixel labeling algorithms for tree metrics. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3153–3160. IEEE, 2010.

14 William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56–65, 2002.

15 Anupam Gupta, Robert Krauthgamer, and James R Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 534–543. IEEE, 2003.

16 Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

17 Howard Karloff, Subhash Khot, Aranyak Mehta, and Yuval Rabani. On earthmover distance, metric labeling, and 0-extension. *SIAM Journal on Computing*, 39(2):371–387, 2009.

18 Alexander V Karzanov. Minimum 0-extensions of graph metrics. *European Journal of Combinatorics*, 19(1):71–101, 1998.

19 Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

20 Tsvi Kopelowitz and Robert Krauthgamer. Faster clustering via preprocessing. *arXiv preprint arXiv:1208.5247*, 2012.

21 Robert Krauthgamer and James R Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Society for Industrial and Applied Mathematics, 2004.

22 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

23 James R Lee and Assaf Naor. Metric decomposition, smooth measures, and clustering. *Preprint*, 2004.

24 Feifei Li, Bin Yao, and Piyush Kumar. Group enclosing queries. *Knowledge and Data Engineering, IEEE Transactions on*, 23(10):1526–1540, 2011.

25 Yang Li, Feifei Li, Ke Yi, Bin Yao, and Min Wang. Flexible aggregate similarity search. In *Proceedings of the 2011 ACM SIGMOD international conference on management of data*, pages 1009–1020. ACM, 2011.

26 David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530–549, 2004.

27 Man Lung Yiu, Nikos Mamoulis, and Dimitris Papadias. Aggregate nearest neighbor queries in road networks. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):820–833, 2005.

A $2r + 1$ approximation

Motivated by the importance of the r-sparse graphs in applications, in this section we focus on them and present another algorithm (besides INN) which solves the SNN problem for these graphs. We note that unlike INN, the algorithm presented in this section is not just a pruning step, but it solves the whole SNN problem.

For a graph $G = (Q, E)$ of pseudoarboricity r, let the mapping function be $f : E \to Q$, such that for every $e = (q_i, q_j)$, $f(e) = q_i$ or $f(e) = q_j$, and that for each $q_i \in Q$, $|C(q_i)| \leq r$, where $C(q_i)$ is defined as $\{e|f(e) = q_i\}$.

Once we have the mapping function f, we can run Algorithm 3 to get an approximate solution. Although the naive implementation of this algorithm needs $O(rkn)$ running time, by using the aggregate nearest neighbor algorithm, it can be done much more efficiently. We have the following lemma on the performance of this algorithm.

▶ **Lemma 15.** *If G has pseudoarboricity r, the solution of Algorithm 3 gives $2r + 1$ approximation to the optimal solution.*

Algorithm 3 Algorithm for graph with pseudoarboricity r

Input Query points $q_1, \cdots, q_k$, the input graph $G = (Q, E)$ with pseudoarboricity r
Output An Assignment $p_1, \cdots, p_k \in P$
1: **for** $i = 1$ **to** k **do**
2: Assign $p_i \leftarrow \min_{p \in P} \operatorname{dist}(q_i, p) + \sum_{j:(q_i,q_j)\in C(q_j)} \frac{\operatorname{dist}(p,q_j)}{r+1}$
3: **end for**

Proof. Denote the optimal solution as $P^* = \{p_1^*, \cdots, p_k^*\}$. We know the optimal cost is

$$\begin{aligned}\operatorname{Cost}(Q, G, P^*) &= \sum_i \operatorname{dist}(q_i, p_i^*) + \sum_{(q_i,q_j)\in E} \operatorname{dist}(p_i^*, p_j^*) \\ &= \sum_i \left(\operatorname{dist}(p_i^*, q_i) + \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_i^*, p_j^*) \right)\end{aligned}$$

Let Sol be the solution reported by Algorithm 3. Then we have

$$\begin{aligned}\operatorname{Cost}(\text{Sol}) &= \sum_i \left(\operatorname{dist}(q_i, p_i) + \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_i, p_j) \right) \\ &\le \sum_i \left(\operatorname{dist}(q_i, p_i) + \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_i, q_j) + \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(q_j, p_j) \right) \\ &\quad \text{(by triangle inequality)} \\ &\le \sum_i \left(\operatorname{dist}(q_i, p_i) + \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_i, q_j) \right) + r \sum_j \operatorname{dist}(q_j, p_j) \\ &\quad \text{(by definition of pseudoarboricity)} \\ &= (r+1) \sum_i \operatorname{dist}(q_i, p_i) + \sum_{(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_i, q_j) \\ &\le (r+1) \sum_i \left(\operatorname{dist}(q_i, p_i^*) + \sum_{j:(q_i,q_j)\in C(q_j)} \frac{\operatorname{dist}(p_i^*, q_j)}{r+1} \right) \\ &\quad \text{(by the optimality of } p_i \text{ in the algorithm)} \\ &\le (r+1) \sum_i \left(\operatorname{dist}(q_i, p_i^*) + \sum_{j:(q_i,q_j)\in C(q_j)} \frac{\operatorname{dist}(p_i^*, p_j^*) + \operatorname{dist}(p_j^*, q_j)}{r+1} \right) \\ &\quad \text{(by triangle inequality)} \\ &\le (r+1)\operatorname{Cost}(Q, G, P^*) + \sum_i \sum_{j:(q_i,q_j)\in C(q_j)} \operatorname{dist}(p_j^*, q_j) \\ &\le (r+1)\operatorname{Cost}(Q, G, P^*) + r \sum_j \operatorname{dist}(p_j^*, q_j) \\ &\quad \text{(by definition of pseudoarboricity)} \\ &= (2r+1)\operatorname{Cost}(Q, G, P^*)\end{aligned}$$

◀

Degree Four Plane Spanners: Simpler and Better

Iyad Kanj[1], Ljubomir Perković[2], and Duru Türkoğlu[3]

1 **School of Computing, DePaul University, Chicago, USA**
ikanj@cs.depaul.edu
2 **School of Computing, DePaul University, Chicago, USA**
lperkovic@cs.depaul.edu
3 **School of Computing, DePaul University, Chicago, USA**
dturkogl@cs.depaul.edu

Abstract

Let $\mathcal{P}$ be a set of n points embedded in the plane, and let $\mathcal{C}$ be the complete Euclidean graph whose point-set is $\mathcal{P}$. Each edge in $\mathcal{C}$ between two points p, q is realized as the line segment $[pq]$, and is assigned a weight equal to the Euclidean distance $|pq|$. In this paper, we show how to construct in $\mathcal{O}(n \lg n)$ time a plane spanner of $\mathcal{C}$ of maximum degree at most 4 and of stretch factor at most 20. This improves a long sequence of results on the construction of bounded degree plane spanners of $\mathcal{C}$. Our result matches the smallest known upper bound of 4 by Bonichon et al. on the maximum degree while significantly improving their stretch factor upper bound from 156.82 to 20. The construction of our spanner is based on Delaunay triangulations defined with respect to the equilateral-triangle distance, and uses a different approach than that used by Bonichon et al. Our approach leads to a simple and intuitive construction of a well-structured spanner, and reveals useful structural properties of the Delaunay triangulations defined with respect to the equilateral-triangle distance.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases geometric spanners; plane spanners; bounded degree spanners; Delaunay triangulations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.45

1 Introduction

Let $\mathcal{P}$ be a set of n points embedded in the plane, and let $\mathcal{C}$ be the complete Euclidean graph whose point-set is $\mathcal{P}$. Each edge in $\mathcal{C}$ between two points p, q is realized as the line segment $[pq]$, and is assigned a weight equal to the Euclidean distance $|pq|$. In this paper, we consider the problem of constructing a plane spanner of $\mathcal{C}$ of small degree and small stretch factor. This problem has received considerable attention, and there is a long list of results on the construction of plane spanners of $\mathcal{C}$ that achieve various trade-offs between the degree and the stretch factor of the spanner.

The problem of constructing a plane spanner of $\mathcal{C}$ was considered as early as the 1980's, if not earlier. Chew [10] proved that the L_1-Delaunay triangulation of $\mathcal{P}$, which is the Delaunay triangulation of $\mathcal{P}$ defined with respect to the L_1-distance, is a spanner of $\mathcal{C}$ of stretch factor at most $\sqrt{10}$. Chew's result was followed by a series of papers showing that other Delaunay triangulations are plane spanners (of $\mathcal{C}$) as well. In 1987, Dobkin *et al.* [13] showed that the classical L_2-Delaunay triangulation of $\mathcal{P}$ is a plane spanner of stretch factor at most $\frac{\pi(1+\sqrt{5})}{2}$. This bound was subsequently improved by Keil and Gutwin [16] to $\frac{4\pi}{3\sqrt{3}}$. In the meantime, Chew [11] showed that the TD-Delaunay triangulation defined using a distance


© Iyad Kanj, Ljubomir Perković, and Duru Türkoğlu;
licensed under Creative Commons License CC-BY

function based on an equilateral triangle – rather than a square (L_1-distance) or a circle (L_2-distance) – is a spanner of stretch factor 2. This result was generalized by Bose et al. [6], who showed that the Delaunay triangulation defined with respect to any convex distance function (*i.e.*, based on a convex shape) is a plane spanner. The bound on the stretch factor of the L_2-Delaunay triangulation by Keil and Gutwin stood unchallenged for many years until Xia recently improved the bound to below 2 [20]. Recently, Bonichon *et al.* [3] improved Chew's original bound on the stretch factor of the L_1-Delaunay triangulation to $\sqrt{4+2\sqrt{2}}$, and showed this bound to be tight.

All the Delaunay triangulations mentioned above can have unbounded degree. In recent years, bounded degree plane spanners have been used as the building block of wireless network topologies. Wireless distributed system technologies, such as wireless ad-hoc and sensor networks, are often modeled as proximity graphs in the Euclidean plane. Spanners of proximity graphs represent topologies that can be used for efficient communication. For these applications, in addition to having low stretch factor, spanners are typically required to be plane and have bounded degree, where both requirements are useful for efficient routing [8, 19].

The wireless network applications motivated researchers to turn their attention to minimizing the maximum degree of the plane spanner as well as its stretch factor. It can be readily seen that 3 is a lower bound on the maximum degree of a spanner of $\mathcal{C}$, because a Hamiltonian path/cycle through a set of points arranged in a grid has unbounded stretch factor. Work on bounded degree but not necessarily plane spanners of $\mathcal{C}$ closely followed the above-mentioned work on plane spanners. In a 1992 breakthrough, Salowe [18] proved the existence of spanners of maximum degree 4. The question was then resolved by Das and Heffernan [12] who showed that spanners of maximum degree 3 always exist. The focus in this line of research was to prove the existence of low degree spanners and the techniques developed to do so were not tuned towards constructing spanners that had both low degree and low stretch factor. For example, the bound on the stretch factor of the degree-4 spanner by Salowe [18] is greater than 10^9, which is far from practical. Furthermore, these bounded-degree spanners are not guaranteed to be plane.

Bose *et al.* [7] were the first to show how to extract a subgraph of the L_2-Delaunay triangulation that is a bounded-degree, plane spanner of $\mathcal{C}$. The maximum degree and stretch factor they obtained were subsequently improved by Li and Wang [17], by Bose *et al.* [9], and by Kanj and Perković [14] (see Table 1). The approach used in all these results was to extract a bounded degree spanning subgraph of the L_2-Delaunay triangulation and the main goal was to obtain a bounded-degree plane spanner of $\mathcal{C}$ with the smallest possible stretch factor. In a breakthrough result, Bonichon *et al.* [2] lowered the bound on the maximum degree of a plane spanner from 14 to 6. Instead of using the L_2-Delaunay triangulation as the starting point of the spanner construction, they used the Delaunay triangulation based on the equilateral-triangle distance, defined originally by Chew [11], and exploited a connection between these Delaunay triangulations and $\frac{1}{2}$-θ graphs. The plane spanner they constructed also has a small stretch factor of 6. Independently, Bose *et al.* [5] were also able to obtain a plane spanner of maximum degree at most 6, by starting from the L_2-Delaunay triangulation. Recently, Bonichon et al. [4] were able to construct a plane spanner of degree at most 4 and of stretch factor at most 156.82. Their construction is based on the L_1-Delaunay triangulation. Most of the above spanner constructions can be performed in time $\mathcal{O}(n \lg n)$, where n is the number of points in $\mathcal{P}$.

In this paper, we present a construction of a plane spanner $\mathcal{S}$ of $\mathcal{C}$ of degree at most 4 and of stretch factor at most 20. This result matches the smallest known upper bound of 4 on the

Table 1 Results on plane spanners with maximum degree bounded by Δ. The constant $C_0 = 1.998$ is the best known upper bound on the stretch factor of the L_2-Delaunay triangulation [20].

Paper	Δ	Stretch factor bound
Bose *et al.* [7]	27	$(\pi+1)C_0 \approx 8.27$
Li and Wang [17]	23	$(1+\pi\sin\frac{\pi}{4})C_0 \approx 6.43$
Bose *et al.* [9]	17	$(2+2\sqrt{3}+\frac{3\pi}{2}+2\pi\sin(\frac{\pi}{12}))C_0 \approx 23.56$
Kanj and Perković [14]	14	$(1+\frac{2\pi}{14\cos(\frac{\pi}{14})})C_0 \approx 2.91$
Bonichon *et al.* [2]	6	6
Bose *et al.* [5]	6	$1/(1-\tan(\pi/7)(1+1/\cos(\pi/14)))C_0 \approx 81.66$
Bonichon *et al.* [4]	4	156.82
This paper	**4**	**20**

maximum degree of the spanner by Bonichon et al. [4], while significantly improving their stretch factor bound from 156.82 to 20. Our construction is also simpler and more intuitive. It is based on Delaunay triangulations defined with respect to the equilateral-triangle distance, similar to the degree 6 spanner construction used by Bonichon et al. [2], which could be viewed as the starting point of our construction. To get down to maximum degree 4, our approach introduces fresh techniques in both the construction and the analysis of the spanner. Unlike the approach in [2], our approach has a bias – from the beginning – towards certain edges of the Delaunay triangulation; this bias ensures that the constructed spanner is well structured. To make up for edges not in the spanner, we make use of recursion which, unlike the construction in [4], may have depth not bounded by a constant. To ensure that the recursion is controlled and yields short paths, we aggressively add shortcut edges to the spanner to ensure the existence of paths with specific properties, which we refer to as monotone weak paths. Finally, in our analysis we use a new type of distance metric and we also take the extra step of analyzing the stretch factor of our spanner with respect to $\mathcal{C}$ directly, rather than with respect to the underlying Delaunay triangulation.

The structure of our spanner guarantees that if the given point-set is in convex position then the constructed spanner has maximum degree at most 3. Therefore, for any point-set in convex position, there exists a plane spanner of $\mathcal{C}$ of maximum degree at most 3. We also show that 3 is a lower bound on the maximum degree of plane spanners of $\mathcal{C}$ for point-sets in convex position. This completely and satisfactorily resolves the question about the maximum degree of plane spanners of $\mathcal{C}$ for point-sets in convex position. Due to the lack of space, the formal statement and the proof of the aforementioned result, as well as some other proofs in the paper are omitted and can be found in a full version of the paper [15].

2 Preliminaries

Given a set of points $\mathcal{P}$ embedded in the Euclidean plane, we consider the complete weighted graph $\mathcal{C}(\mathcal{P})$, or simply $\mathcal{C}$, where each edge between any two points $p, q \in \mathcal{P}$ is associated with the line segment $[pq]$, and is assigned a weight equal to the Euclidean distance $|pq|$.

Given a subgraph G of $\mathcal{C}$, G is said to be *plane* if the edges of G do not cross each other, *i.e.*, the line segments associated with the edges of G intersect only at their endpoints. The *maximum degree* of G is the maximum degree (in G) over all points in $\mathcal{P}$; we say that a family of graphs has *bounded degree* if there is an integer constant $c \geq 0$ such that every graph in the family has a maximum degree at most c.

If graph G is connected, we define the *distance* between any two points $p, q \in \mathcal{P}$, denoted $d_G(p, q)$, to be the weight of a minimum-weight path between p and q in G, where the weight of a path is the sum of the weights of its edges.

Given a constant $\rho \geq 1$, we say that G is a *ρ-spanner* of $\mathcal{C}$ if for any two points $p, q \in \mathcal{P}$, $d_G(p, q) \leq \rho \cdot |pq|$; we refer to the minimum such constant ρ as the *stretch factor* of G. We also say that a family of geometric graphs, one for every finite set $\mathcal{P}$ of points in the plane, is a *spanner* if there is a constant $\rho \geq 1$ such that every graph $G(\mathcal{P})$ in the family is a ρ-spanner of $\mathcal{C}(\mathcal{P})$; we refer to the minimum such constant ρ as the *stretch factor* of the family. In this paper, the family we construct consists of the set of spanners $G(\mathcal{P})$, where $G(P)$ is the spanner obtained by applying our algorithm to a point-set $\mathcal{P}$.

In this paper, we rely on a metric that is different from the Euclidean metric. In order to define this metric, we fix an equilateral triangle with two of its points lying on the x-axis at coordinates $(0, 0)$ and $(1, 0)$, and the third point lying below the x-axis; we use the symbol $\bigtriangledown$ to refer to this equilateral triangle. We define a triangle to be a *$\bigtriangledown$-homothet* if it can be obtained through a translation of $\bigtriangledown$ followed by a scaling. We define the *triangular metric*, $d_{\bigtriangledown}$, as follows:

▶ **Definition 1.** For any two points $p, q \in \mathcal{P}$, define $d_{\bigtriangledown}(p, q)$ to be the side-length of the smallest $\bigtriangledown$-homothet that contains p and q on its boundary; we denote this triangle $\bigtriangledown(p, q)$.

It is easy to verify that $d_{\bigtriangledown}$ is indeed a metric. In particular, for any two points p, q, we have $d_{\bigtriangledown}(p, q) = 0 \Leftrightarrow p = q$, we have symmetry as in $d_{\bigtriangledown}(p, q) = d_{\bigtriangledown}(q, p)$, and for any third point r, we have the triangle inequality $d_{\bigtriangledown}(p, q) \leq d_{\bigtriangledown}(p, r) + d_{\bigtriangledown}(r, q)$. It is also easy to see that p or q must be a vertex of the triangle $\bigtriangledown(p, q)$ and that $|pq| \leq d_{\bigtriangledown}(p, q)$.

Using the triangular metric $d_{\bigtriangledown}$, we define a subgraph $\mathcal{D}$ of $\mathcal{C}$ as follows. For every point $w \in \mathcal{P}$, we partition the space around w into six equiangular cones whose common apex is w, three above and three below the horizontal line passing through w, as illustrated in Figure 1(a). We denote the middle cone above the horizontal line and the two outer cones below the horizontal line as the *positive* cones of w, and the remaining three cones as the *negative* cones of w. Each point w chooses an edge in each of its three positive cones by selecting the point $v \neq w$ in the cone such that $d_{\bigtriangledown}(w, v)$ is minimum. Assuming that $\mathcal{P}$ is in general position[1], for any two distinct points v, v' in a positive cone of w, we obtain $d_{\bigtriangledown}(w, v) \neq d_{\bigtriangledown}(w, v')$. We define $\mathcal{D}$ to be the graph whose vertex-set is $\mathcal{P}$ and whose edge-set is the set of edges selected as described.

We make the following observation regarding the graph $\mathcal{D}$. The $\frac{1}{2}$-θ *graph* of $\mathcal{P}$ is the graph whose point-set is $\mathcal{P}$, and whose edges are obtained as follows: at each point w, and for each of the three positive cones of apex w, select the edge wv in the cone where v is the point whose projection distance to the angular bisector of the cone is minimum. Bonichon et al. [1] showed that the $\frac{1}{2}$-θ graph of $\mathcal{P}$ is the same as the *TD-Delaunay triangulation* of $\mathcal{P}$ [11] defined based on the empty triangle property: there is an edge between two points $v, w \in P$ if there exists a homothet of $\bigtriangledown$ containing v and w on its boundary whose interior is empty of points of $\mathcal{P}$. It is easy to see that the $\frac{1}{2}$-θ graph of $\mathcal{P}$ coincides with the graph $\mathcal{D}$ defined above, and hence with the TD-Delaunay triangulation[2].

---

[1] $\mathcal{P}$ is in general position if no pair of points $v, w \in \mathcal{P}$ lie on a line parallel to any of the boundary lines defining the six cones. We note that it is always possible to rotate the equilateral triangle that defines the metric $d_{\bigtriangledown}$ to ensure that the finite set $\mathcal{P}$ is in general position and so the results in this paper hold for all sets of points and not just for points in general position.

[2] A TD-Delaunay triangulation of $\mathcal{P}$ is not necessarily a triangulation of $\mathcal{P}$ as defined traditionally (a triangulation of the convex hull of the set of points). Just as Chew [11] did, we abuse the term *triangulation* because TD-Delaunay triangulations are closely related to classical L_2-Delaunay triangulations.

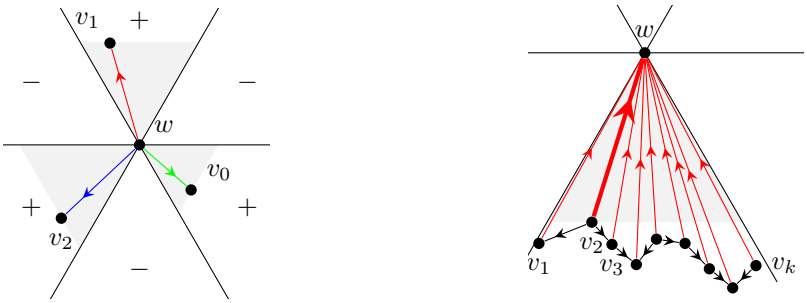


Figure 1 (a) To construct graph $\mathcal{D}$, every point w chooses the shortest edge, according to the $d_\triangledown$ distance, in every positive cone. (b) Edge (v_2, w) is the anchor of w in the negative cone shown because it is the shortest edge according to the $d_\triangledown$ distance, among edges incoming to w in the cone; the path $v_1, \ldots, v_k$ is the canonical path of anchor (v_2, w).

For convenience, we label the positive cones at each point of $\mathcal{P}$, in clockwise order and starting with the positive cone above the horizontal line, *red*, *green*, and *blue*; we also label the negative cones, in clockwise order and starting with the negative cone below the horizontal line, red, green, and blue. We assign an orientation and a color to the edges of $\mathcal{D}$ by orienting each edge outwards from the point w that selects it and by coloring it red, blue, or green depending on whether the edge lies in the positive red, blue, or green cone of point w, as illustrated in Figure 1(a). We emphasize that the edge orientations are only used for the purpose of constructing the spanner and proving its desired properties; the final spanner in our construction is an undirected graph obtained by removing edge orientations. In fact, we abuse terminology and, throughout the paper, use the term *path* to refer to weak paths in $\mathcal{D}$; we always use the term *directed path* when edge orientations are relevant.

We observe that for any point $w \in \mathcal{P}$ there is at most one edge outgoing from w in a positive cone of w, but there can be an unbounded number of edges incoming to w in a negative cone of w, and that in such cases all these edges have the same color as the cone itself (e.g., see Figure 1(b)). We follow the same approach as Bonichon et al. [2], and identify in each negative cone of point w an edge that plays a key role in the spanner construction:

▶ **Definition 2.** For any point $w \in \mathcal{P}$, and for each negative cone of w that contains at least one edge incoming to w, let (directed) edge $(v, w) \in \mathcal{D}$ be the edge in the cone such that $d_\triangledown(v, w)$ is minimum. We define (v, w) to be the *anchor* of w in the cone.

We say that anchors incident to the same point w are *adjacent* if their cones are adjacent. Note that for any two adjacent anchors incident to w, one of the two adjacent anchors must lie in a positive cone of w and must be an anchor of a point other than w.

We introduce next more terminology that we will need. Consider a negative cone of a point $w \in \mathcal{P}$ containing at least one incoming edge to w in $\mathcal{D}$. Let $(v_1, w), \ldots, (v_k, w) \in \mathcal{D}$, where $k \geq 1$, be all the incoming edges to w that lie in the cone, listed in counterclockwise order, as illustrated in Figure 1(b), and let (v_j, w), for some j such that $1 \leq j \leq k$, be the anchor of w in the cone. We call $(v_1, w), \ldots, (v_k, w)$ the *fan* of the anchor (v_j, w). We identify (v_j, w) as the anchor of each edge in the fan. Note that every edge in $\mathcal{D}$ has an anchor which could be itself. We call the first edge (v_1, w) and the last edge (v_k, w) of the fan the *boundary edges* of the anchor (v_j, w). Note that either one (possibly both) of the boundary edges of an anchor could be the anchor itself. If $k \geq 2$, since $\mathcal{D}$ is a triangulation, it follows that (v_i, v_{i+1}, w) is a triangle in $\mathcal{D}$, for $i = 1, \ldots, k-1$. Hence, $v_1, \ldots, v_k$ is a (weak)

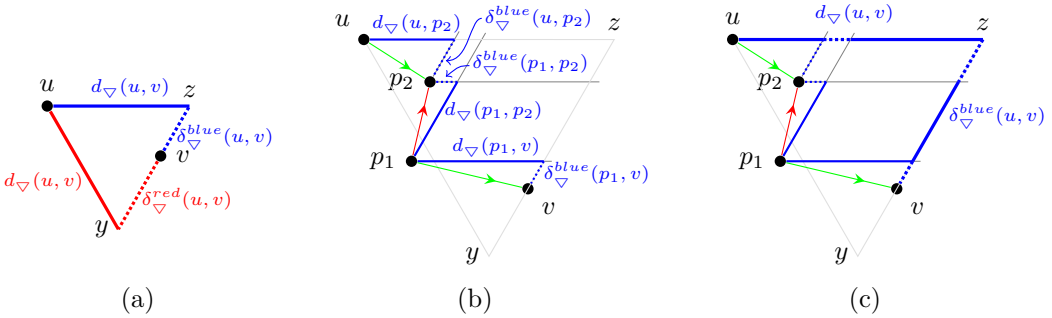


Figure 2 (a) If point v lies in the positive green cone of point u, and the vertices of $\triangledown(u,v)$, u, y, z, are colored green, red, and blue, respectively, then $d_\triangledown(u,v) = |zu|$ and $\delta^{blue}_\triangledown(u,v) = |zv|$. (b) and (c) P is a monotone path between u and v with edges colored green or red. The projection onto zu (resp. zy) of (u,p_2), (p_1,p_2), and (p_1,v) do not overlap and are contained within $[zu]$ (resp. $[zv]$).

path in $\mathcal{D}$ between the endpoints v_1 and v_k. We call this path the *canonical path* of w in the designated cone; we also call each edge on this path a *canonical edge* of w. Finally, we refer to the (weak) subpath $v_r, \ldots, v_s$ of the canonical path $v_1, \ldots, v_k$ of w as the canonical path between v_r and v_s of w. The two *sides* of an edge are the two half-planes defined by the line obtained by extending the edge. We say that a canonical edge e is canonical on a side of e if it is a canonical edge of a point that lies on that side of e. Note that a canonical edge can be canonical on both sides.

We state the following easy to verify facts without proof:

▶ **Lemma 3.** *Let (s,t) be a canonical edge of a point w, and let (s',t) be the anchor of (s,t).*
(a) *The edges (s,w) and (t,w) are in $\mathcal{D}$.*
(b) *The edge (s,w) cannot be a canonical edge on the side containing t.*
(c) *The edge (t,w) is not an anchor.*
(d) *The edge (s,t) is a boundary edge of its anchor (s',t).*

3 Monotone (weak) paths

We define next a type of path in $\mathcal{D}$ that generalizes canonical paths and that will be a key tool in our construction. We give two equivalent definitions of such a path; we leave the proof of the equivalence between these two definitions to the reader.

▶ **Definition 4.** Let v be a point lying in the positive cone of u whose color is c. A (weak) path in $\mathcal{D}$ between u and v is *monotone* if the path is bi-colored, with c being one of the colors, and the path satisfies the two equivalent properties:
- After reversing the direction of all edges not colored c, the path is directed from u to v.
- No two consecutive edges of the path lie in neighboring cones of the shared endpoint.

The key property of a monotone path between u and v is that its length can be bounded by twice the side-length of $\triangledown(u,v)$, *i.e.*, by $2d_\triangledown(u,v)$. This follows from a stronger insight which we develop next. To facilitate our discussion, we label the vertices of a $\triangledown$-homothet *green*, *blue*, and *red*, in clockwise order starting from the upper left vertex.

▶ **Definition 5.** Let v be a point lying in a positive cone of u of color c_1. With u being the vertex of $\triangledown(u,v)$ of color c_1, let y and z be the vertices of $\triangledown(u,v)$ of colors c_2 and c_3

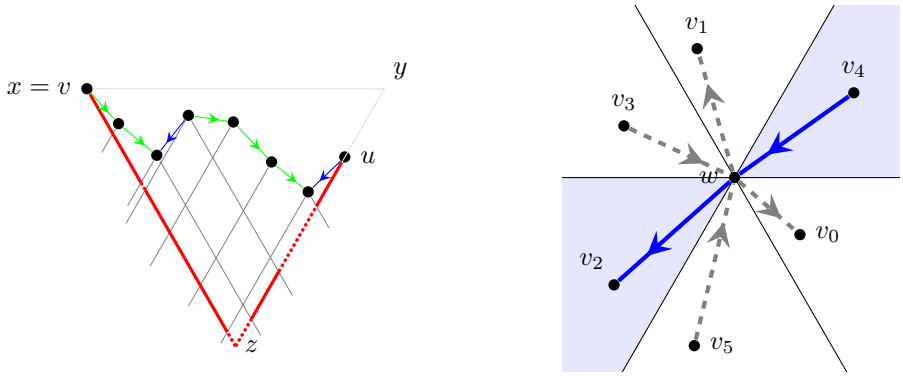


Figure 3 (a) Illustration of Lemma 7. (b) w is incident to an anchor in every cone. In step 1, both of the blue anchors are added. In step 2, on the other hand, no more than one white anchor is added below the horizontal line through w and no more than one white anchor is added above.

respectively (refer to Figure 2 where $c_1 = green$, $c_2 = red$, and $c_3 = blue$). We define the following distance functions $\delta_{\triangledown}^{c_2}, \delta_{\triangledown}^{c_3}$, and $\delta_{\triangledown}^{\min}$:

1. $\delta_{\triangledown}^{c_2}(u,v) = \delta_{\triangledown}^{c_2}(v,u) = |yv|$.
2. $\delta_{\triangledown}^{c_3}(u,v) = \delta_{\triangledown}^{c_3}(v,u) = |zv|$.
3. $\delta_{\triangledown}^{\min}(u,v) = \delta_{\triangledown}^{\min}(v,u) = \min\{\delta_{\triangledown}^{c_2}(u,v), \delta_{\triangledown}^{c_3}(u,v)\}$.

Given the assumptions of Definition 5, let P be a monotone path in $\mathcal{D}$ between u and v whose edges are colored c_1 or c_2. We define, using the lines zv and zu as axes of a coordinate system of the Euclidean plane, *the projection onto zv and onto zu*. In the following lemma, we use this projection to map the edges of P and derive an upper bound on the length of P:

▶ **Lemma 6.** *Let v be a point lying in a positive cone of u of color c_1. Let P be a monotone path between u and v whose edges are colored c_1 or c_2 (refer to Figure 2 where $c_1 = green$, $c_2 = red$, and $c_3 = blue$). With u being the vertex of $\triangledown(u,v)$ of color c_1, let z be the vertex of $\triangledown(u,v)$ of color c_3. Then, the monotone path P satisfies the following:*

- **(a)** *The projections of all edges of P onto zu (resp., zv) do not overlap and are contained within the segment $[zu]$ (resp., $[zv]$); see Figures 2(b) and 2(c).*
- **(b)** *If (p,q) is an edge of P colored c_1 (resp., c_2) then the projection onto zu (resp., zv) of (p,q) has length $d_{\triangledown}(p,q) \geq |pq|$.*
- **(c)** *The sum of the lengths of the edges of P colored c_1 is at most $d_{\triangledown}(u,v) = |zu|$.*
- **(d)** *The sum of the lengths of the edges of P colored c_2 is at most $\delta_{\triangledown}^{c_3}(u,v) = |zv|$.*
- **(e)** *The length of P is at most $d_{\triangledown}(u,v) + \delta_{\triangledown}^{c_3}(u,v) \leq 2d_{\triangledown}(u,v)$.*

Proof. For part (a), we consider the coordinates of the points of P in the coordinate system of the Euclidean plane defined by using the lines zv and zu as axes. When visiting the points of P in the order in which they appear on P, the coordinates of the points along the zu (resp., zv) axis form a monotonic sequence (decreasing or increasing) between the coordinates of u and z (resp. z and v), and part (a) follows. Since zu is parallel to an edge of $\triangledown(p,q)$, and hence the projection of (p,q) onto zu has length $d_{\triangledown}(p,q)$, part (b) follows. Parts (c) and (d) follow from parts (a) and (b), and part (e) follows from parts (c) and (d). ◀

Implied by Lemma 6, the following lemma makes explicit an insight implicit in Lemma 2 of [2] on canonical paths (see Figure 3(a)).

▶ **Lemma 7.** *For any two edges (v, w) and (u, w) that lie in the same fan:*
1. *The canonical path P between v and u is monotone.*
2. *The sum of the lengths of all monochromatic edges on P is at most $d_{\triangledown}(v, u)$.*
3. *The length of the canonical path P between v to u is at most $2d_{\triangledown}(v, u)$.*

Proof. For part *(a)*, we assume, without loss of generality, that w lies in the red positive cones of v and of u. We then observe that for every point p on P, (p, w) is an edge in $\mathcal{D}$. Therefore, every edge of P must lie in the blue or green positive cones of its tail, and thus path P is bi-colored. Furthermore, since $\mathcal{D}$ is planar, at every intermediate point p of P, the two edges of P incident to p must lie in non-adjacent cones. The canonical path P between v and u is thus monotone. Hence, parts (b) and (c) follow by Lemma 6. ◀

4 The Spanner

In this section, we describe the construction of a plane spanner of $\mathcal{C}$ of maximum degree at most 4 and stretch factor at most 20. In our construction, we will bias blue – positive and negative – cones and edges. This bias results in a spanner satisfying structural properties that allow us to prove the desired upper bounds on the spanner degree and stretch factor. These structural properties also ensure that the spanner has maximum degree at most 3 when the point-set $\mathcal{P}$ is in convex position. In the algorithm description and the remainder of the paper, we find it convenient to refer to the four non-blue cones, as well as all the red and green edges, as *white* (see Figure 3(b)). We also use some new terminology which we define next.

If e is a canonical edge of a point w that lies in a white (resp., blue) cone of w, we say that e is a canonical edge *in a white* (resp., *blue*) *cone*. We note that a canonical edge could be in a white cone of one point and in a blue cone of another. Given a white anchor (v, w), the ray starting from w extending (v, w) partitions the (white) negative cone of w containing v into two *sides*: we refer to the side of the cone that is adjacent to a blue cone as the *blue side*, and we refer to the other side that is adjacent to a white cone as the *white side*. We say that an edge (u, w) in the fan of (v, w) is on the *white side* (resp. *blue side*) if it is on the white side (resp. blue side) of (v, w).

The following describes the construction of the spanner $\mathcal{S}$ of $\mathcal{C}$. The construction is based on the underlying triangulation $\mathcal{D}$ of $\mathcal{C}$. We start by constructing a degree-4 anchor subgraph $\mathcal{A}$ of $\mathcal{S}$ that includes all blue anchors. We then augment $\mathcal{S}$ by adding *some* white canonical edges and shortcut edges.

1. We add to $\mathcal{A}$ (which is initially empty) every blue anchor.
2. In increasing order of length with respect to the metric $d_{\triangledown}$, for every white anchor a, we add a to $\mathcal{A}$ if no white anchor adjacent to a is already in $\mathcal{A}$.
3. We set $\mathcal{S}$ to $\mathcal{A}$ and then add to $\mathcal{S}$ every (white) canonical edge in a blue cone if the edge is not in $\mathcal{A}$.
4. For every pair of canonical edges $(p, q), (r, q)$ in a blue cone such that $(p, q), (r, q) \in \mathcal{S} \setminus \mathcal{A}$, we add to $\mathcal{S}$ the shortcut edge (p, r), color it white, and remove (p, q) and (r, q) from $\mathcal{S}$.
5. We add to $\mathcal{S}$ every white canonical edge that is on the white side of its (white) anchor, but only if its anchor is not in $\mathcal{A}$.
6. For every white anchor (v, w) and its boundary edge $(u, w) \neq (v, w)$ on the white side, let P be the canonical path $(u = p_0, p_1, \ldots, p_k = v)$. We apply the following procedure at a current point p_i starting with $i = 0$ and stopping when $i = k$:
   a. If the canonical edge (p_{i+1}, p_i) is white, we skip this edge and set i to $i + 1$;

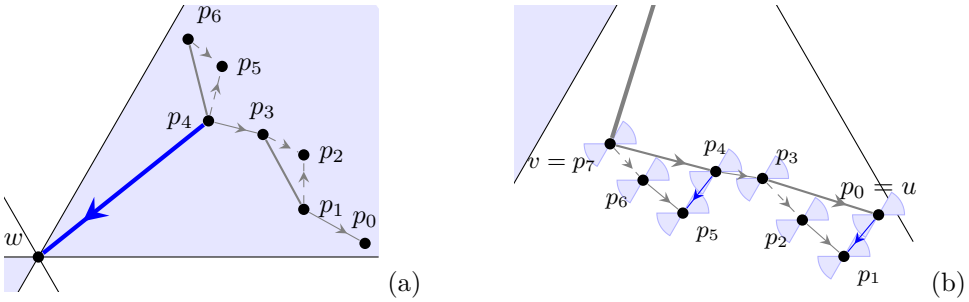


Figure 4 (a) In step 3, white canonical edges of w in the negative blue cone of w are added to $\mathcal{S}$ if not in $\mathcal{A}$ already; in step 4, any pair of canonical edges of w added in step 3 that are incoming at the same point are replaced by a shortcut between the outgoing endpoints ((p_6, p_5) and (p_4, p_5) replaced by shortcut (p_6, p_4) and $(p_{3,2})$ and (p_1, p_2) replaced by shortcut (p_3, p_1).) (b) Shortcut edges (p_3, p_0) and (p_7, p_4) are added to $\mathcal{S}$ in step 6; edges (p_7, p_6) and (p_3, p_2) are not in $\mathcal{S}$ unless they are anchors in $\mathcal{A}$.

b. Otherwise, (p_i, p_{i+1}) must be blue. Let $j > i$ be the largest index of a point on P such that the line segment $[p_i p_j]$ does not intersect the canonical path from p_i to p_j (except at p_i and p_j). We add the shortcut (p_j, p_i) to $\mathcal{S}$ and color it white; we remove the (white) canonical edge (p_j, p_{j-1}) from $\mathcal{S}$ if $(p_j, p_{j-1}) \in \mathcal{S} \setminus \mathcal{A}$; and we set i to j.

In the following section we prove that this algorithm yields a plane spanner of maximum degree at most 4 and stretch factor at most 20. We provide here a high-level overview of our arguments.

To show planarity, we note that the underlying graph $\mathcal{D}$ is planar and that the only edges of $\mathcal{S}$ not in $\mathcal{D}$ are the shortcut edges added in steps 4 and 6.b. We prove in Lemmas 10 and 11 that each such edge does not intersect any other edge of $\mathcal{S}$. For the degree upper bound, we note that the first two steps of the algorithm yield the subgraph $\mathcal{A}$ of maximum degree at most 4. In the remaining steps, we carefully add additional edges, whether canonical edges or shortcuts of canonical paths. To prove the degree bound, we develop a charging argument that assigns each edge of $\mathcal{S}$ to a cone at each endpoint and show, in Lemma 12, that no more than 4 cones are charged at every point.

To prove that $\mathcal{S}$ is a spanner, we show that every edge (u, w) in $\mathcal{D}$ but not in $\mathcal{S}$ can be *reconstructed*, by which we mean that there is a short path between u and w in $\mathcal{S}$. To do this, we consider the path between u and w in $\mathcal{D}$ consisting of the anchor (v, w) of (u, w) and the canonical path from u to v of anchor (v, w), and we argue that every edge on that path can be reconstructed. Because canonical edges are boundary edges, it is sufficient to show that all anchors and boundary edges are reconstructible.

In step 1, we add all blue anchors to $\mathcal{S}$ and in steps 3 and 4 we add to $\mathcal{S}$ all white canonical edges in blue cones, except for some consecutive pairs of canonical edges that are replaced with shortcut edges. Together, these steps ensure that (almost) all blue edges are reconstructible as we show in Lemma 13; in particular, all blue boundary edges are reconstructible.

If (u, w) is a white boundary edge, the edges on the canonical path between u and v are blue boundary edges (which are reconstructible, as discussed above) or white boundary edges on the white side of their anchor. Steps 5 and 6 ensure that white boundary edges on the white side of their anchor are reconstructible with a monotone path that is constructed recursively using shortcuts added in step 6. Therefore, if white anchor (v, w) is in $\mathcal{S}$, edge (u, w) is reconstructible as we show in Lemma 14. If (v, w) is not in $\mathcal{S}$ then (v, w) must be a

white anchor (since step 1 added all blue anchors to $\mathcal{S}$) and there must exist a shorter anchor adjacent to (v, w) in $\mathcal{S}$ (by step 2.) In Lemma 17, we show that this shorter anchor can be used to reconstruct anchor (v, w) implying that (u, w) is reconstructible as well.

5 Properties of the Spanner

In this section, we prove the three properties of the spanner $\mathcal{S}$ obtained using our algorithm: planarity, the maximum degree upper bound of 4, and the stretch factor bound of 20. We start with the following justification for coloring white the shortcut edges added in step 6:

▶ **Lemma 8.** *For every shortcut edge (p_j, p_i) added to $\mathcal{S}$ in step 6, p_j and $v = p_k$ both lie in the same negative white cone of p_i, and they both lie in the same negative white cone of p_{j-1}.*

Proof. Because $d_{\triangledown}(w, p_k) < d_{\triangledown}(w, p_i)$ and $(p_i, w) \in \mathcal{D}$, p_k must lie in a negative white cone of p_i. The lemma thus holds if $j = k$. Otherwise, by the choice of p_j, p_j must lie on the same side of line $p_i p_k$ as point w; again, because $(p_i, w) \in \mathcal{D}$, p_j must lie in a (negative) white cone of p_i that also contains p_k. Similar arguments apply to p_{j-1}. ◀

Next, we show that $\mathcal{S}$ is plane. We first need the following definition and lemma.

▶ **Definition 9.** An edge $(u, w) \in \mathcal{D}$ is *uncrossed* if no shortcut in $\mathcal{S}$ crosses (u, w).

▶ **Lemma 10.** *All anchors, all canonical edges, and all boundary edges are uncrossed.*

Proof. Let (p, r) be a shortcut that was added in step 4 of the spanner construction, let (p, q) and (r, q) be the pair of canonical edges in the blue cone as described in step 4, and let w be the apex of this blue cone. It is easy to verify that $(q, w) \in \mathcal{D}$ is the only edge in $\mathcal{D}$ that (p, r) crosses, and that (q, w) is not a boundary edge, a canonical edge, or an anchor. Next, consider a shortcut (p_j, p_i) that was added in step 6 of the spanner construction, and let (v, w) be the white anchor and $p_{i+1}, p_{i+2}, \ldots, p_{j-1}$ be the points on the canonical path between p_i and p_j as described in step 6. Again, it is easy to verify that $(p_{i+1}, w), (p_{i+2}, w), \ldots, (p_{j-1}, w)$ are the only edges in $\mathcal{D}$ that the shortcut (p_j, p_i) crosses, and that none of them is a boundary edge, a canonical edge, or an anchor. ◀

▶ **Lemma 11.** *The subgraph $\mathcal{S}$ is a plane subgraph of $\mathcal{C}$.*

Proof. Let $\mathcal{S}_1 = \mathcal{D} \cap \mathcal{S}$ and $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$. Note that $\mathcal{S}_1$ consists of $\mathcal{A}$ plus those canonical edges that are added in steps 3 or 5 and kept after steps 4 and 6. Note also that $\mathcal{S}_2$ consists only of the shortcuts which are added in steps 4 and 6. Since $\mathcal{S}_1$ is a subgraph of $\mathcal{D}$, $\mathcal{S}_1$ is plane. By Lemma 10, all the edges in $\mathcal{S}_1$ are uncrossed, *i.e.*, no shortcut (edge in $\mathcal{S}_2$) crosses an edge in $\mathcal{S}_1$. To conclude the proof, we show that no two edges in $\mathcal{S}_2$ cross either. Observe that any two shortcuts connect pairs of endpoints of canonical paths that either belong to different fans or that belong to the same fan. In the former case, the shortcuts do not cross because they belong to different fans. In the latter case, the shortcuts do not cross because shortcuts always connect the endpoints of non-overlapping canonical paths. ◀

To facilitate the discussion in the proof of the degree upper bound, we refer to the two adjacent white cones above (resp., below) the horizontal line through a point $p \in \mathcal{P}$ as the *upper* (resp., *lower*) *white sector* of p; we also refer to the two blue cones at p as the *left* and *right blue sectors* of p. We develop a charging scheme to show that, for each point p, each edge incident to p in $\mathcal{S}$ can be mapped in a one-to-one fashion to one of the four sectors at p. To describe the charging scheme for every edge $e \in \mathcal{S}$ and for every

endpoint p of e, we define $\sigma(e,p)$ to be the sector of p that contains e. Also for a point p, we denote by LB_p, RB_p, UW_p, and LW_p, the left blue, the right blue, the upper white, and the lower white sectors of p respectively. We describe in the table below the charging scheme for every edge $e=(x,y)\in\mathcal{S}$ based on which step of the construction e is added to $\mathcal{S}$.

Step	**Classification of** $e=(x,y)$	**Charge at** x	**Charge at** y
1	Blue anchor in $\mathcal{A}$	$\sigma(e,x)=LB_x$	$\sigma(e,y)=RB_y$
2	White anchor in $\mathcal{A}$	$\sigma(e,x)=UW_x$ or LW_x	$\sigma(e,y)=LW_y$ or UW_y
3	White canonical edge in a blue cone	$\sigma(e,x)=UW_x$ or LW_x	LB_y
4	(White) shortcut in a blue cone	$\sigma(e,x)=UW_x$ or LW_x	$\sigma(e,y)=LW_y$ or UW_y
5	White canonical edge in a white cone	RB_x	$\sigma(e,y)=UW_y$ or LW_y
6	(White) shortcut in a white cone	RB_x	$\sigma(e,y)=UW_y$ or LW_y

▶ **Lemma 12.** *For any point $p\in\mathcal{P}$, each sector of p is charged with at most one edge in $\mathcal{S}$. Therefore, the maximum degree of $\mathcal{S}$ is at most 4.*

The remainder of this section is devoted to proving the upper bound of 20 on the stretch factor of $\mathcal{S}$. We do so by first proving a sequence of lemmas that derive upper bounds on the distance in $\mathcal{S}$ between the endpoints of different types of edges in $\mathcal{D}$; we then use these lemmas to derive the upper bound of 20 on the stretch factor of $\mathcal{S}$.

▶ **Lemma 13.** *For any uncrossed blue edge $(u,w)\in\mathcal{D}$, $d_{\mathcal{S}}(u,w)\leq 3d_{\triangledown}(u,w)$.*

Proof. Let (v,w) be the blue anchor of the blue edge (u,w). In step 1 of the algorithm, we add all the blue anchors in $\mathcal{S}$, and thus $(v,w)\in\mathcal{S}$. Also, in step 3 of the algorithm, we add in $\mathcal{S}$ all the canonical edges in blue cones except that, in step 4, we substitute some pairs of these canonical edges with shortcuts. Since (u,w) is uncrossed, these canonical edges and shortcuts provide a path for connecting v and u. Using the triangle inequality, this path that includes the shortcuts is not longer than the canonical path between v and u. Hence, in the worst case, we may assume that the path connecting v and u consists only of canonical edges on the canonical path. This canonical path plus the anchor constitutes a path between u and w. By Lemma 7, the length of this canonical path is bounded by $2d_{\triangledown}(v,u)\leq 2d_{\triangledown}(u,w)$. We also have that $|vw|\leq d_{\triangledown}(v,w)\leq d_{\triangledown}(u,w)$. Consequently, the length of this path is bounded by $d_{\triangledown}(u,w)$ (anchor) plus $2d_{\triangledown}(u,w)$ (canonical path). It follows that $d_{\mathcal{S}}(u,w)\leq 3d_{\triangledown}(u,w)$. ◀

▶ **Lemma 14.** *For any white anchor (v,w) and any uncrossed white edge $(u,w)\in\mathcal{D}$ that lies on the white side of (v,w), $d_{\mathcal{S}}(v,u)\leq d_{\triangledown}(v,u)+\delta^{blue}_{\triangledown}(v,u)\leq 2d_{\triangledown}(v,u)$. Furthermore, if $(v,w)\in\mathcal{S}$, then $d_{\mathcal{S}}(u,w)\leq d_{\triangledown}(u,w)+\delta^{blue}_{\triangledown}(u,w)\leq 2d_{\triangledown}(u,w)$.*

Proof. We describe below how to construct a white monotone path in $\mathcal{S}$ between u and v. If $(v,w)\in\mathcal{S}$, we extend this path to a white monotone path between u and w. Then, we obtain the desired bounds using Lemma 6.

To describe the white monotone path between u and v, we consider the uncrossed edges of $\mathcal{D}$ on the fan of (v,w) whose endpoints lie on the canonical path between v and u. We observe that shortcuts and white canonical edges connect the (distinct) endpoints of those uncrossed edges, and that they form a white monotone path between u and v because at each point the white edges of the path incident to the point lie on opposite sides of the horizontal line through the point. We call this white monotone path the *white monotone connection* between v and u. We know that all of the shortcuts on this white monotone connection are in $\mathcal{S}$ and even though some of the white canonical edges may not be in $\mathcal{S}$, we know, for all such white canonical edges, that we have their anchors in $\mathcal{S}$. For each such white canonical edge (s,t), we recursively expand the current white monotone path by including the anchor

(r, t) of (s, t) and by including the white monotone connection between r and s. We point out that the path obtained after the expansion of (s, t) continues to be a white monotone path. This is because the anchor (r, t) already conforms to the existing white monotone path, and so does the white monotone connection between r and s. Therefore, by recursively expanding this path for white canonical edges that are not in $\mathcal{S}$, we obtain a white monotone path between v and u. Furthermore, if $(v, w) \in \mathcal{S}$, we expand this path to include the white anchor (v, w) while preserving its monotonicity. ◀

▶ **Lemma 15.** *For any white anchor (v, w) and any white edge $(u, w) \in \mathcal{D}$ that lies on the blue side of (v, w), $d_{\mathcal{S}}(v, u) \leq 5d_{\triangledown}(v, u)$. Furthermore, if $(v, w) \in \mathcal{S}$, then $d_{\mathcal{S}}(u, w) \leq 6d_{\triangledown}(u, w)$.*

Proof. The canonical path from v to u consists of blue and white canonical edges. The total length of the blue canonical edges does not exceed $d_{\triangledown}(v, u)$, and the total length of the white canonical edges does not exceed $d_{\triangledown}(v, u)$ by Lemma 7. By Lemma 10, we know that all of these canonical edges are uncrossed. By Lemma 13, the total length of the paths needed to reconstruct these blue canonical edges can be bounded by $3d_{\triangledown}(v, u)$. Also, since either the white canonical edges themselves or their anchors are in $\mathcal{S}$, the total length of the white canonical edges can be bounded by $2d_{\triangledown}(v, u)$ by Lemma 14. Therefore, $d_{\mathcal{S}}(v, u)$ can be bounded by $5d_{\triangledown}(v, u)$ for the edge (v, u) as stated. Furthermore, if $(v, w) \in \mathcal{S}$, $d_{\mathcal{S}}(u, w)$ can be bounded by $5d_{\triangledown}(v, u) + d_{\triangledown}(v, w)$, which in turn is bounded by $6d_{\triangledown}(u, w)$. ◀

▶ **Definition 16.** For any two points $p, q \in \mathcal{P}$ such that p lies in a white cone of q, we define $\delta^{white}_{\triangledown}(p, q) = \delta^{white}_{\triangledown}(q, p) = d_{\triangledown}(p, q) - \delta^{blue}_{\triangledown}(p, q)$.

▶ **Lemma 17.** *For any white anchor (v, w), $d_{\mathcal{S}}(v, w) \leq 9d_{\triangledown}(v, w)$. Furthermore, for any uncrossed white edge (u, w) in the fan of (v, w), we have $d_{\mathcal{S}}(u, w) \leq 9d_{\triangledown}(u, w) + \delta^{blue}_{\triangledown}(u, w)$ if (u, w) lies on the white side of (v, w), and $d_{\mathcal{S}}(u, w) \leq 9d_{\triangledown}(u, w)$ otherwise.*

Proof. If $(v, w) \in \mathcal{S}$, then clearly $d_{\mathcal{S}}(v, w) \leq d_{\triangledown}(v, w)$. As for any uncrossed edge (u, w) in the fan, by Lemma 14, we get a bound of $2d_{\triangledown}(u, w)$ on $d_{\mathcal{S}}(u, w)$ if (u, w) lies on the white side of (v, w), and by Lemma 15, we get a bound of $6d_{\triangledown}(u, w)$ on $d_{\mathcal{S}}(u, w)$ if (u, w) lies on the blue side of (v, w). Then, we consider $(v, w) \notin \mathcal{S}$ and analyze two cases: (v, w) was not added in $\mathcal{A}$ because of an adjacent anchor at w, or because of an adjacent anchor at v.

If (v, w) was not added in $\mathcal{A}$ because of an adjacent (white) anchor at w, let (w, w') be that anchor (see Figure 5(a)). By our construction of $\mathcal{A}$, we know that (w, w') must be shorter than (v, w), *i.e.*, $d_{\triangledown}(w, w') < d_{\triangledown}(v, w)$. Therefore, v lies in the positive blue cone of w', and hence, there must be an outgoing blue edge at w'. Let (w', u') be that blue edge; then, (u', w) must be a white boundary edge of (v, w), and possibly $u' = v$. Using the fact that u' lies in the positive blue cone of w', it is easy to verify that $d_{\triangledown}(v, u') \leq \delta^{white}_{\triangledown}(v, w) \leq d_{\triangledown}(v, w)$. Similarly, using the fact that u' lies in a positive white cone of v, and that $d_{\triangledown}(w, w') < d_{\triangledown}(v, w)$, it is easy to verify that $d_{\triangledown}(w', u') \leq d_{\triangledown}(w, w') + \delta^{white}_{\triangledown}(v, w) < 2d_{\triangledown}(v, w)$. Following the path from w to w' to u' to v, we bound $d_{\mathcal{S}}(v, w)$ by $d_{\triangledown}(w, w')$ for the edge (w, w'), by $3d_{\triangledown}(w', u')$ for the edge (w', u') using Lemmas 10 and 13, and by $2d_{\triangledown}(v, u')$ for the edge (v, u') using Lemmas 10 and 14. Also, using the above inequalities $d_{\triangledown}(w, w') < d_{\triangledown}(v, w)$, $d_{\triangledown}(v, u') \leq d_{\triangledown}(v, w)$, and $d_{\triangledown}(w', u') \leq 2d_{\triangledown}(v, w)$, we get the desired upper bound $d_{\mathcal{S}}(v, w) \leq 9d_{\triangledown}(v, w)$.

Next, we consider the uncrossed white edges. For any uncrossed edge (u, w) on the white side of the anchor, the bound on $d_{\mathcal{S}}(v, w)$ applies directly to $d_{\mathcal{S}}(u, w)$ because the path between v and w already connects u and w. Also, since $d_{\triangledown}(v, w) \leq d_{\triangledown}(u, w)$, we immediately get $d_{\mathcal{S}}(u, w) \leq 9d_{\triangledown}(u, w)$. As for any white edge (u, w) on the blue side of the anchor, we start by observing that $\delta^{white}_{\triangledown}(v, w) \leq d_{\triangledown}(u, w) - d_{\triangledown}(v, u)$, and that $d_{\triangledown}(v, w) \leq d_{\triangledown}(u, w)$. Also, using the above inequalities $d_{\triangledown}(w, w') < d_{\triangledown}(v, w)$, and $d_{\triangledown}(v, u') \leq \delta^{white}_{\triangledown}(v, w)$, and

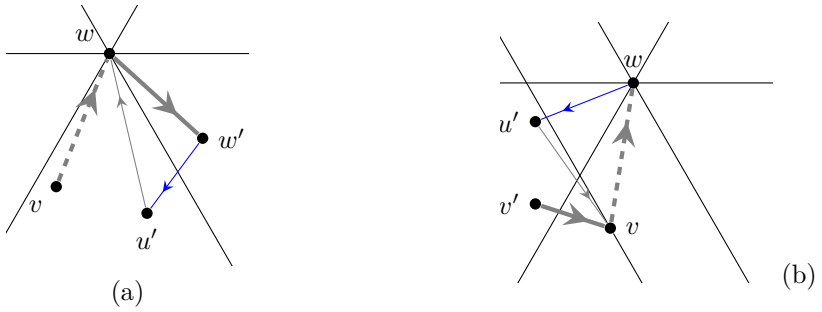


Figure 5 Illustrations of the proof of Lemma 17. (a) The case when $(v, w) \notin \mathcal{A}$ because a shorter adjacent anchor (w, w') was added first. Edge (w', u') is a blue boundary edge and there is a white monotone path from u' to v in $\mathcal{S}$. (b) The case when $(v, w) \notin \mathcal{A}$ because a shorter adjacent anchor (v', v) was added first. Edge (w, u') is a blue boundary edge and (u', v) is a white boundary edge on the white side of its cone and there is a white monotone path between u' and v' in $\mathcal{S}$.

$d_{\triangledown}(w', u') \leq d_{\triangledown}(w, w') + \delta^{white}_{\triangledown}(v, w)$, we obtain the inequalities $d_{\triangledown}(w, w') \leq d_{\triangledown}(u, w)$, and $d_{\triangledown}(v, u') \leq d_{\triangledown}(u, w) - d_{\triangledown}(v, u)$, and $d_{\triangledown}(w', u') \leq 2d_{\triangledown}(u, w) - d_{\triangledown}(v, u)$. Then, following the above path from w to v and extending it to u using the canonical edges, we get $d_{\mathcal{S}}(u, w) \leq d_{\triangledown}(w, w') + 3(d_{\triangledown}(w', u') + d_{\triangledown}(v, u)) + 2(d_{\triangledown}(v, u') + d_{\triangledown}(v, u))$, which is then bounded by $d_{\mathcal{S}}(u, w) \leq d_{\triangledown}(u, w) + 6d_{\triangledown}(u, w) + 2d_{\triangledown}(u, w) = 9d_{\triangledown}(u, w)$.

In the case when (v, w) was not added in $\mathcal{A}$ because of an adjacent (white) anchor at v, (see Figure 5(b)) one can obtain the desired bounds using similar analysis. ◀

▶ **Lemma 18.** *For any crossed blue edge $(u, w) \in \mathcal{D}$, $d_{\mathcal{S}}(u, w) \leq 3d_{\triangledown}(u, w) + 9\delta^{\min}_{\triangledown}(u, w)$.*

Proof. Let (p, q) be a shortcut that crosses (u, w). Since this shortcut is in the blue cone, it must have been added in $\mathcal{S}$ replacing two white canonical edges incoming at u, namely (p, u) and (q, u). As p and q are endpoints of the shortcut (p, q), both blue edges (p, w) and (q, w) are uncrossed. Consequently, we have $d_{\mathcal{S}}(p, w) \leq 3d_{\triangledown}(p, w)$ and $d_{\mathcal{S}}(q, w) \leq 3d_{\triangledown}(q, w)$ by Lemma 13. Furthermore, by Lemmas 10 and 17 we know that both of the canonical edges (p, u) and (q, u) satisfy the inequalities $d_{\mathcal{S}}(p, u) \leq 9d_{\triangledown}(p, u)$ and $d_{\mathcal{S}}(q, u) \leq 9d_{\triangledown}(q, u)$. Since both of these canonical edges are incoming at u, we also know that one of them, say (p, u), is not longer than $\delta^{\min}_{\triangledown}(u, w)$, i.e., $d_{\triangledown}(p, u) \leq \delta^{\min}_{\triangledown}(u, w)$. Following the path from w to p to u, we bound $d_{\mathcal{S}}(u, w)$ by $3d_{\triangledown}(p, w) + 9d_{\triangledown}(p, u)$, which in turn is bounded by $3d_{\triangledown}(u, w) + 9\delta^{\min}_{\triangledown}(u, w)$ as stated in the lemma. ◀

Due to space constraints, we omit the technical proof of the following lemma on crossed white edges; the interested reader can find the proof in the full version of the paper.

▶ **Lemma 19.** *For any crossed white edge $(u, w) \in \mathcal{D}, d_{\mathcal{S}}(u, w) \leq 10d_{\triangledown}(u, w) + 10\delta^{\min}_{\triangledown}(u, w)$.*

By Lemmas 13, 17, 18, and 19 we have that $\mathcal{S}$ is a 20-spanner of $\mathcal{D}$. Since $\mathcal{D}$ is a 2-spanner of $\mathcal{C}$ ([11]) it follows that $\mathcal{S}$ is a 40-spanner of $\mathcal{C}$. We prove, however, a much better stretch factor upper bound of 20 next.

▶ **Lemma 20.** *For any two points $p, q \in \mathcal{P}$, $d_{\mathcal{S}}(p, q)$ is bounded by $20|pq|$.*

Proof. We prove the lemma by first constructing, in $\mathcal{D}$, a monotone path π between p and q that lies inside $\triangledown(p, q)$. We then consider the path π' in $\mathcal{S}$ obtained by replacing every edge of π not in $\mathcal{S}$ with a short path in $\mathcal{S}$.

We define the path π between p and q consisting of k edges in $\mathcal{D}$ using a sequence of pairs of points $\{p,q\} = \{p_0,q_0\},\{p_1,q_1\},\ldots,\{p_k,q_k\}$ such that any two consecutive pairs of points $\{p_{i-1},q_{i-1}\}$ and $\{p_i,q_i\}$ satisfy exactly one of the equations $p_i = p_{i-1}$ and $q_i = q_{i-1}$ and the equation that is not satisfied describes the i^{th} edge. If $p_i \neq p_{i-1}$, then the i^{th} edge is $(p_{i-1},p_i) \in \mathcal{D}$, otherwise the i^{th} edge is $(q_{i-1},q_i) \in \mathcal{D}$. We define this sequence recursively for the next pair of points $\{p_{i+1},q_{i+1}\}$ by first identifying which of the points p_i and q_i lie in the other's positive cone. If q_i lies in the positive cone of p_i, then we define $q_{i+1} = q_i$ and $p_{i+1} = r$ such that $(p_i,r) \in \mathcal{D}$, noting that by definition of $\mathcal{D}$, r is the unique such point in the positive cone of p_i that contains q_i. Otherwise, if p_i lies in a positive cone of q_i, then we define $p_{i+1} = p_i$ and $q_{i+1} = r'$ such that $(q_i,r') \in \mathcal{D}$. We stop when $p_k = q_k$.

We prove inductively that the aforementioned path π lies within $\bigtriangledown(p,q)$. For the base case, clearly the path consisting of the only edge in the sequence $\{p_{k-1},q_{k-1}\},\{p_k,q_k\}$ lies within $\bigtriangledown(p_{k-1},q_{k-1})$. For the inductive step, assuming that the path for the sequence $\{p_i,q_i\},\ldots,\{p_k,q_k\}$ lies within $\bigtriangledown(p_i,q_i)$, we show that the path for the sequence $\{p_{i-1},q_{i-1}\},\{p_i,q_i\},\ldots,\{p_k,q_k\}$ lies within $\bigtriangledown(p_{i-1},q_{i-1})$. First, we observe that in either case that the first edge is (p_{i-1},p_i) or (q_{i-1},q_i), it lies within $\bigtriangledown(p_{i-1},q_{i-1})$ by definition. Finally, we observe that $\bigtriangledown(p_i,q_i)$, hence the rest of the path lies within $\bigtriangledown(p_{i-1},q_{i-1})$ as well. Therefore, we prove the inductive step.

We then prove that all of the edges of the form (p_i,p_{i+1}) lie in the same corresponding positive cones of their respective points p_i. More specifically, we prove by induction on the sequence of such edges $e_0 = (p = p_{i_0-1},p_{i_0}), e_1 = (p_{i_0} = p_{i_1-1},p_{i_1}),\ldots,e_\ell = (p_{i_{\ell-1}} = p_{i_\ell-1},p_{i_\ell} = p_k)$, that $p_{i_0},p_{i_1},\ldots,p_{i_\ell}$ lie in the same corresponding cones of $p_{i_0-1},p_{i_1-1},\ldots,p_{i_\ell-1}$ respectively.

The base case follows trivially, and for the inductive step we assume for edges $e_0,e_1,\ldots,e_\lambda$ that $p_{i_0},p_{i_1},\ldots,p_{i_\lambda}$ lie in the same corresponding cones of $p_{i_0-1},p_{i_1-1},\ldots,p_{i_\lambda-1}$ respectively. For the inductive step we need to prove for the edge $e_{\lambda+1}$ that $p_{i_{\lambda+1}}$ lies in the same corresponding cone of $p_{i_{\lambda+1}-1} = p_{i_\lambda}$. We have already proven that the edge $e_{\lambda+1}$ lies within $\bigtriangledown(p_{i_\lambda},q_{i_\lambda})$. Also, by definition of $\mathcal{D}$, we know that $\bigtriangledown(p_{i_\lambda-1},p_{i_\lambda})$ is empty of points of $\mathcal{P}$ in its interior. Then we conclude the inductive proof by observing that the only positive cone that can possibly include $e_{\lambda+1}$ at p_{i_λ} is part of the same corresponding cone of p_{i_λ}. Having proven this critical property about this path, we denote it by π_p and refer to it as one of the two *branches*, where the other *branch* π_q is defined analogously using points $q_0,q_1,\ldots,q_k$. We conclude that the path π consisting of these two branches π_p and π_q is monotone.

Finally, we prove the claimed bound on the length of the path π' between p and q. Because the constants in Lemma 19 are the largest among Lemmas 13, 17, 18, and 19, the worst case happens when q lies in the white positive cone of p and π is white monotone. Letting z be the blue vertex of $\bigtriangledown(p,q)$, by Lemma 6, the projections of all edges of π onto zp (resp. zq) do not overlap, are contained within $[zp]$ (resp. zq), $|zp| = d_\bigtriangledown(p,q)$, and $|zq| = \delta^{blue}_\bigtriangledown(p,q)$. In the worst case, each edge of π is crossed, and Lemma 19 applies to reconstruct each edge (s,t) of π. Therefore, the length of the path π' and thus $d_\mathcal{S}(p,q)$ can be upper bounded by $\sum_{(s,t)\in\pi} 10d_\bigtriangledown(s,t) + 10\delta^{\min}_\bigtriangledown(s,t) \leq 10\sum_{(s,t)\in\pi} d_\bigtriangledown(s,t) + \delta^{blue}_\bigtriangledown(s,t)$. It follows that $d_\mathcal{S}(p,q) \leq 10(|zp| + |zq|)$ and therefore that $d_\mathcal{S}(p,q) \leq 20|pq|$ as desired. ◀

▶ **Theorem 21.** *$\mathcal{S}$ is a plane spanner of $\mathcal{C}$ with maximum degree at most 4 and stretch factor at most 20 and $\mathcal{S}$ can be constructed in $\mathcal{O}(n\log n)$ time.*

Proof. The planarity, maximum degree, and stretch factor properties of $\mathcal{S}$ were proven in Lemmas 11, 12, and 20, respectively. The TD-Delaunay triangulation $\mathcal{D}$ can be constructed in $\mathcal{O}(n\log n)$ time [11]. Given $\mathcal{D}$ and the fact that it is plane, $\mathcal{S}$ can be constructed in $\mathcal{O}(n\log n)$ time: sorting the anchors takes $\mathcal{O}(n\log n)$ time, and adding edges to $\mathcal{S}$ can be done in $\mathcal{O}(n)$ time. ◀

References

1. N. Bonichon, C. Gavoille, N. Hanusse, and D. Ilcinkas. Connections between theta-graphs, delaunay triangulations, and orthogonal surfaces. In *Proceedings of the 36th International Workshop on Graph Theoretic Concepts in Computer Science*, volume 6410 of *Lecture Notes in Computer Science*, pages 266–278, 2010.
2. N. Bonichon, C. Gavoille, N. Hanusse, and L. Perković. Plane spanners of maximum degree six. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6198 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 2010.
3. N. Bonichon, C. Gavoille, N. Hanusse, and L. Perković. The stretch factor of L_1- and L_∞-Delaunay triangulations. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA)*, volume 7501 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2012.
4. N. Bonichon, I. Kanj, L. Perkovic, and G. Xia. There are plane spanners of degree 4 and moderate stretch factor. *Discrete & Computational Geometry*, 53(3):514–546, 2015.
5. P. Bose, P. Carmi, and L. Chaitman-Yerushalmi. On bounded degree plane strong geometric spanners. *J. Discrete Algorithms*, 15:16–31, 2012.
6. P. Bose, P. Carmi, S. Collette, and M. Smid. On the stretch factor of convex delaunay graphs. *Journal of Computational Geometry*, 1(1):41–56, 2010.
7. P. Bose, J. Gudmundsson, and M. Smid. Constructing plane spanners of bounded degree and low weight. *Algorithmica*, 42(3-4):249–264, 2005.
8. P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.
9. P. Bose, M. Smid, and D. Xu. Delaunay and diamond triangulations contain spanners of bounded degree. *International Journal of Computational Geometry and Applications*, 19(2):119–140, 2009.
10. L. P. Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry (SoCG)*, pages 169–177, 1986.
11. L. P. Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.
12. G. Das and P.J. Heffernan. Constructing degree-3 spanners with other sparseness properties. *Int. J. Found. Comput. Sci.*, 7(2):121–136, 1996.
13. D. Dobkin, S. Friedman, and K. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5(4):399–407, December 1990. doi:10.1007/BF02187801.
14. I. Kanj and L. Perković. On geometric spanners of Euclidean and unit disk graphs. In *In Proceedings of the* 25^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume hal-00231084, pages 409–420. HAL, 2008.
15. I. Kanj, L. Perković, and D. Türkoğlu. Degree Four Plane Spanners: Simpler and Better. *CoRR*, abs/1603.03818, 2016. URL: http://arxiv.org/abs/1603.03818.
16. J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992.
17. X.-Y. Li and Y. Wang. Efficient construction of low weight bounded degree planar spanner. *International Journal of Computational Geometry and Applications*, 14(1-2):69–84, 2004.
18. J. Salowe. Euclidean spanner graphs with degree four. *Discrete Applied Mathematics*, 54(1):55–66, 1994.
19. Y. Wang and Xiang-Yang L. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications*, 11(2):161–175, 2006.
20. G. Xia. The stretch factor of the Delaunay triangulation is less than 1.998. *SIAM J. Comput.*, 42(4):1620–1659, 2013. doi:10.1137/110832458.

A Lower Bound on Opaque Sets*

Akitoshi Kawamura[1], Sonoko Moriyama[2], Yota Otachi[3], and János Pach[4]

1 **University of Tokyo, Japan**
kawamura@graco.c.u-tokyo.ac.jp
2 **Nihon University, Japan**
moriso@chs.nihon-u.ac.jp
3 **Japan Advanced Institute of Science and Technology, Japan**
otachi@jaist.ac.jp
4 **EPFL, Lausanne, Switzerland; and**
Rényi Institute, Budapest, Hungary
pach@cims.nyu.edu

Abstract

It is proved that the total length of any set of countably many rectifiable curves, whose union meets all straight lines that intersect the unit square U, is at least 2.00002. This is the first improvement on the lower bound of 2 by Jones in 1964. A similar bound is proved for all convex sets U other than a triangle.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases barriers, Cauchy–Crofton formula, lower bound, opaque sets

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.46

1 Introduction

A *barrier* or an *opaque set* for $U \subseteq \mathbb{R}^2$ is a set $B \subseteq \mathbb{R}^2$ that intersects every line that intersects U. For example, when U is a square, any of the four sets depicted in thick lines in Figure 1 is a barrier. The question of finding small barriers for polygons was first considered by Mazurkiewicz a century ago [12]. Note that some part of the barrier may lie outside U in our setting (Figure 2), and the barrier need not be connected.

We are interested in "short" barriers B for a given object U, and hence we restrict attention to those barriers that can be written as a union $B = \bigcup_{b\in\mathcal{B}} b$ of some countable set $\mathcal{B}$ of curves[1] b that each have finite length $|b|$ and the sum of these lengths $|\mathcal{B}| = \sum_{b\in\mathcal{B}}|b|$ is finite. We call such a set $\mathcal{B}$ (and not the union B, strictly speaking) a *rectifiable barrier*, and $|\mathcal{B}|$ its *length*.

Finding the shortest barrier is difficult, even for simple shapes U, such as the square, the equilateral triangle, and the disk [6, 10]. The shortest known barrier for the unit square is the rightmost one in Figure 1, with length 2.638.... This problem and its relatives have been considered by many authors. See [6, 11] and the introduction of [5] for more history, background, and related problems.

* The work presented here was supported in part by the Asahi Glass Foundation, by the ELC project (Grant-in-Aid for Scientific Research on Innovative Areas, MEXT, Japan), by OTKA under EUROGIGA projects GraDR and ComPoSe 10-EuroGIGA-OP-003, and by Swiss National Science Foundation Grants 200020-144531 and 200021-137574.

[1] In this paper, a *curve* or a *line segment* can by definition include both, one or none of the endpoints.

© Akitoshi Kawamura, Sonoko Moriyama, Yota Otachi, and János Pach;
licensed under Creative Commons License CC-BY

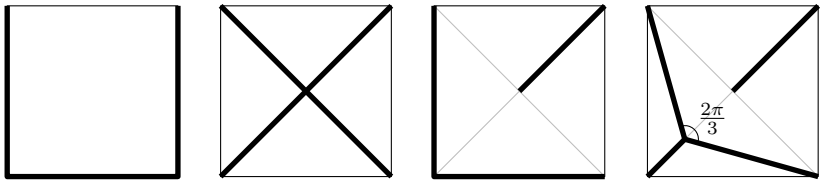


Figure 1 Barriers (in thick lines) for the unit square. The first one (three sides) and the second one (diagonals) have lengths 3 and $2\sqrt{2} = 2.828\ldots$, respectively. The third barrier consists of two sides and half of a diagonal, and has length $2 + 1/\sqrt{2} = 2.707\ldots$. The last one is the shortest known barrier for the unit square, with length $\sqrt{2} + \sqrt{6}/2 = 2.638\ldots$, consisting of half a diagonal and the Steiner tree of the lower left triangle.

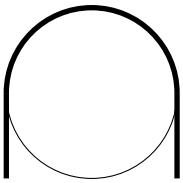


Figure 2 A barrier (in thick lines) for a disk that is shorter than the perimeter. This is not the shortest one; see [6].

The best known lower bound for the unit square has been 2, established by Jones in 1964 [9]. In general, for convex U, a barrier needs to have length at least half the perimeter of U (we review a proof in Section 2):

▶ **Lemma 1.** *$|\mathcal{B}| \geq p$ for any rectifiable barrier $\mathcal{B}$ of a convex set $U \subseteq \mathbb{R}^2$ with perimeter $2p$.*

Thus, from the point of view of finding short barriers, the trivial strategy of enclosing the entire perimeter (or the perimeter of the convex hull if U is a non-convex connected set) gives a 2-approximation. See [4] and references therein for algorithms that find shorter barriers. The current best approximation ratio is $1.58\ldots$ [5].

Proving a better lower bound has been elusive (again, even for specific shapes U). There has been some partial progress under additional assumptions about the shape (single arc, connected, etc.) and location (inside U, near U, etc.) of the barrier [1, 3, 7, 11, 14], but establishing an unconditional lower bound strictly greater than 2 for the unit square has been open (see [4, Open Problem 5] or [3, Footnote 1]). We prove such a lower bound in Section 4:

▶ **Theorem 2.** *$|\mathcal{B}| \geq 2.00002$ for any rectifiable barrier $\mathcal{B}$ of the unit square $\square$.*

Dumitrescu and Jiang [3] recently obtained a lower bound of $2 + 10^{-12}$ under the assumption that the barrier lies in the square obtained by magnifying $\square$ by 2 about its centre. Their proof, conceived independently of ours and at about the same time, is based on quite different ideas, most notably the line-sweeping technique. It will be worth exploring whether their techniques can be combined with ours.

Our proof can be generalized (Section 5):

▶ **Theorem 3.** *For any closed convex set U with perimeter $2p$ that is not a triangle, there is $\varepsilon > 0$ such that every rectifiable barrier of U has length at least $p + \varepsilon$.*

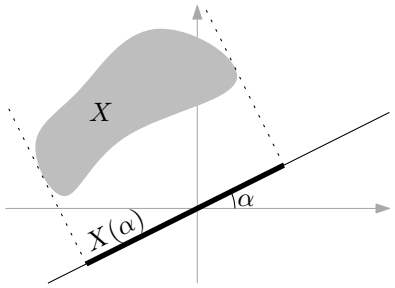


Figure 3 $X(\alpha)$ is the projection of X onto the angle-α (directed) line identified with $\mathbb{R}$.

Thus, the only convex objects for which we fail to establish a lower bound better than Lemma 1 are triangles[2].

The rest of this paper is structured as follows. In Section 2, we present a (known) proof for Lemma 1. We also prove that instead of rectifiable barriers, it is sufficient to restrict our attention to barriers comprised of line segments. In Section 3, we present three preliminary lemmas, analyzing some important special cases in which we can expect to improve on the bound from Lemma 1. The proof of one of these lemmas is postponed to Section 6. These lemmas are combined in Section 4 to obtain our lower bound for the length of a barrier for the square (Theorem 2). In Section 5, we show how to generalize these arguments to other convex sets (Theorem 3). In the last section, we discuss a closely related question.

2 Preliminaries: A general lower bound

For a set $X \subseteq \mathbb{R}^2$ and an angle $\alpha \in [0, 2\pi)$ (all angle calculation in this paper will be performed modulo 2π), we write

$$X(\alpha) = \{\, x\cos\alpha + y\sin\alpha : (x, y) \in X \,\} \tag{1}$$

for the projection of X at angle α (Figure 3). To say that a set $B \subseteq \mathbb{R}^2$ is a barrier of $U \subseteq \mathbb{R}^2$ means that $B(\alpha) \supseteq U(\alpha)$ for all α. We are interested in lower bounds on the length $|\mathcal{B}|$ of a rectifiable barrier $\mathcal{B}$ such that the union $B = \bigcup_{b\in\mathcal{B}} b$ satisfies this. For this purpose, it is no loss of generality to assume that $\mathcal{B}$ consists of line segments, as the following lemma shows. We call such $\mathcal{B}$ a *straight barrier*.

▶ **Lemma 4** ([5, Lemma 1]). *Let $\mathcal{B}$ be a rectifiable barrier for $U \subseteq \mathbb{R}^2$. Then, for any $\varepsilon > 0$, there exists a straight barrier $\mathcal{B}_\varepsilon$ for U such that $|B_\varepsilon| \le (1+\varepsilon)|B|$.*

Proof. Since the proof in [5] has a gap, we provide another proof. We will show that for any $\varepsilon > 0$ and any rectifiable curve b, there is a straight barrier $\mathcal{B}^b_\varepsilon$ of b of length $\le (1+\varepsilon)|b|$. This then gives a straight barrier $\mathcal{B}_\varepsilon = \bigcup_{b\in\mathcal{B}} \mathcal{B}^b_\varepsilon$ of U.

If b is already a line segment, we are done by setting $\mathcal{B}^b_\varepsilon = \{b\}$. Otherwise, the convex hull H of b has an interior point. Let b' be the curve obtained by magnifying b by $1+\varepsilon$ about this point. Since the convex hull of b' contains the compact set H in its interior, so does the convex hull of some finitely many points on b'. The set $\mathcal{B}^b_\varepsilon$ of line segments connecting these points along b' is a barrier of b of length at most $|b'| = (1+\varepsilon)|b|$. ◀

[2] During the preparation of this manuscript, Izumi [8] announced such a nontrivial lower bound for the equilateral triangle.

By Lemma 4, we may focus attention on straight barriers: U has a rectifiable barrier of length $< l$ if and only if it has a straight barrier of length $< l$.

As mentioned in the introduction (Lemma 1), it has been known that any barrier of a convex set must be at least half the perimeter. We include a short proof of this bound here, for completeness and further reference. See [2] for another elegant proof.

Proof of Lemma 1. By Lemma 4, we may assume that $\mathcal{B}$ is straight. We have

$$|U(\alpha)| \le \Big|\bigcup_{b\in\mathcal{B}} b(\alpha)\Big| \le \sum_{b\in\mathcal{B}} |b(\alpha)| = \sum_{b\in\mathcal{B}} |b| \cdot |\cos(\alpha-\theta_b)| \tag{2}$$

for each $\alpha \in [0, 2\pi)$, where θ_b is the angle of a line segment b. Integrating over $[0, 2\pi)$, we obtain

$$\int_{\alpha=0}^{2\pi} |U(\alpha)|\,\mathrm{d}\alpha \le \sum_{b\in\mathcal{B}} \left(|b| \cdot \int_{\alpha=0}^{2\pi} |\cos(\alpha-\theta_b)|\,\mathrm{d}\alpha \right) = 4\sum_{b\in\mathcal{B}} |b| = 4|\mathcal{B}|. \tag{3}$$

When U is a convex set, the left-hand side equals twice the perimeter (cf. the Cauchy–Crofton formula [13, Theorem 16.15]). ◀

3 Preliminary lemmas

Note that Theorems 2 and 3 do not merely state the non-existence of a straight barrier $\mathcal{B}$ of length exactly half the perimeter of U. Such a claim can be proved easily as follows: If $\mathcal{B}$ is such a barrier, the inequality (3) must hold with equality, and so must (2) for almost every α. Thus, the second inequality in (2) must hold with equality, which means that segments in $\mathcal{B}$ never overlaps one another when projected onto the line with angle α. Since this must be the case for almost every α, the entire $\mathcal{B}$ must lie on a line, which is clearly impossible.

The theorems claim more strongly that a barrier must be longer by an absolute constant. The following lemma says that in order to obtain such a bound, it suffices to find a part $\mathcal{B}' \subseteq \mathcal{B}$ of the barrier whose contribution to covering U is less than the optimal by at least a fixed positive constant (the proof is not hard and will be presented in the journal version).

▶ **Lemma 5.** *Let $\mathcal{B}$ be a rectifiable barrier of a closed convex set U of perimeter $2p$. Then $|\mathcal{B}| \ge p + \delta$ if there is a subset $\mathcal{B}' \subseteq \mathcal{B}$ with*

$$\int_{\alpha=0}^{2\pi} \big|\big(\bigcup_{b\in\mathcal{B}'} b\big)(\alpha) \cap U(\alpha)\big|\,\mathrm{d}\alpha \le 4|\mathcal{B}'| - 4\delta. \tag{4}$$

There are several ways in which such a "waste" can occur, and we make use of two of them (Figure 4). The first one is when there is a significant part of the barrier that lies far outside U, as described in the following lemma (the proof is again not hard and will be included in the journal version):

▶ **Lemma 6.** *Let b be a line segment that lies outside a convex region U. Suppose that the set $A := \{\,\alpha \in [0, 2\pi) : U(\alpha) \cap b(\alpha) \neq \emptyset\,\}$ (of angles of all lines through U and b) has measure $\le 2\pi - 4\varepsilon$. Then*

$$\int_{\alpha=0}^{2\pi} |b(\alpha) \cap U(\alpha)|\,\mathrm{d}\alpha \le 4|b|\cos\varepsilon. \tag{5}$$

The second situation where we have a significant waste required in Lemma 5 is when there are two sets of barrier segments that roughly face each other:

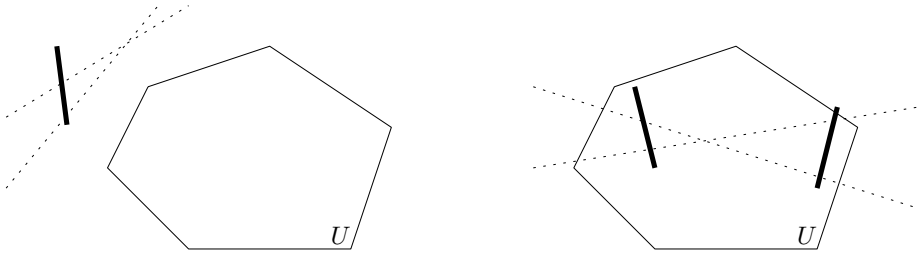


Figure 4 Two wasteful situations. In the left figure, a barrier segment (thick) lies far outside the object U, which leads to significant waste because this segment covers in vain some lines (dotted) that do not pass through U; this is discussed in Lemma 6. In the right figure, there are two parts of the barrier (thick) that face each other, which also results in significant waste because they cover some lines (dotted) doubly; this is roughly the situation discussed in Lemma 7.

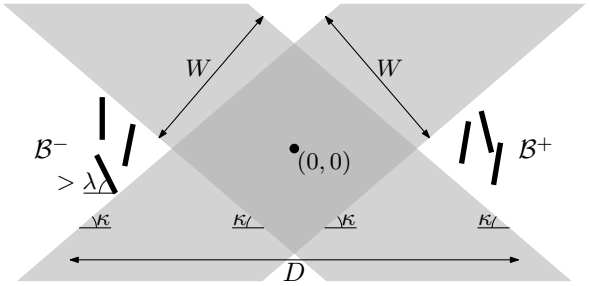


Figure 5 Sets $\mathcal{B}^-$ and $\mathcal{B}^+$ (Lemma 7).

▶ **Lemma 7.** *Let $\lambda \in (0, \frac{\pi}{2})$, $\kappa \in (0, \lambda)$ and l, $D > 0$. Let $\mathcal{B}^-$ and $\mathcal{B}^+$ be sets of n line segments of length l (Figure 5) such that*

1. *every segment of $\mathcal{B}^- \cup \mathcal{B}^+$ makes angle $> \lambda$ with the horizontal axis, and lies entirely in the disk of diameter D centred at the origin;*
2. *the segments in $\mathcal{B}^-$ and the segments in $\mathcal{B}^+$ are separated by bands of angle κ and width $W := nl\sin(\lambda - \kappa)$ centred at the origin, as depicted in Figure 5—that is, each point (x, y) on each segment in $\mathcal{B}^\pm$ satisfies $\pm(x\sin\kappa + y\cos\kappa) \geq W/2$ and $\pm(x\sin\kappa - y\cos\kappa) \geq W/2$ (where $\pm$ should be read consistently as $+$ and $-$).*

Then

$$\int_{\alpha=0}^{2\pi} \Big| \bigcup_{b \in \mathcal{B}^- \cup \mathcal{B}^+} b(\alpha) \Big| \, \mathrm{d}\alpha \leq 8nl - \frac{2W^2}{D}. \tag{6}$$

Note that $8nl = 4|\mathcal{B}^- \cup \mathcal{B}^+|$, so (6) is of the form (4) in Lemma 5.

The proof of Lemma 7 requires a more involved argument than Lemma 6, and will be given in Section 6. Before that, we prove Theorems 2 and 3 using Lemmas 6 and 7.

4 Proof of Theorem 2

We prove Theorem 2 using Lemmas 5, 6 and 7. The proof roughly goes as follows. Consider a barrier whose length is very close to 2.

1. There cannot be too much of the barrier far outside $\square$, because that would be too wasteful by Lemma 6.

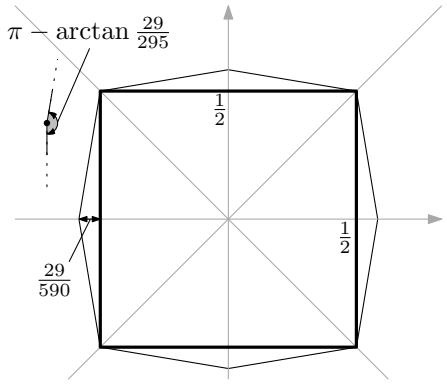


Figure 6 Viewed from any point outside the octagon ⯃, the square □ lies inside an angle that is smaller than π by the constant $\arctan\frac{29}{295}$.

2. This implies that there must be a significant part of the barrier near each corner of □, because this is the only place to put barrier segments that block those lines that clip this corner closely.
3. Among the parts of the barrier that lie near the four corners, there are parts that face each other and thus lead to waste by Lemma 7.

Proof of Theorem 2. Let □ be the unit square (including the boundary), which we assume to be axis-aligned and centred at the origin. Let $\mathcal{B}$ be a rectifiable barrier of □. By Lemma 4, we may assume that $\mathcal{B}$ is a straight barrier. Let ⯃ be the octagon (Figure 6) obtained by attaching to each edge of □ an isosceles triangle of height $\frac{29}{590}$ (and thus whose identical angles are $\arctan\frac{29}{295}$). By splitting some of the segments in $\mathcal{B}$ into several pieces, we may assume that $\mathcal{B} = \mathcal{B}_{\text{out}} \cup \mathcal{B}_{\text{in}}$, where each segment in $\mathcal{B}_{\text{in}}$ lies entirely in ⯃, and each segment in $\mathcal{B}_{\text{out}}$ lies entirely outside ⯃ and inside one of the eight regions delimited by the two axes and the two bisectors of the axes.

Suppose that $|\mathcal{B}_{\text{out}}| > \frac{1}{60}$. For each $b \in \mathcal{B}_{\text{out}}$, observe that, viewed from each point on b, the square □ lies entirely in an angle of size $\pi - \arctan\frac{29}{295}$ (Figure 6). This allows us to apply Lemma 6 and obtain

$$\int_{\alpha=0}^{2\pi} |b(\alpha) \cap \square(\alpha)| \, \mathrm{d}\alpha \leq 4|b| \cos\left(\frac{1}{2}\arctan\frac{29}{295}\right) < (4 - 0.0048)|b|. \tag{7}$$

Summing up for all $b \in \mathcal{B}_{\text{out}}$ (and using the triangle inequality), we have

$$\int_{\alpha=0}^{2\pi} \Big| \bigcup_{b\in\mathcal{B}_{\text{out}}} b(\alpha) \cap \square(\alpha) \Big| \, \mathrm{d}\alpha < (4 - 0.0048)|\mathcal{B}_{\text{out}}| \leq 4|\mathcal{B}_{\text{out}}| - 0.0048 \cdot \frac{1}{60}, \tag{8}$$

which yields $|\mathcal{B}| \geq 2.00002$ by Lemma 5. From now on, we can and will assume that $|\mathcal{B}_{\text{out}}| \leq \frac{1}{60}$.

Let $I_0 := \{\, (x, y) \in \mathbb{R}^2 : \frac{7}{8} \leq x + y \leq 1 \,\}$ and $R_0 := I_0 \cap$ ⯃ (Figure 7). Again, by splitting some of the segments in $\mathcal{B}$ into several pieces (which may or may not include endpoints), we may assume that each segment in $\mathcal{B}$ lies entirely in R_0 or entirely outside R_0. Let $\mathcal{B}_0 \subseteq \mathcal{B}$ consist of those that lie in R_0. Since $\bigcup_{b\in\mathcal{B}} b(\frac{\pi}{4}) \supseteq \square(\frac{\pi}{4}) = [-\sqrt{2}/2, \sqrt{2}/2] \supseteq [\frac{7}{8}\sqrt{2}/2, \sqrt{2}/2]$, and since the only segments $b \in \mathcal{B}$ for which $b(\frac{\pi}{4})$ can intersect this interval $[\frac{7}{8}\sqrt{2}/2, \sqrt{2}/2]$ of length $\sqrt{2}/16$ are those in $\mathcal{B}_0 \cup \mathcal{B}_{\text{out}}$, we have $|\mathcal{B}_0 \cup \mathcal{B}_{\text{out}}| \geq \sqrt{2}/16$, and hence

$$|\mathcal{B}_0| \geq \frac{\sqrt{2}}{16} - \frac{1}{60} > 0.07172 =: 2\eta. \tag{9}$$

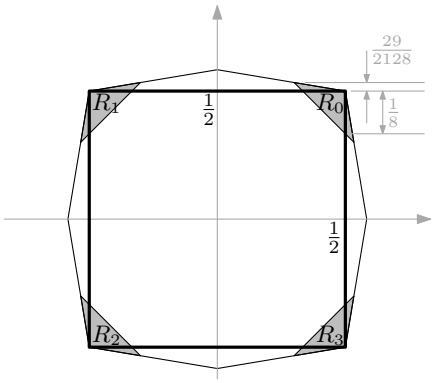


Figure 7 The regions R_0, R_1, R_2, R_3.

Likewise, let R_1, R_2, R_3 be the upper left, lower left, and lower right corners of $\bigcirc$, respectively, and define $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ analogously to $\mathcal{B}_0$, so that $|\mathcal{B}_1|$, $|\mathcal{B}_2|$, $|\mathcal{B}_3| > 2\eta$. Observe that the interval $R_0(\frac{\pi}{2} - 0.1813)$ lies above $R_1(\frac{\pi}{2} - 0.1813)$, with a gap of size

$$\frac{7}{8}\sin 0.1813 - \left(\frac{1}{8} + 2 \cdot \frac{29}{2128}\right)\cos 0.1813 > 0.008; \tag{10}$$

and $R_0(\frac{3\pi}{4} - 0.1813)$ lies above $R_2(\frac{3\pi}{4} - 0.1813)$, with an even bigger gap.

For each i, we partition $\mathcal{B}_i$ into three parts $\mathcal{B}_{i,i+1}$, $\mathcal{B}_{i,i+2}$, $\mathcal{B}_{i,i+3}$ (the subscripts are modulo 4), consisting respectively of segments whose angles are in $[\frac{\pi}{2}i - \frac{\pi}{4}, \frac{\pi}{2}i + \frac{\pi}{8})$, $[\frac{\pi}{2}i + \frac{\pi}{8}, \frac{\pi}{2}i + \frac{3\pi}{8})$ and $[\frac{\pi}{2}i + \frac{3\pi}{8}, \frac{\pi}{2}i + \frac{3\pi}{4})$. Thus, $\mathcal{B}_{i,j}$ consists of segments in $\mathcal{B}_i$ that "roughly point towards R_j." Since $|\mathcal{B}_i| > 2\eta$, we have $|\mathcal{B}_i \setminus \mathcal{B}_{i,j}| > \eta$ for at least two of the three j for each i, and thus, for at least eight of the twelve pairs (i, j). Hence, there is (i, j) such that $|\mathcal{B}_i \setminus \mathcal{B}_{i,j}| > \eta$ and $|\mathcal{B}_j \setminus \mathcal{B}_{j,i}| > \eta$.

Let $\mathcal{B}^-$ and $\mathcal{B}^+$ be finite sets of line segments of the same length such that $|\mathcal{B}^-| = |\mathcal{B}^+| = \eta$ and $\bigcup_{b\in\mathcal{B}^-} b \subseteq \bigcup_{b\in\mathcal{B}_i\setminus\mathcal{B}_{i,j}} b$, $\bigcup_{b\in\mathcal{B}^+} b \subseteq \bigcup_{b\in\mathcal{B}_j\setminus\mathcal{B}_{j,i}} b$. Apply Lemma 7 to these $\mathcal{B}^-$ and $\mathcal{B}^+$, rotated and translated appropriately, and the constants $\kappa = 0.1813$, $\lambda = \frac{\pi}{8}$, $D = \sqrt{2}$. Note that the last assumption of Lemma 7 is satisfied because $W := \eta\sin(\lambda - \kappa) = 0.03586\sin(\frac{\pi}{8} - 0.1813) = 0.007524\ldots < 0.008$. This gives

$$\int_{\alpha=0}^{2\pi} \left| \bigcup_{b\in\mathcal{B}^-\cup\mathcal{B}^+} b(\alpha) \right| \mathrm{d}\alpha \le 8\eta - \frac{2W^2}{D} < 8\eta - 0.00008, \tag{11}$$

whence $|\mathcal{B}| \ge 2.00002$ by Lemma 5. ◀

We did not attempt to seriously optimize the specific numbers in the above proof (such as $\frac{29}{590}$, $\frac{1}{60}$, $\frac{1}{8}$, ...), but they are somewhat carefully chosen, for the following intuition. What we needed to do is to cut out the four small regions R_i from the corners, and argue that some significant part of the barrier segments in these regions are in the relative position described in the assumption of Lemma 7 (Figure 5). If these regions are too small, we would obtain only a small amount of such segments (i.e., η would be smaller). If they are too big, we would know less about the relative position of the segments (and thus have to use worse values of λ and κ). The number $\frac{1}{8}$ for the size of the corners R_i was (roughly) chosen for the right balance between these factors. The number $\frac{29}{590}$ was then chosen big enough to imply (via Lemma 6) that $|\mathcal{B}_{\mathrm{out}}|$ is quite small (specifically $\frac{1}{60}$, which is so small that η defined in (9) is still a significant positive value), but not so big that the regions R_i stretch too much and deteriorate κ.

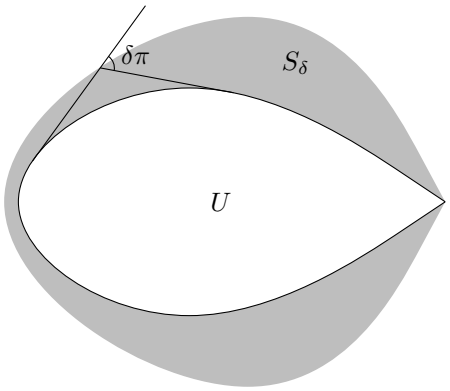


Figure 8 S_δ is the set of points from which U looks big. Putting too much of the barrier outside S_δ is wasteful.

5 Proof of Theorem 3

Theorem 3 is proved by modifying the proof of Theorem 2 (Section 4) as follows. Let x_i be distinct points ($i = 1, 2, 3, 4$) on the boundary of U at which U is strictly convex, i.e., there is a line that intersects U only at x_i; let α_i be the angle of this line. Note that such four points exist unless U is a triangle. Let R_i be a sufficiently small closed neighbourhood of x_i, so that no three of R_1, R_2, R_3, R_4 are stabbed by a line.

Instead of the octagon $\bigcirc$, we consider the set $S_\delta \supseteq U$ of points such that a random line through this point avoids U with probability less than a positive constant δ (Figure 8). By applying Lemma 6 in the same way (with some routine compactness argument), we know that $\mathcal{B}_{\text{out}}$ (the segments in the assumed straight barrier $\mathcal{B}$ that lie outside S_δ) must be small (under the assumption of $|\mathcal{B}| \le p + \varepsilon$, for an appropriately small ε). By taking δ sufficiently small, S_δ comes so close to U that the following happens for each $i = 1, 2, 3, 4$: there is a neighbourhood $N \subseteq U$ of x_i in U such that every angle-α_i line that intersects N intersects S_δ only in R_i. This guarantees that the part $\mathcal{B}_i$ of $\mathcal{B}$ that lies in R_i must have length at least some positive constant (just to block those angle-α_i lines that hit N). This allows us to define $\mathcal{B}_{i,j}$ in the way similar to Theorem 2 and apply Lemma 7 with appropriate κ, λ, D.

6 Proof of Lemma 7

It remains to prove Lemma 7. Let us first interpret what it roughly claims. By symmetry, we can halve the interval $[0, 2\pi]$ and replace (6) by

$$4nl - \int_{\alpha=0}^{\pi} \Big| \bigcup_{b \in \mathcal{B}^- \cup \mathcal{B}^+} b(\alpha) \Big| \, \mathrm{d}\alpha \ge \frac{W^2}{D}. \tag{12}$$

For each $b \in \mathcal{B}^- \cup \mathcal{B}^+$, consider the region

$$R_b := \{\, (\alpha, v) \in [0, \pi] \times \mathbb{R} : v \in b(\alpha) \,\}, \tag{13}$$

whose area is $2l$. Note that the first term $4nl$ of (12) is the sum of this area for all $b \in \mathcal{B}^- \cup \mathcal{B}^+$, whereas the second term is the area of the union. Thus, (12) says that the area of the overlap (considering multiplicity) is at least W^2/D. To prove such a bound, we start with the following lemma, which provides a similar estimate on the size of potentially complicated overlaps, but of simpler objects, namely bands with fixed width.

▶ **Lemma 8.** *Let $I \subseteq \mathbb{R}$ be an interval and let W, $D \geq 0$. Let $\mathcal{U}$ be the set of functions f which take each $\alpha \in I$ to an interval $f(\alpha) = [\underline{f}(\alpha), \overline{f}(\alpha)]$ of length W/n and are $\frac{1}{2}D$-Lipschitz, that is, $|\underline{f}(\alpha_0) - \underline{f}(\alpha_1)| \leq \frac{1}{2}D \cdot |\alpha_0 - \alpha_1|$ for each α_0, $\alpha_1 \in I$. Suppose that $2n$ functions f_1, $\dots$, f_n, g_1, $\dots$, $g_n \in \mathcal{U}$ satisfy*

$$\underline{g_j}(\min I) - \overline{f_i}(\min I) \geq W, \qquad \underline{f_i}(\max I) - \overline{g_j}(\max I) \geq W \tag{14}$$

for each i, j (i.e., the functions f_i start far below g_j and end up far above). Then

$$\left|R_{f_1} \cup \cdots \cup R_{f_n} \cup R_{g_1} \cup \cdots \cup R_{g_n}\right| \leq 2W|I| - \frac{W^2}{D}, \tag{15}$$

where $R_f := \{\, (\alpha, v) \in I \times \mathbb{R} : v \in f(\alpha) \,\}$ denotes the graph of $f \in \mathcal{U}$.

The proof will be given in the full version.

Proof of Lemma 7. Let R_b be as in (13). As explained there, our goal is to prove (12), which says that the area of the overlap among R_b for $b \in \mathcal{B}^- \cup \mathcal{B}^+$ is at least W^2/D. We claim that this is true *even if we replace each R_b by its subset $\tilde{R}_b \subseteq R_b$ defined below.*

Let $I := [\frac{\pi}{2} - \kappa, \frac{\pi}{2} + \kappa]$. Note that, because of the configuration of segments (Figure 5), we have $|b(\alpha)| \geq l \sin(\lambda - \kappa) = W/n$ for each $\alpha \in I$ and $b \in \mathcal{B}^- \cup \mathcal{B}^+$. We define $\tilde{R}_b$ from R_b by restricting α to I and replacing the interval $b(\alpha)$ by its subinterval $\tilde{b}(\alpha) := [\min(b(\alpha)), \min(b(\alpha)) + W/n]$. Thus,

$$\tilde{R}_b := \{\, (\alpha, v) \in I \times \mathbb{R} : v \in \tilde{b}(\alpha) \,\}. \tag{16}$$

Note that these functions $\tilde{b}$ are $\frac{1}{2}D$-Lipschitz (see Lemma 8), because the segments $b \in \mathcal{B}^- \cup \mathcal{B}^+$ lie within distance $\frac{1}{2}D$ of the origin. We can thus apply Lemma 8 to $\{f_1, \dots, f_n\} := \{\, \tilde{b} : b \in \mathcal{B}^- \,\}$ and $\{g_1, \dots, g_n\} := \{\, \tilde{b} : b \in \mathcal{B}^+ \,\}$, since (14) is satisfied because of the width-W separation (Figure 5) assumed in Lemma 7. We can thus apply Lemma 8 and obtain $\left|\bigcup_{b \in \mathcal{B}^- \cup \mathcal{B}^+} \tilde{R}_b\right| \leq 2W|I| - W^2/D$, as was desired. ◀

7 Half-line barriers

We propose an analogous question, obtained by replacing lines by half-lines in the definition of barriers: a set $B \subseteq \mathbb{R}^2$ is a *half-line barrier* of $U \subseteq \mathbb{R}^2$ if all half-lines intersecting U intersect B. This intuitively means "hiding the object U from outside," which we find perhaps as natural, if not more, than the notion of opaque sets. Similarly to Lemma 1, we have

▶ **Lemma 9.** *$|B| \geq p$ for any rectifiable half-line barrier B of a convex set $U \subseteq \mathbb{R}^2$ that is not a line segment and has perimeter p.*

Thus, unlike for line barriers, the question is completely answered when U is connected: the shortest half-line barrier is the boundary of the convex hull.

If U is disconnected, there can be shorter half-line barriers. For example, if U consists of two connected components that are enough far apart from each other, it is more efficient to cover them separately than together. One might hope that an optimal half-line barrier is always obtained by grouping the connected components of U in some way and taking convex hulls of each. This is not true, as the example in Figure 9 shows. We have not been able to find an algorithm that achieves a nontrivial approximation ratio for this problem.

Acknowledgements. We are grateful to Gábor Tardos for many interesting discussions on the subject. In particular, the present proof of Lemma 4 is based on his idea.

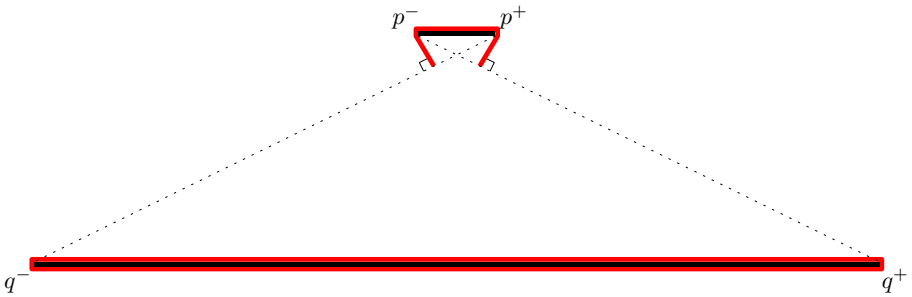


Figure 9 Consider the line segments p^-p^+ and q^-q^+, where $p^\pm = (\pm 1, 8)$ and $q^\pm = (\pm 15, 0)$, and let U be the union of these segments with small "thickness": U consists of a rectangle with vertices $(\pm 1, 8 \pm \varepsilon)$ and another with vertices $(\pm 15, \pm\varepsilon)$, for a small $\varepsilon > 0$. The boundaries of these thick line segments have total length 64 (plus a small amount due to the thickness). The boundary of the convex hull of all of U has length $2 + 30 + 2\sqrt{260} > 64.24$ (plus thickness). But we have another half-line barrier depicted above, whose total length is $2 + 60 + 2/\sqrt{5} + 2/\sqrt{5} < 63.79$ (plus thickness, which can be made arbitrarily small).

References

1. H. T. Croft. Curves intersecting certain sets of great-circles on the sphere. *Journal of the London Mathematical Society (2)*, 1, 461–469, 1969.
2. E. Demaine and J. O'Rourke. Open problems from CCCG 2007. In *Proc. 20th Canadian Conference on Computational Geometry* (CCCG 2008), 183–186, 2008.
3. A. Dumitrescu and M. Jiang. The opaque square. In *Proc. 30th Annual Symposium on Computational Geometry* (SoCG 2014), 529–538, 2014.
4. A. Dumitrescu and M. Jiang. Computational Geometry Column 58. *SIGACT News*, 44(4), 73–78, 2013.
5. A. Dumitrescu, M. Jiang, and J. Pach. Opaque sets. *Algorithmica*, 69(2), 315–334, 2014.
6. V. Faber and J. Mycielski. The shortest curve that meets all the lines that meet a convex body. *American Mathematical Monthly*, 93, 796–801, 1986.
7. V. Faber, J. Mycielski, and P. Pedersen. On the shortest curve which meets all the lines which meet a circle. *Annales Polonici Mathematici*, 44, 249–266, 1984.
8. T. Izumi. Improving lower bound on opaque set for equilateral triangle. Preprint, arXiv:1509.03846.
9. R. E. D. Jones. Opaque sets of degree α. *American Mathematical Monthly*, 71, 535–537, 1964.
10. B. Kawohl. The opaque square and the opaque circle. *General Inequalities 7*, ISNM International Series of Numerical Mathematics Volume 123, 339–346, 1997.
11. B. Kawohl. Some nonconvex shape optimization problems. In *Optimal Shape Design*, 7–46, Springer, 2000.
12. S. Mazurkiewicz. Przykład zbioru domkniętego, punktokształtnego, mającego punkty wspólne z każdą prostą, przecinającą pewien obszar domknięty. (Sur un ensemble fermé, punctiforme, qui rencontre toute droite passant par un certain domaine.) *Prace Matematyczno-Fizyczne*, 27, 11–16, 1916. In Polish (French summary).
13. J. Pach and P. K. Agarwal. *Combinatorial Geometry*. Wiley, New York, 1995.
14. J. S. Provan, M. Brazil, D. Thomas, J. F. Weng. Minimum opaque covers for polygonal regions. Preprint, arXiv:1210.8139v1.

Fixed Points of the Restricted Delaunay Triangulation Operator

Marc Khoury[1] **and Jonathan Richard Shewchuk**[2]

1 University of California, Berkeley, USA
2 University of California, Berkeley, USA

Abstract

The restricted Delaunay triangulation can be conceived as an operator that takes as input a k-manifold (typically smooth) embedded in $\mathbb{R}^d$ and a set of points sampled with sufficient density on that manifold, and produces as output a k-dimensional triangulation of the manifold, the input points serving as its vertices. What happens if we feed that triangulation back into the operator, replacing the original manifold, while retaining the same set of input points? If $k = 2$ and the sample points are sufficiently dense, we obtain another triangulation of the manifold. Iterating this process, we soon reach an iteration for which the input and output triangulations are the same. We call this triangulation a *fixed point* of the restricted Delaunay triangulation operator.

With this observation, and a new test for distinguishing "critical points" near the manifold from those near its medial axis, we develop a provably good surface reconstruction algorithm for $\mathbb{R}^3$ with unusually modest sampling requirements. We develop a similar algorithm for constructing a simplicial complex that models a 2-manifold embedded in a high-dimensional space $\mathbb{R}^d$, also with modest sampling requirements (especially compared to algorithms that depend on sliver exudation). The latter algorithm builds a non-manifold representation similar to the flow complex, but made solely of Delaunay simplices. The algorithm avoids the curse of dimensionality: its running time is polynomial, not exponential, in d.

1998 ACM Subject Classification F.2.2 Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems

Keywords and phrases restricted Delaunay triangulation, fixed point, manifold reconstruction, surface reconstruction, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.47

1 Introduction

Manifold reconstruction is the problem of discovering the structure of a k-dimensional manifold embedded in $\mathbb{R}^d$, given only a set of points sampled from the manifold. The classic application is *surface reconstruction*, the problem of discovering a surface (2-manifold) in three-dimensional space. Manifold reconstruction also has applications in higher-dimensional spaces, such as discovering relationships in data and studying algebraic varieties.

A large vein of research in surface and manifold reconstruction develops algorithms that are *provably good* [1, 2, 13, 5]: if the points sampled from a manifold are sufficiently dense, these algorithms are guaranteed to produce a geometrically accurate representation of the unknown manifold with the correct topology. Most of these algorithms are based on Delaunay triangulations, and their output is a set of Delaunay simplices that approximates the unknown manifold. In particular, they generate *restricted Delaunay triangulations*. In this paper, we study restricted Delaunay triangulations that model 2-manifolds embedded in ambient spaces of three or more dimensions. Our algorithm's guarantees hold for relatively sparse point samples, especially in four or more dimensions.


© Marc Khoury and Jonathan Richard Shewchuk;
licensed under Creative Commons License CC-BY

The restricted Delaunay triangulation (RDT) is a geometric structure that formally extends the Delaunay triangulation to curved manifolds. Think of the RDT as a mathematical operator that takes two inputs—a manifold and a finite set of points that lie on the manifold—and produces as output a triangulation of the manifold whose vertices are the input points. This result is conditional: if the points are not sampled densely enough, the RDT will not be a triangulation of the manifold. The RDT is (unconditionally) a subcomplex of the full-dimensional Delaunay triangulation in $\mathbb{R}^d$. It has proven itself as a mathematically powerful tool for surface meshing and surface reconstruction.

What happens if we feed an RDT of a 2-manifold back into the operator, replacing the original manifold, while retaining the same set of input points? If the sample points are sufficiently dense, and the triangulation is a sufficiently good approximation of the manifold from which they were sampled, we obtain another triangulation of the manifold. We prove that if we iterate this process, we eventually reach an iteration for which the input triangulation and output triangulation are the same. We call this triangulation a *fixed point* of the restricted Delaunay triangulation operator for the specified point set. Such *fixed-point triangulations* appear to be particularly accurate representations of the unknown manifold.

Fixed-point triangulations bifurcate into those that stay everywhere close to the unknown manifold and those that somewhere come very close to its medial axis. We call the set of simplices that can appear in the former the *manifold fixed-point complex* (MF complex), and we discuss how to compute it. The MF complex is a fixed point of the RDT operator; moreover, it is the union of all the fixed points that stay far from the medial axis.

The MF complex closely resembles the well-known *flow complex* [17], but it better approximates (in Hausdorff distance and tangents) the unknown manifold. Unlike the flow complex, the MF complex is a subcomplex of the Delaunay triangulation, so it offers greater simplicity. In three dimensions, it is easy to extract a subcomplex of the MF complex that is itself a manifold, homeomorphic to the unknown manifold. In higher dimensions, we prove that such a subcomplex exists. We believe that heuristic algorithms will have little difficulty extracting a homeomorphic manifold, but the problem seems difficult in theory.

Manifold reconstruction algorithms in high-dimensional ambient spaces face the *curse of dimensionality*: a Delaunay triangulation of n vertices in the ambient space $\mathbb{R}^d$ can have up to $\Theta(n^{\lceil d/2 \rceil})$ d-simplices. Even if the complexity is close to linear, as often happens, it is not feasible to triangulate large point clouds of dimension much higher than 10. Our algorithm's running time is polynomial in d, thanks to a method for computing low-dimensional cross-sections of high-dimensional Voronoi diagrams that we borrow from Flötotto [16].

An important ingredient for computing either the MF complex or the flow complex is a test that examines "critical points" (where Delaunay simplices intersect their Voronoi duals) and distinguishes those near the unknown manifold from those near its medial axis. We propose a test that is correct for sparser samples than the test proposed by Dey, Giesen, Ramos, and Sadri [12] in their paper on computing flow complexes. More importantly, we extend it to higher dimensions, without constructing the Delaunay triangulation in $\mathbb{R}^d$. Our new test can be used to construct flow complexes as well.

A benefit of our approach is that our analysis guarantees good reconstructions for coarser point samples than traditional Delaunay reconstruction algorithms such as the Crust [1] and Cocone [2, 13] algorithms or the Dey et al. flow complex algorithm. In $\mathbb{R}^3$, we compute a homeomorphic reconstruction from a 0.143-sample of a surface (see the definition in Section 2); by comparison, the analysis of the Cocone algorithm guarantees a homeomorphic reconstruction only for a 0.05-sample. We attribute the improved constant to our new critical point classifier, a new method for proving homeomorphisms, and the exploitation of

critical points (like the flow complex, but with Delaunay triangles). In higher dimensions, a 0.143-sample also suffices, though we do not guarantee a homeomorphic reconstruction. This coarse sampling requirement sharply distinguishes our algorithm from previous algorithms for manifold reconstruction in codimension two or higher [5, 9]. (See Section 3).

In $\mathbb{R}^3$, our algorithm produces triangles that are similar but not identical to those produced by Edelsbrunner's Wrap algorithm [14] (which is based on flows); and similar but not identical to those used by Dey et al. to help construct the flow complex. The differences are subtle but theoretically important, as they permit us to benefit from the forthcoming Lemma 5.

Unfortunately, our algorithm reconstructs only 2-manifolds. The problem is that the Delaunay 3-simplices used to triangulate a 3-manifold can have extremely unstable affine hulls and normals; they can be perpendicular to the original manifold, and their normals parallel. If we feed such a defective triangulation back into the RDT operator, it will likely produce a triangulation with nasty folds and self-intersections, not homeomorphic to Σ. Nevertheless, we suspect that RDTs will someday be adapted to reconstructing general k-manifolds. For that reason, we discuss some parts of our algorithm in a general-dimensional framework.

2 The Restricted Delaunay Triangulation

Let Σ be a k-manifold embedded in $\mathbb{R}^d$. Let $V \subset \Sigma$ (for "Vertices") be a finite set of n points sampled from Σ. Let $\operatorname{Del} V$ and $\operatorname{Vor} V$ denote, respectively, the d-dimensional Delaunay triangulation and Voronoi diagram of V. Our goal is to find a subset of the k-faces in $\operatorname{Del} V$ that forms a triangulation that geometrically approximates Σ and is topologically equivalent to Σ. For $k = 2$, this is always possible if Σ is sufficiently smooth and V is sufficiently dense.

Throughout this paper, we use pq to denote the line segment connecting p to q, and $|pq|$ to denote its length; i.e., the distance from p to q. The *codimension* of Σ is $d-k$. We presume familiarity with the duality between Delaunay triangulations and Voronoi diagrams: every simplex τ of dimension j in $\operatorname{Del} V$ has a dual face of dimension $d-j$ in $\operatorname{Vor} V$, denoted τ^*. In particular, the Voronoi cell of a site $u \in V$ is denoted $u^* = \{p \in \mathbb{R}^d : \forall w \in V, |up| \leq |wp|\}$.

The *restricted Voronoi diagram* of V with respect to Σ, denoted $\operatorname{Vor}|_\Sigma V$, is a cell complex much like $\operatorname{Vor} V$, but the Voronoi cells contain only points on the manifold Σ. Thus $\operatorname{Vor}|_\Sigma V = \{c \cap \Sigma : c \in \operatorname{Vor} V\}$, where c ranges over all the cells of all dimensions in the Voronoi diagram. In other words, we replace each Voronoi face c with its *restriction* $c|_\Sigma = c \cap \Sigma$. For each site $u \in V$, $\operatorname{Vor}|_\Sigma V$ contains the *restricted Voronoi cell* of u, namely, $u^*|_\Sigma = \{p \in \Sigma : \forall w \in V, |up| \leq |wp|\}$.

RDTs are defined *not* by restricting Delaunay simplices to a manifold, but by dualizing the restricted Voronoi diagram. The *restricted Delaunay triangulation* of V with respect to Σ, denoted $\operatorname{Del}|_\Sigma V$, is the subcomplex of $\operatorname{Del} V$ that contains every simplex in $\operatorname{Del} V$ whose Voronoi dual face intersects Σ. (I.e., $\operatorname{Del}|_\Sigma V$ contains every simplex dual to a face of $\operatorname{Vor} V$ that contributes a nonempty face to $\operatorname{Vor}|_\Sigma V$.)

We find it useful to define $\operatorname{Del}|_P V$ not just for manifolds, but for any arbitrary point set $P \subseteq \mathbb{R}^d$. Although the RDT operator is usually applied to smooth manifolds, we will apply it to piecewise linear manifolds and even individual simplices.

Let us characterize the restricted Delaunay simplices. A *circumsphere* of a j-simplex is any hypersphere in $\mathbb{R}^d$ that passes through all $j+1$ vertices of the simplex. A d-simplex has one unique circumsphere, whereas a lower-dimensional simplex has infinitely many. The *diametric sphere* of a simplex τ is the unique circumsphere with least radius. The center of τ's diametric sphere, called the *circumcenter* of τ, lies on the affine hull of τ. The *affine circumsphere* of τ is the unique $(j-1)$-sphere that passes through all $j+1$ vertices; it lies on τ's affine hull and it is a cross-section of every circumsphere of τ. The *circumradius* of τ is the radius of its affine circumsphere, which is also the radius of its diametric sphere.

A circumsphere S is *empty* if no vertex in V is inside S. (Vertices precisely on S don't count.) Simplices in an ordinary Delaunay triangulation $\operatorname{Del} V$ are characterized by the well-known *empty circumsphere condition*: τ is *Delaunay* if its vertices are in V and it has at least one empty circumsphere. For τ to be *restricted Delaunay*, it must satisfy these conditions plus one more: it has at least one empty circumsphere whose center lies on the manifold Σ.

If Σ is a k-manifold, $\operatorname{Del}|_\Sigma V$ typically contains only simplices of dimensions k and lower. If simplices of higher dimension appear, we perturb Σ infinitesimally to make them go away. For any cell complex $\mathcal{T}$, the *underlying space* $|\mathcal{T}|$ of $\mathcal{T}$ is the union of simplices in $\mathcal{T}$; i.e., $|\mathcal{T}| = \bigcup_{\tau\in\mathcal{T}} \tau$. (The operator $|\cdot|$ collapses a set of sets of points into a set of points.) Under favorable conditions, $|\operatorname{Del}|_\Sigma V|$ can be a k-manifold and a topologically and geometrically good approximation of Σ.

RDTs have become a standard tool in provably good surface reconstruction algorithms such as Crust [1] and Cocone [2, 13] and in many guaranteed-quality surface meshing algorithms [6, 10, 15]. In the mesh generation problem, Σ is known and we choose the mesh vertices $V \in \Sigma$ so that $\operatorname{Del}|_\Sigma V$ is a high-quality mesh. In the manifold reconstruction problem, V is known but Σ is not, and we try to infer Σ from V by guessing the subcomplex of $\operatorname{Del} V$ most likely to be $\operatorname{Del}|_\Sigma V$ or a good approximation thereof. Without knowing Σ, however, we cannot necessarily determine $\operatorname{Del}|_\Sigma V$, even if V is sampled extremely densely.

A major research victory in surface reconstruction was the characterization of how dense a point sample $V \subset \Sigma$ needs to be to guarantee a correct reconstruction, taking into account that the requirements vary through space. Let M be Σ's medial axis. The *local feature size* function is $\operatorname{lfs} : \Sigma \to \mathbb{R}, p \mapsto d(p, M)$, where $d(p, M)$ denotes the distance from p to M. We require that Σ is smooth enough that $\inf_{p\in\Sigma} \operatorname{lfs}(p) > 0$. A finite point set $V \subset \Sigma$ is an *ϵ-sample* of Σ if for every point $p \in \Sigma$, $d(p, V) \le \epsilon \operatorname{lfs}(p)$. That is, there is some sample $v \in V$ in the ball of radius $\epsilon \operatorname{lfs}(p)$ centered at p. The function $\operatorname{lfs}(\cdot)$ has proven itself as an appropriate, space-varying measure of the sampling density necessary so that algorithms for surface reconstruction and surface meshing can guarantee the correctness of their outputs.

For a 2-manifold $\Sigma \subset \mathbb{R}^d$ and a 0.2695-sample $V \subset \Sigma$, we can show that $|\operatorname{Del}|_\Sigma V|$ is homeomorphic to Σ. (See the full-length version of this paper.) There is a homeomorphism that doesn't move any point too far, so the Hausdorff distance between Σ and $|\operatorname{Del}|_\Sigma V|$ is small. The triangles in $\operatorname{Del}|_\Sigma V$ are approximately parallel to the surface Σ nearby.

To beat the curse of dimensionality, we borrow from Flötotto [16] and Boissonnat and Ghosh [5] the idea of the *tangential Delaunay complex*, which is simply the restricted Delaunay triangulation $\operatorname{Del}|_\Pi V$ where Π is a k-flat (i.e., a k-dimensional affine subspace). Whereas a d-dimensional Voronoi diagram $\operatorname{Vor} V$ with n vertices may have complexity as high as $\Theta(n^{\lceil d/2\rceil})$, a k-dimensional cross-section of $\operatorname{Vor} V$ has complexity only $O(n^{\lceil k/2\rceil})$ and can be computed much faster. The trick is to observe that a k-dimensional cross-section $\operatorname{Vor}|_\Pi V$ of $\operatorname{Vor} V$ is a *power diagram*. If we project each point $v \in V$ orthogonally onto Π and assign each projected point a weight of $-d(v, \Pi)^2$, where $d(v, \Pi)$ is the distance from v to Π, then the k-dimensional power diagram of the weighted, projected points is the desired cross-section of the d-dimensional Voronoi diagram. The dual of $\operatorname{Vor}|_\Pi V$ is a *weighted Delaunay triangulation* $\operatorname{Del}|_\Pi V$ and can be computed by standard algorithms for computing $(k+1)$-dimensional convex hulls such as the randomized incremental algorithm of Clarkson and Shor [11], which takes expected $O(n^{\lceil k/2\rceil} + n \log n)$ time.

For more details about Delaunay triangulations, restricted Delaunay triangulations, medial axes, the local feature size, and ϵ-samples, see Dey [13] or Cheng et al. [10].

3 Slivers and Ambiguities in Manifold Reconstruction

All Delaunay-based surface reconstruction algorithms face a problem that doesn't sound like it should be a problem: $\mathrm{Del}\, V$ typically contains many different subcomplexes that would serve as good triangulations of Σ. Even in three dimensions, four nearly-cocircular vertices on Σ may form a flat, kite-shaped tetrahedron called a *sliver*, which is usually aligned roughly parallel to the surface. Slivers create ambiguities for surface reconstruction, because the top two faces or the bottom two faces of a sliver might equally well represent the surface. The choice might not matter, but to obtain a manifold a choice must be made.

Extracting a manifold from $\mathrm{Del}\, V$ typically requires many such choices. Unfortunately, these choices are not independent. Ultra-thin slivers can stack up in interlocking layers. In higher dimensions, slivers and their higher-dimensional analogs layer along multiple axes.

Sections 4 through 6 analyze the effects of taking the output of the RDT operator and feeding it back into the RDT operator. Slivers are the reason why the RDTs produced by successive iterations may differ from each other. We will see that successive iterates tend to be better than their predecessors until they arrive at a fixed point. However, fixed points are not unique. For example, if a sliver in $\mathbb{R}^3$ contains its own circumcenter, there may be one fixed-point triangulation that chooses the sliver's bottom two faces, and another that chooses the top two. Making the point sample V very dense does nothing to prevent this.

Section 8 presents an algorithm that computes a manifold fixed-point complex (MF complex) that is the union of all fixed-point triangulations that don't get close to the medial axis. The MF complex might contain all four faces of a sliver that contains its own circumcenter (or, if $d > 3$, intersects its own Voronoi dual face). However, typically only a minority of slivers contain their own circumcenters. (The circumcenter of a sliver is very unstable: a small perturbation of a vertex can shoot the circumcenter far away.) The MF complex accepts fewer triangles than algorithms like Crust and Cocone, resolving most, but not all, of the ambiguities, and also obtaining higher quality.

In $\mathbb{R}^3$, there is a fast, simple method for resolving the remaining ambiguities and extracting a manifold, described in Section 8, but it relies on the fact that a codimension-one manifold ($d - k = 1$) partitions space into "inside" and "outside"; it finds the outermost boundary.

For codimension two or higher, Cheng, Dey, and Ramos [9] and Boissonnat and Ghosh [5] solve this problem by replacing $\mathrm{Del}\, V$ with a weighted Delaunay triangulation and using a technique called *sliver exudation* [8] to select vertex weights so that the slivers disappear and the triangulation has only one subcomplex that approximates Σ. At least in theory, sliver exudation also has the virtue that it enables the reconstruction of manifolds of dimension greater than 2, unlike our method. Unfortunately, sliver exudation is so fragile that it is only guaranteed to work with an *extremely* dense sample V—the bounds on ϵ are not derived explicitly, but these algorithms probably require a 10^{-10}-sample or finer to guarantee success. Both groups of authors admit their algorithms are not practical.

In theory, the problem of manifold extraction from an MF complex seems challenging and we wonder if it is NP-hard. (Bern and Eppstein [4] observe that given a collection of triangles in the plane, it is NP-hard to determine whether some subset of them form a triangulation of the triangles' vertex set.) Nevertheless, we suspect that manifold extraction from an MF complex is not hard in practice, as slivers usually appear as isolated singletons or in small groups, especially after the MF complex filters out most slivers. We predict that in practice, 2-manifolds can reliably be extracted from MF complexes by methods based on heuristic search; whereas sliver exudation will never be practical, as its weight perturbations degrade the quality of triangulations that may already be fragile due to sparse sampling.

4 Fixed-Point Triangulations

In the 2-manifold reconstruction problem, we do not know Σ, so we cannot compute $\mathrm{Del}|_\Sigma V$. But suppose we could somehow generate a rough approximation Λ of Σ that is piecewise linear and homeomorphic to Σ, that stays somewhat close to Σ, and that is locally approximately parallel to Σ (in ways we formalize later). We will show that the triangulation $\mathcal{T}_1 = \mathrm{Del}|_\Lambda V$ is also homeomorphic to Σ, and it is likely to be a better approximation to Σ than Λ.

As Λ is piecewise linear, we can compute $\mathrm{Del}|_\Lambda V$ in high dimensions without suffering the curse of dimensionality. We take each flat piece $f \subset \Lambda$ and compute $\mathrm{Del}|_f V$ by computing a Voronoi cross-section as described in Section 2. (Observe that $\mathrm{Del}|_f V$ is a subcomplex of $\mathrm{Del}|_{\mathrm{aff}\, f} V$ where $\mathrm{aff}\, f$ denotes f's affine hull.) We stitch these fragmentary triangulations together to yield $\mathcal{T}_1$. This method requires the computation of many low-dimensional power diagrams, but nonetheless it may be much faster than computing a high-dimensional Voronoi diagram. Moreover, it can more easily take advantage of locality: if the sample points are uniformly distributed on Σ, simple spatial data structures such as spatial hashing might speed up computing $\mathrm{Del}|_f V$ to as fast as constant time. The Voronoi cross-section computations are mutually independent, making them amenable to multicore parallelization.

Next, consider computing $\mathcal{T}_2 = \mathrm{Del}|_{|\mathcal{T}_1|} V$ the same way; now each flat piece is a triangle $\tau \in \mathcal{T}_1$. Likewise compute $\mathcal{T}_3 = \mathrm{Del}|_{|\mathcal{T}_2|} V$, $\mathcal{T}_4 = \mathrm{Del}|_{|\mathcal{T}_3|} V$, etc. We eventually obtain a triangulation $\mathcal{T}^* = \mathrm{Del}|_{|\mathcal{T}^*|} V$ identical to the last input. We call $\mathcal{T}^*$ a *fixed point* of the restricted Delaunay triangulation operator, and we denote it $\mathrm{Del}^*|_\Lambda V$.

Each simplex τ in a triangulation $\mathcal{T}_i$, $i \geq 2$, is there because of a Delaunay simplex $\sigma \in \mathcal{T}_{i-1}$ that intersects τ^*. We say that σ *generates* τ. Next we show that, with one minor exception, a simplex only generates simplices with smaller circumradii—plus it might regenerate itself. This leads us to conclude that the iterations always reach a fixed point. Moreover, each successive iteration tends to shrink the circumradii. The fixed-point triangulation $\mathrm{Del}^*|_\Lambda V$ is composed of small-radius triangles; under the right conditions, this helps us to guarantee that it is close to the true manifold Σ and has similar tangents and normals.

▶ **Lemma 1** (Monotonic Circumradii Lemma)**.** *Consider the restricted Delaunay triangulation $\mathcal{T} = \mathrm{Del}|_{|\mathcal{D}|} V$, where $\mathcal{D} \subseteq \mathrm{Del}\, V$ is a set of Delaunay simplices. Each restricted Delaunay simplex $\tau \in \mathcal{T}$ is generated by a simplex in $\mathcal{D}$ with greater circumradius, or by a simplex with the same circumradius and the same circumcenter as τ (possibly by τ itself).*

Proof. Consider a simplex $\tau \in \mathcal{T}$. By the definition of restricted Delaunay triangulation, the Voronoi face τ^* dual to τ intersects some simplex $\sigma \in \mathcal{D}$, and thus σ generates τ.

Let p be a point in $\sigma \cap \tau^*$. As τ^* is τ's dual Voronoi face, the hypersphere S with center p that passes through every vertex of τ is empty (encloses no vertex in V). Let R be the radius of S, and let r_τ be the circumradius of τ. The affine circumsphere of τ is a cross-section of S, so $r_\tau \leq R$. Moreover, if $r_\tau = R$ then τ's circumcenter is the center p of S.

Let D be the diametric sphere of σ, and let r_σ be the radius of D. If $D = S$, then $r_\sigma = R$ and p is the circumcenter of σ; whereas if D encloses S, then $r_\sigma > R$. As $p \in \sigma$ and S is empty, there is only one other possibility, illustrated in Figure 1: D intersects S, and $D \cap S$ is a $(d-2)$-sphere whose affine hull is a hyperplane Π. As S is empty, σ's vertices are restricted to one side of Π (possibly lying on Π), and therefore so is σ itself. As $p \in \sigma$, at least half of S lies on that side of Π, and that portion of S is enclosed by D. Hence $r_\sigma > R$.

Therefore, $r_\tau \leq r_\sigma$, and equality is possible only if p is the circumcenter of σ and τ. ◀

To better understand fixed-point triangulations and algorithms for computing them, we define a directed almost-acyclic graph called the *restriction DAG* G. There is one node in G for each simplex in $\mathrm{Del}\, V$. G has a directed edge (σ, τ) if σ generates τ.

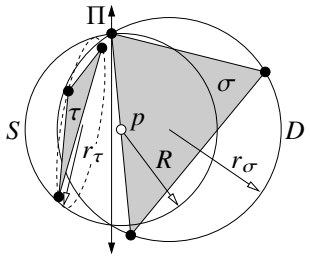


Figure 1 The Monotonic Circumradii Lemma: if σ generates τ, then $r_\tau \leq R \leq r_\sigma$.

Lemma 1 implies that G is almost acyclic. Two kinds of cycles are possible. We say that a Delaunay simplex τ is *anchored* if it intersects its own dual τ^*. An anchored simplex has a self-edge (τ, τ) in G and can generate itself. The second kind of cycle is a clique of two or more anchored simplices, representing a "degenerate" configuration in which several simplices share the same affine circumsphere, and they all intersect their shared circumcenter (which implies that the circumcenter is on the boundary of each simplex). As the simplices in such a clique are all present or all absent in a fixed-point triangulation, we contract each such clique into a single node of G (with self-edge). Then the only cycles are self-edges.

As G (after clique contraction) is a DAG (plus self-edges), we use standard tree terminology: given edges (σ, τ) and (τ, ζ), σ is a *parent* of τ and an *ancestor* of ζ, τ, and σ; ζ is a *child* of τ and a *descendant* of σ, τ, and ζ. The acyclicity of G implies that iterations of the restricted Delaunay operator always reach a fixed point, and allows us to characterize it.

▶ **Theorem 2.** *Let $\mathcal{D} \subseteq \operatorname{Del} V$ be a Delaunay subcomplex. Then the fixed-point triangulation $\operatorname{Del}^*|_{|\mathcal{D}|} V$ exists and contains every anchored simplex that has an ancestor in $\mathcal{D}$ (possibly itself), every descendant of those anchored simplices, and no other simplex.*

▶ **Corollary 3.** *For every arbitrary point set $P \subseteq \mathbb{R}^d$ and every finite point set $V \subset \mathbb{R}^d$, $\operatorname{Del}^*|_P V$ exists and can be computed with a finite number of iterations of the restricted Delaunay operator.*

The corollary is unconditional. For every P and V, we can take $\mathcal{D} = \operatorname{Del}|_P V$ and characterize the simplices in $\operatorname{Del}^*|_P V = \operatorname{Del}^*|_{|\mathcal{D}|} V$ with Theorem 2. Of course, the fact that a fixed point exists doesn't imply that it is anything more than a messy subcomplex of $\operatorname{Del} V$. If the fixed-point triangulation is to be a good manifold, P should be a good approximation of Σ and V should be an ϵ-sample for a small ϵ.

Theorem 2 serves an algorithmic purpose as well as a theoretical one. The fastest way to compute a fixed-point triangulation is not to iterate the restricted Delaunay triangulation operator; rather, we apply Theorem 2 and build part of the restriction DAG. (After we compute which simplices a particular simplex generates, there is no need to compute them again.) It suffices to generate the nodes of G that represent simplices that arise in one of the iterations. Moreover, it suffices to consider just the k-simplices. (We perturb $|\mathcal{D}|$ to prevent higher-dimensional simplices from being generated, and we gain no benefit from generating a lower-dimensional simplex that is not a face of a generated k-simplex.) If $\operatorname{Del} V$ has many slivers, we might need just a small portion of the DAG.

We do not know a useful bound on the number of iterations required to reach a fixed point (we doubt it will ever reach 8 in practice), but we present evidence in the next section that convergence is fast. One can generate a good triangulation by stopping after a few iterations, without reaching a fixed point. Moreover, if many iterations are performed, the

extra computation is likely needed only in a few locations. At any rate, the reconstruction algorithms we present in Section 8 do not actually perform these iterations; rather, they directly find triangles that can appear in a fixed-point triangulation.

We give some intuition why a fixed-point triangulation $\mathcal{T}$ partly resolves some of the ambiguities posed by slivers. In codimension one ($d-k=1$), $|\mathcal{T}|$ partitions $\mathbb{R}^d$ into an "inside" region and an "outside" region. If the circumcenter of a sliver d-simplex τ lies outside $|\mathcal{T}|$, then τ is outside also; if τ's circumcenter is inside, so is τ. (If τ contains its own circumcenter, it can be classified either way, so the fixed-point triangulation is not unique.) In codimension two or higher ($d-k \geq 2$), a fixed-point triangulation maintains an analogous relationship among the manifold k-simplices and the Voronoi $(d-k)$-cells they intersect.

5 Proximity of the Fixed-Point Triangulation to Σ

Lemma 1 tells us that iterations of the RDT operator shrink the circumradii, but says nothing about how much they shrink. Here, we show that if the input manifold does not come too close to the medial axis, the consecutive triangulations fit into a region that shrinks until it is everywhere close to Σ. We show that a parameter governing the distance from an iterate $\mathcal{T}_i$ to Σ converges quickly to a small fixed point. The distance bound yields bounds on the circumradii of the triangles and the accuracy of the tangents for each iterated triangulation, which we use to show that subsequent triangulations are k-manifolds. As a bonus, we considerably tighten the proximity (e.g., Hausdorff distance) bounds that apply to many standard algorithms for Delaunay surface reconstruction and surface mesh generation.

Let M be the medial axis of Σ. Define the *nearest-point map* $\nu : \mathbb{R}^d \setminus M \to \Sigma, q \mapsto \tilde{q}$, a function that maps each point to the nearest point on Σ. For brevity, we write $\tilde{q}$ for $\nu(q)$. If a point q does not have a unique nearest point on Σ, then $q \in M$ by the definition of medial axis, and $\tilde{q}$ is undefined. The homeomorphisms we employ will be the nearest-point map from some reconstruction $|\mathcal{T}|$ to Σ.

For $\omega \in [0,1)$, let the *ω-sausage* be the region $\Sigma_\omega = \{q : d(q,\Sigma) \leq \omega \operatorname{lfs}(\tilde{q})\}$, where $d(q,\Sigma)$ is the distance from q to Σ. Note that $\Sigma_0 = \Sigma$. The ω-sausage is what topologists call a *tubular neighborhood* of Σ. For $k=1$ its shape resembles a traditional sausage (albeit tied in a loop), whereas for $k=2$ it is more of a breakfast patty. Σ_ω is a d-dimensional union of $(d-k)$-balls: at each point $p \in \Sigma$ is centered one $(d-k)$-ball of radius $\omega \operatorname{lfs}(p)$, normal to Σ.

Next, we show that for any point set $P \subseteq \Sigma_{\bar{\omega}}$ with $\bar{\omega}$ a little less than 1, the fixed-point triangulation $\mathrm{Del}^*|_P V$ is included in $\Sigma_{\underline{\omega}}$ with $\underline{\omega}$ close to zero. The following lemma (see the full-length paper) bounds the interpolation error over a restricted Delaunay simplex in Σ_ω.

▶ **Lemma 4** (ϵ-Interpolation Lemma). *Let $\Sigma \subset \mathbb{R}^d$ be a smooth k-manifold without boundary. Let V be an ϵ-sample of Σ for some $\epsilon < 0.5$. Let $\tau \in \mathrm{Del}\, V$ be a simplex (of any dimension) whose dual Voronoi face τ^* intersects the ω-sausage Σ_ω. If $\omega \leq 1-\epsilon$ and $\omega < \frac{1-4\epsilon^2}{1+4\epsilon^2}$ (the latter precondition is equivalent to $\lambda < 0.5$), then for every point $x \in \tau$,*

$$|x\tilde{x}| \leq \Omega(\omega) \operatorname{lfs}(\tilde{x}), \quad \textit{where} \quad \Omega(\omega) = 1 - \sqrt{1 - \frac{\lambda^2}{(1-\lambda)^2}} \quad \textit{and} \quad \lambda = \sqrt{\frac{1+\omega}{1-\omega}}\epsilon.$$

In particular, for $\tau \in \mathrm{Del}|_\Sigma V$ (i.e., τ^* intersects Σ so we can set $\omega = 0$), Lemma 4 gives $|x\tilde{x}| \leq \left(1 - \sqrt{1 - \frac{\epsilon^2}{(1-\epsilon)^2}}\right) \operatorname{lfs}(\tilde{x}) \approx \frac{\epsilon^2}{2} \operatorname{lfs}(\tilde{x})$. Compare with the bound of $15\epsilon^2 \operatorname{lfs}(\tilde{x})$ in Cheng et al. [10, Theorem 13.22]—an improvement by a factor of up to 30.

Lemma 4 implies that if an input to the restricted Delaunay triangulation operator is in the ω-sausage for a middling value of ω, then the output is in the ω-sausage for a smaller ω.

▶ **Lemma 5** (Shrinking Sausage Lemma). *Let $\Sigma \subset \mathbb{R}^d$ be a bounded, smooth k-manifold without boundary. Let V be an ϵ-sample of Σ with $\epsilon \leq 0.3084$. Let $\underline{\omega}$ and $\bar{\omega}$ be two real solutions of $\omega = \Omega(\omega)$, where $\Omega(\omega)$ is defined in Lemma 4 and the solutions of interest are approximately*

$$\underline{\omega} = \frac{1}{2}\epsilon^2 + \epsilon^3 + \frac{17}{8}\epsilon^4 + 5\epsilon^5 + \frac{199}{16}\epsilon^6 + O(\epsilon^7) \quad \text{and} \quad \bar{\omega} = 1 - 8\epsilon^2 + 32\epsilon^4 - 384\epsilon^6 + O(\epsilon^8).$$

Let P be a point set included in the ω-sausage Σ_ω for some $\omega < \min\{1-\epsilon, \bar{\omega}\}$. Let the simplicial complex $\mathcal{T} = \mathrm{Del}^*|_P V$ be the fixed point of the restricted Delaunay triangulation operator given the input P. Then $|\mathcal{T}|$ is a subset of the $\underline{\omega}$-sausage $\Sigma_{\underline{\omega}}$.

Proof. By Lemma 4, if an input to the restricted Delaunay triangulation operator lies in the ω-sausage for $\omega < \min\left\{1-\epsilon, \frac{1-4\epsilon^2}{1+4\epsilon^2}\right\}$, then the output lies in the $\Omega(\omega)$-sausage. We observe that $\Omega(\underline{\omega}) = \underline{\omega}$; $\Omega(\bar{\omega}) = \bar{\omega}$; for any $\omega \in (\underline{\omega}, \bar{\omega})$, $\Omega(\omega) \in (\underline{\omega}, \omega)$; and for any $\omega \in [0, \underline{\omega})$, $\Omega(\omega) \in (\omega, \underline{\omega})$. (Note that $\underline{\omega} = \bar{\omega}$ for $\epsilon \doteq 0.308406$, hence the requirement that ϵ must be smaller.) Therefore, if we take an input in the ω-sausage with $\omega \in \left[0, \min\left\{1-\epsilon, \frac{1-4\epsilon^2}{1+4\epsilon^2}, \bar{\omega}\right\}\right)$ and iterate the restricted Delaunay triangulation operator until the output is a fixed point, then the output is in the $\underline{\omega}$-sausage. (As $\bar{\omega} \leq \frac{1-4\epsilon^2}{1+4\epsilon^2}$ for all $\epsilon \in [0, 0.308406]$, we omit $\frac{1-4\epsilon^2}{1+4\epsilon^2}$ from the lemma's preconditions.) ◀

We emphasize that convergence is rapid. For example, when $\epsilon = 0.143$, $\underline{\omega} \doteq 0.0145$ and $\bar{\omega} \doteq 0.8471$. If we start with a triangulation in the 0.847-sausage, after five iterations we obtain a triangulation in the 0.0147-sausage. Convergence is even faster for smaller values of ϵ. In the unlikely event that an input takes many iterations to reach its fixed point, one can always stop early; a small constant number of iterations suffices to reach a triangulation nearly as good as the fixed point.

Observe that the conditions of Lemma 5 are mild: P can be an arbitrary point set, so long as it is in the ω-sausage for a suitable ω. We have not yet guaranteed that the fixed-point triangulation is more than a messy subcomplex of $\mathrm{Del}\, V$—but it is a mess close to Σ. The next section describes conditions that ensure that the output is a 2-manifold.

6 Homeomorphism of the Restricted Delaunay Triangulation to Σ

Computational geometers have developed a theory of surface sampling to show that under the right conditions, a restricted Delaunay triangulation $\mathrm{Del}|_\Sigma V$ is homeomorphic to a smooth surface Σ [1, 2, 10, 13]. (More precisely, its underlying space $|\mathrm{Del}|_\Lambda V|$ is homeomorphic.) We present here an alternative theorem that allows us to handle a nonsmooth Λ. Specifically, we show that $|\mathrm{Del}|_\Lambda V|$ is homeomorphic to a smooth 2-manifold Σ without boundary if $V \subset \Sigma$ is a sufficiently dense ϵ-sample and Λ is a piecewise smooth 2-manifold that approximates Σ well (in both location and tangents). We note that Boissonnat and Oudot [7] developed a method for treating restricted Delaunay triangulations when Σ itself is a nonsmooth, Lipschitz surface, but our method meets our needs better.

An advantage of our proof is that it enlarges the value of ϵ for which certain properties hold. For example, we can show that the restricted Delaunay triangulation $\mathrm{Del}|_\Sigma V$ is homeomorphic to a 2-manifold $\Sigma \subset \mathbb{R}^d$ for any 0.2695-sample V, and the fixed-point triangulation $\mathrm{Del}^*|_\Sigma V$ is likewise for any 0.143-sample. (See full-length paper.) Not only does the former claim improve the constant $\epsilon = 0.18$ proven in Dey's book [13] for $\mathbb{R}^3$; it holds in any dimension d.

Let Σ be a smooth k-manifold embedded in $\mathbb{R}^d$. For every point $p \in \Sigma$, let $T_p\Sigma$ denote the k-flat (i.e., a k-dimensional affine subspace) tangent to Σ at p, called the *tangent space* at p, and let $N_p\Sigma$ denote the $(d-k)$-flat normal to Σ at p, called the *normal space* at p. In this paper we are mainly interested in $k = 2$, for which $T_p\Sigma$ is a plane.

Let $\Gamma, \Pi \subseteq \mathbb{R}^d$ be two flats such that $1 \le \dim\Gamma \le \dim\Pi \le d-1$. If they intersect each other at a point x, the angle $\angle(\Gamma,\Pi)$ separating them is $\max_{p\in\Gamma\setminus\{x\}} \min_{q\in\Pi\setminus\{x\}} \angle pxq$. If they do not intersect each other, translate Γ so they intersect, then measure the angle as above. This angle is never obtuse. We sometimes substitute line segments or Voronoi faces for flats, in which case we mean to measure the nonobtuse angles separating their affine hulls. It is well known that for any two points $p, q \in \Sigma$, $\angle(N_p\Sigma, N_q\Sigma) = \angle(T_p\Sigma, T_q\Sigma)$.

Let $\Lambda \subset \mathbb{R}^d$ be a piecewise smooth k-manifold. We say that Λ is *θ-parallel* to Σ if for every smooth piece $\Gamma \subseteq \Lambda$ and every point $p \in \Gamma$, $\angle(T_p\Gamma, T_{\tilde{p}}\Sigma) \le \theta$.

For any Voronoi face $f \in \operatorname{Vor} V$ of dimension 1 or greater, we say that f is *ϕ-normal* to Σ at $p \in f$ if $\angle(f, N_{\tilde{p}}\Sigma) \le \phi$. That is, f is nearly parallel to $\tilde{p}$'s normal space.

To prove that Σ and $|\operatorname{Del}|_\Lambda V|$ are homeomorphic for $k = 2$, we require that $\Lambda \subset \Sigma_\omega$ for a suitable ω, that Λ be everywhere θ-parallel to Σ, and that the faces of $\operatorname{Vor} V$ of dimension $d-2$ or greater be ϕ-normal to Σ everywhere in the ω-sausage for an appropriate ω, where $\theta + \phi < 90°$. The following lemma, proven in the full-length version of this paper, shows that Λ approximates the tangent and normal spaces of Σ with an error linear in ϵ.

▶ **Lemma 6** (Normal-Parallel Lemma). *Let $\Sigma \subset \mathbb{R}^d$ be a bounded, smooth 2-manifold without boundary. Let $V \subset \Sigma$ be an ϵ-sample of Σ. Let $\tau \in \operatorname{Del} V$ be a triangle whose Voronoi dual face τ^* intersects Σ_ω for some $\omega \le 1-\epsilon$. Let N_τ be a flat dual (complementary) to* $\operatorname{aff}\tau$. *Let*

$$\chi = \arcsin\delta + \arcsin\left(\frac{2}{\sqrt{3}}\sin(2\arcsin\delta)\right),\ \ \delta = \frac{\sqrt{\omega^2 + (1+\omega)\epsilon^2}}{1-\lambda},\ \ \lambda = \sqrt{\frac{1+\omega}{1-\omega}}\epsilon,\ \text{and}$$

$$\eta = \arctan\frac{\lambda\sqrt[4]{1-\lambda^2}}{\sqrt{1-\lambda^2\left(1+\sqrt{1-\lambda^2}\right)}} \approx \lambda + \frac{5}{12}\lambda^3 + \frac{57}{160}\lambda^5 + \frac{327}{896}\lambda^7 + O(\lambda^9).$$

Then for every point $p \in \tau$, $\angle(\operatorname{aff}\tau, T_{\tilde{p}}\Sigma) = \angle(N_\tau, N_{\tilde{p}}\Sigma) \le \chi + 2\eta$, and for every point $z \in \tau^* \cap \Sigma_\omega$, $\angle(\operatorname{aff}\tau, T_{\tilde{z}}\Sigma) = \angle(N_\tau, N_{\tilde{z}}\Sigma) \le \chi + \eta$. Therefore, τ is $(\chi+2\eta)$-parallel to Σ, and τ^* is $(\chi+\eta)$-normal to Σ wherever it intersects Σ_ω.

Proofs about the homeomorphism of restricted Delaunay triangulations usually rely on the *Topological Ball Theorem* of Edelsbrunner and Shah [15], and ours is no exception. The Topological Ball Theorem states that $|\operatorname{Del}|_\Lambda V|$ is homeomorphic to Λ if the following *topological ball property* holds: for every Voronoi j-face f that intersects Λ, $j \in [d-k, d]$, $f \cap \Lambda$ is a topological $(j+k-d)$-ball and $(\operatorname{int} f) \cap \Lambda = \operatorname{int}(f \cap \Lambda)$. (Here $\operatorname{int} f$ is an "interior" with respect to $\operatorname{aff} f$, and $\operatorname{int}(f \cap \Lambda)$ is an "interior" with respect to the manifold $f \cap \Lambda$, not with respect to the ambient space $\mathbb{R}^d$.) Moreover, Λ intersects no Voronoi j-face for $j < d-k$. The last condition can be simulated by infinitesimal perturbations of Λ. The following theorem is one of our main contributions; see the full-length paper for a proof.

▶ **Theorem 7** (θ-ϕ Homeomorphism Theorem). *Let $\Sigma \subset \mathbb{R}^d$ be a connected, smooth 2-manifold. Let $\Lambda \subset \Sigma_\omega$ be a piecewise smooth 2-manifold where $\omega < (\sqrt{5}-1-2\epsilon^2)/(\sqrt{5}-1+2\epsilon^2)$ (equivalently, $\lambda < \sqrt{\left(\sqrt{5}-1\right)/2} \doteq 0.7861$). Suppose that there is a $\theta \ge 0°$ and a $\phi \ge 0°$ such that $\theta + \phi < 90°$, Λ is θ-parallel to Σ, and every face $f \in \operatorname{Vor} V$ of dimension $d-2$ or $d-1$ is ϕ-normal to Σ at every point on $f \cap \Sigma_\omega$. Suppose that for every $p \in \Sigma$, Λ intersects $N_p\Sigma$ at most once (i.e., $\nu : \Lambda \to \Sigma$ is injective), and that Λ intersects no i-face of $\operatorname{Vor} V$ for $i < d-2$. Then for every $j \in [0,2]$ and every $(d-2+j)$-dimensional face $f \in \operatorname{Vor} V$ such that $f \cap \Lambda \ne \emptyset$, the intersection $f \cap \Lambda$ is a topological j-ball and $(\operatorname{int} f) \cap \Lambda = \operatorname{int}(f \cap \Lambda)$. Thus Λ and $\operatorname{Vor} V$ satisfy the topological ball property and $|\operatorname{Del}|_\Lambda V|$ is homeomorphic to Λ.*

▶ **Corollary 8.** *Under the assumptions of Theorem 7, $|\operatorname{Del}|_\Lambda V|$ is homeomorphic to Σ.*

The obvious use of Corollary 8 is to show that, given a suitable Λ, the fixed-point procedure produces a triangulation of Σ. We have a second use for it: our algorithm in Section 8 does not construct a suitable Λ; instead, it directly computes all the triangles that might be in a fixed-point triangulation. By the following corollary (proven in the full-length paper), we know that some subset of those triangles is a good triangulation of Σ.

▶ **Corollary 9.** *Let $\Sigma \subset \mathbb{R}^3$ be a bounded, smooth 2-manifold without boundary. Let $V \subset \Sigma$ be a finite 0.143-sample of Σ. Then the underlying space of the fixed-point triangulation $\mathrm{Del}^*|_\Sigma V$ is homeomorphic to Σ.*

7 Classifying Critical Points

Recall that a simplex $\tau \in \mathrm{Del}\, V$ is *anchored* if it intersects its own Voronoi dual τ^*. We call the intersection $\tau \cap \tau^*$ a *critical point* (of the distance function $d(p, V) = \min_{u \in V} |pu|$). Dey, Giesen, Ramos, and Sadri [12] show that for a sufficiently dense sample, every critical point is either a *manifold critical point* close to the manifold Σ or a *medial critical point* close to Σ's medial axis M. The following theorem is our tighter version of their result.

▶ **Theorem 10** (Critical Point Separation Theorem)**.** *Let $\Sigma \subset \mathbb{R}^d$ be a smooth k-manifold. Let V be an ϵ-sample of Σ for some $\epsilon \leq 0.5$. Let $\tau \in \mathrm{Del}\, V$ be a simplex (of any dimension) whose dual Voronoi face τ^* intersects τ. Let $z = \tau \cap \tau^*$. Let*

$$\mathring{\omega} = \frac{2 - \sqrt{4 - 12\epsilon^2 + \epsilon^4} - \epsilon^2}{4} = \frac{1}{2}\epsilon^2 + \frac{1}{2}\epsilon^4 + \frac{3}{4}\epsilon^6 + \frac{11}{8}\epsilon^8 + O(\epsilon^{10}) \quad \textit{and}$$

$$\mu = \frac{1 - \sqrt{1 - 4\epsilon^2}}{2} = \epsilon^2 + \epsilon^4 + 2\epsilon^6 + 5\epsilon^8 + O(\epsilon^{10}).$$

Then either $|z\tilde{z}| \leq \mathring{\omega}\, \mathrm{lfs}(\tilde{z})$ (i.e., $z \in \Sigma_{\mathring{\omega}}$) or $|zm| \leq \mu |\tilde{z}m|$ for some $m \in M$, where M is the medial axis of Σ.

Here, we describe a method for distinguishing the two types of critical points that works for larger values of ϵ (in theory) than the well-known *cocone criterion* of Amenta, Choi, Dey, and Leekha [2]. Our method is useful not only for our algorithm, but also for Dey et al.'s algorithm for computing flow complexes. In particular, our test works in any dimension, and enables the computation of fixed-point complexes and flow complexes in higher dimensions. The cocone criterion depends on having accurate estimates of the local tangent planes, but in high dimensions only poor approximations can be obtained cheaply.

Let v be a vertex of an anchored simplex τ. Let $v^* \in \mathrm{Vor}\, V$ denote its Voronoi cell. Consider the set of points at the centers of the medial balls tangent to Σ at v: these centers all lie on the normal space $N_v\Sigma$, and each one lies either on the medial axis M or at infinity. If the codimension is 1 (i.e., $d - k = 1$), there are two such medial points; if the codimension is greater, there are infinitely many. In either case, at least one medial point is not at infinity, and all these medial points, including those at infinity, lie in the Voronoi cell v^*.

Consider first the case of codimension 1. Suppose that we can explicitly construct the Voronoi diagram (i.e., d is small). Following Amenta and Bern [1], we define the *poles of v* to be two vertices of v^*, one on each side of Σ. Let the pole p^+ be the vertex of v^* furthest from v, unless v^* is unbounded, in which case p^+ is a point at infinity whose direction is chosen so the ray vp^+ lies in v^*. Let the pole p^- be the vertex of v^* furthest from v such that $\angle p^+ v p^- > 90°$. Amenta and Bern show that p^- is on the opposite side of Σ from p^+.

Our classifier for codimension 1 is simple. Given a critical point $z = \tau \cap \tau^*$, choose any vertex v of τ and compute v's poles p^+ and p^-. Set $p = p^-$ if v^* is unbounded; otherwise,

select the pole $p \in \{p^+, p^-\}$ that minimizes $\angle pvz$. Then, z is a medial critical point if $|vz|/|vp| > 0.313$; a manifold critical point otherwise. This classifier relies on a lemma of Amenta and Bern, which generalizes to higher dimensions and codimensions without change.

▶ **Lemma 11** (Normal Lemma [1]). *Let $\Sigma \subset \mathbb{R}^d$ be a bounded, smooth k-manifold without boundary. Let $V \subset \Sigma$ be an ϵ-sample of Σ for $\epsilon < 1$. Let p be a point in the Voronoi cell v^* of a sample $v \in V$ such that $|vp| \geq \rho \operatorname{lfs}(v)$ for some $\rho > 0$. Then $\angle(vp, N_v\Sigma) \leq \beta(\rho)$, where $\beta(\rho) = \arcsin \frac{\epsilon}{1-\epsilon} + \arcsin \frac{\epsilon}{\rho(1-\epsilon)}$.*

▶ **Lemma 12** (Classification Lemma for Codimension 1). *Let $\Sigma \subset \mathbb{R}^d$ be a bounded, smooth $(d-1)$-manifold without boundary. Let $V \subset \Sigma$ be an ϵ-sample of Σ for $\epsilon \leq 0.2327$. Let $\tau \in \operatorname{Del} V$ be a simplex such that $\tau \cap \tau^* \neq \emptyset$, and let z be τ's circumcenter $\tau \cap \tau^*$. Let v be any vertex of τ. Choose a pole p of the Voronoi cell v^* as described above. If $|vz|/|vp| > 0.313$, then z is a medial critical point; otherwise, z is a manifold critical point.*

Proof. As v^* contains medial points, each at a distance of at least $\operatorname{lfs}(v)$ from v, $|vp^+| \geq \operatorname{lfs}(v)$. Amenta and Bern [1] show that $|vp^-| \geq \operatorname{lfs}(v)$ too. Hence $|vp| \geq \operatorname{lfs}(v)$.

If z is a manifold critical point, then z lies in the $\mathring{\omega}$-sausage by Theorem 10. In the full-length paper we show that the circumradius of τ is $|vz| \leq \sqrt{\mathring{\omega}^2 + (1+\mathring{\omega})\epsilon^2} \operatorname{lfs}(\tilde{z})$, and $|\tilde{z}v| \leq \lambda \operatorname{lfs}(\tilde{z})$, where $\lambda = \sqrt{\frac{1+\mathring{\omega}}{1-\mathring{\omega}}}\epsilon$. By the Feature Translation Lemma [13, 10], $|vz| \leq \sqrt{\mathring{\omega}^2 + (1+\mathring{\omega})\epsilon^2} \operatorname{lfs}(v)/(1-\lambda)$. For $\epsilon \leq 0.2327$, $\mathring{\omega} < 0.0287$ and $|vz|/|vp| < 0.3127$.

If z is a medial critical point, then there is a medial axis point $m \in M$ such that $|zm| \leq \mu|\tilde{z}m| \leq \mu(|\tilde{z}z| + |zm|)$; therefore $|zm| \leq |\tilde{z}z|\mu/(1-\mu)$. Hence, $\operatorname{lfs}(v) \leq |vm| \leq |vz| + |zm| \leq |vz| + |\tilde{z}z|\mu/(1-\mu) \leq |vz| + |vz|\mu/(1-\mu) = |vz|/(1-\mu)$ and $|vz| \geq (1-\mu)\operatorname{lfs}(v)$.

By the Normal Lemma, $\angle(vp, N_v\Sigma) \leq \beta(1)$ and $\angle(vz, N_v\Sigma) \leq \beta(1-\mu)$. For codimension 1, $N_v\Sigma$ is a line, so one pole satisfies $\angle pvz \leq \beta(1) + \beta(1-\mu)$ and one satisfies $\angle pvz \geq 180° - \beta(1) - \beta(1-\mu)$. We choose p to be the former.

As v is a vertex of τ and z is the circumcenter of τ, every point q for which $\angle vzq > 90°$ is closer to another vertex of τ and cannot lie in the Voronoi cell v^*. But $p \in v^*$, so $\angle vzp \leq 90°$ and $|vz|/|vp| \geq \cos\angle pvz \geq \cos(\beta(1) + \beta(1-\mu))$. For $\epsilon \leq 0.2327$, $|vz|/|vp| > 0.3134$. ◀

We adapt this test to 2-manifolds in codimension 2 or higher by taking a three-dimensional cross-section of the Voronoi cell v^*. First, we use the following simple algorithm to compute an approximate tangent plane $\hat{T}_v\Sigma$ for v: find v's nearest neighbor $w \in V$, then find the vertex $x \in V$ that minimizes the circumradius of $\triangle vwx$. We show in the full-length paper that $\triangle vwx$ has a small circumradius and its affine hull $\hat{T}_v\Sigma = \operatorname{aff} \triangle vwx$ is an adequate approximation of v's tangent plane $T_v\Sigma$. Let Ξ be the 3-flat that includes $\hat{T}_v\Sigma$ and the critical point z. We compute the cross-section $\Xi \cap v^*$ with the method described in Section 2. As we seek just one cell in a three-dimensional power diagram, this computation can be dualized to a three-dimensional convex hull computation, which can be performed in $O(n \log n)$ time.

If we knew the true tangent plane $T_v\Sigma$, we could use the same test and analysis as Lemma 12; but only an approximation $\hat{T}_v\Sigma$ is available, so we adjust the test to compensate and require a stricter constraint on ϵ. (A better tangent plane approximation method would enable us to loosen this constraint.) Given a critical point $z = \tau \cap \tau^*$, choose any vertex v of τ, compute $\hat{T}_v\Sigma$, let $\Xi = \operatorname{aff}(\hat{T}_v\Sigma \cup \{z\})$, and compute the three-dimensional polyhedron $\Xi \cap v^*$ and its poles p^+ and p^-. Select a pole $p \in \{p^+, p^-\}$ in the same manner as for codimension 1. Then, z is a medial critical point if $|vz|/|vp| > 0.182$; a manifold critical point otherwise. This test works for any $\epsilon \leq 0.1522$. We defer the proof to the full-length paper.

8 Constructing Fixed-Point Complexes

We classify each anchored triangle as a *manifold anchored triangle* or a *medial anchored triangle* according to whether its circumcenter is a manifold critical point or a medial critical point. A *chained triangle* is a triangle that is generated by a manifold anchored triangle or another chained triangle. The *manifold fixed-point complex* (MF complex) is the set of all manifold anchored triangles and chained triangles. Like the flow complex, the MF complex is not a manifold (because of slivers), but it is a useful geometric representation of Σ in other ways: its Hausdorff distance to Σ is small and all its triangles are good estimates of Σ's local tangent space. Unlike the flow complex, the MF complex obtains the latter guarantee without prohibiting sample points from being too close together. Being a subcomplex of $\operatorname{Del} V$, the MF complex is a simpler alternative to the flow complex that offers better approximation bounds. Moreover, for $\epsilon \leq 0.143$, some subcomplex of the MF complex is a 2-manifold that approximates Σ well (namely, $\operatorname{Del}^*|_\Sigma V$). (We conjecture that for small ϵ, every MF triangle participates in at least one such 2-manifold.)

Here, we present an algorithm that constructs the MF complex for a 2-manifold $\Sigma \subset \mathbb{R}^d$ in time polynomial in d. In the special case $d = 3$, standard methods can efficiently extract a 2-manifold from the MF complex; thus we have a surface reconstruction algorithm. In higher dimensions, we believe heuristics can do the same in practice.

We take as input an n-point ϵ-sample $V \subset \Sigma$. If possible, we begin by constructing $\operatorname{Del} V$ and $\operatorname{Vor} V$, then we use $\operatorname{Del} V$ to identify the manifold anchored triangles. If d is too large for the construction of $\operatorname{Del} V$ to be feasible, we identify the anchored triangles by enumerating candidate triangles and checking each one to see if its diametric sphere is empty.

Next, we classify the anchored triangles as manifold or medial as described in Section 7. We find the chained triangles as described in Section 4. This requires us to determine which $(d-2)$-faces of $\operatorname{Vor} V$ intersect anchored or chained triangles. If we have constructed $\operatorname{Del} V$, an efficient way to find these Delaunay-Voronoi intersections is to simultaneously "walk" through both $\operatorname{Del} V$ and $\operatorname{Vor} V$ in synchrony, working outward from each anchored triangle along the chained triangles in a depth-first manner. If constructing $\operatorname{Del} V$ in infeasible, we construct a Voronoi cross-section for each MF triangle as described in Section 2.

Let $\mathcal{T}$ be the MF complex. In $\mathbb{R}^3$, we can extract a 2-manifold from $\mathcal{T}$ with a standard method [2, 13] whose correctness is well known. All MF triangles lie sufficiently close to Σ (in $\Sigma_{\underline{\omega}}$) that we can assign each edge two "sides": two triangles in $\mathcal{T}$ are on the same "side" of a shared edge if they meet at a dihedral angle less than 90°; on opposite sides otherwise. We *prune* from $\mathcal{T}$ every triangle that has an edge with no matching triangle on the other side. Each pruned triangle may render other edges one-sided, prompting further pruning. When no such triangle survives, we label the tetrahedra of $\operatorname{Del} V$ that are not enclosed by the surviving triangles by performing a depth-first search. The search begins at the unbounded outer cell of $\operatorname{Del} V$ and it labels every tetrahedron it visits "outside," but it never crosses a triangle still in $\mathcal{T}$. Finally, we output every triangle that separates an unlabeled tetrahedron from a tetrahedron labeled "outside" or the outer cell of $\operatorname{Del} V$. The method succeeds because by Corollary 9, $\mathcal{T}$ includes the 2-manifold $\operatorname{Del}^*|_\Sigma V$, whose triangles are not pruned and which encloses tetrahedra that will not be labeled "outside." (The output manifold might enclose additional tetrahedra, which do no harm.)

▶ **Theorem 13.** *Let $\Sigma \subset \mathbb{R}^d$ be a bounded, smooth 2-manifold without boundary. Let $V \subset \Sigma$ be a finite ϵ-sample of Σ for $\epsilon \leq 0.143$. Our fixed-point reconstruction algorithm produces a Delaunay subcomplex $\mathcal{T} \subset \operatorname{Del} V$ such that the map $\nu : p \mapsto \tilde{p}$ sends each point $p \in \mathcal{T}$ a distance of at most $\underline{\omega}\operatorname{lfs}(\tilde{p})$, where $\underline{\omega} \approx \frac{1}{2}\epsilon^2 + \epsilon^3 + \frac{17}{8}\epsilon^4 + 5\epsilon^5 + \frac{199}{16}\epsilon^6$ and $\tilde{p}$ is the point*

nearest p on Σ; and for every triangle $\tau \in \mathcal{T}$ and every point $p \in \tau$, $\angle(\text{aff}\,\tau, T_{\tilde{p}}\Sigma) \leq \chi + 2\eta$, where χ and η are defined in Lemma 6 with $\omega \leftarrow \underline{\omega}$. Moreover, some subset of $\mathcal{T}$ has an underlying space homeomorphic to Σ, and for $d = 3$ we can extract it efficiently.

In almost all practical cases, the MF complex has $O(n)$ triangles, but by placing $\Theta(n)$ sample points on the boundary of an empty sphere, it is possible to create pathological, sliver-packed examples that have $\Theta(n^2)$ triangles in $\mathbb{R}^3$ or $\Theta(n^3)$ triangles in $\mathbb{R}^5$.

If we explicitly construct $\operatorname{Del} V$ and $\operatorname{Vor} V$, the randomized incremental algorithm of Clarkson and Shor [11] constructs them in expected $O(n^{\lceil d/2 \rceil})$ time, but for many point sets that arise in practice it takes expected $O(n \log n)$ time [3]. Then it takes time linear in the size of $\operatorname{Del} V$ to identify the anchored triangles and (in $\mathbb{R}^3$) to extract the final manifold at the end. By walking through $\operatorname{Del} V$, we identify the chained triangles at a worst-case cost of $O(n)$ time per MF triangle, but the typical cost is closer to $O(1)$ per triangle.

If d is large enough that we cannot construct $\operatorname{Del} V$, the brute force algorithm for identifying the anchor triangles (test all $\binom{n}{3}$ potential triangles for an empty diametric sphere) takes $O(n^4)$ time. However, as manifold anchored triangles tend to be small, for many point sets that arise in practice, it may be possible to reduce this time substantially by use of bucketing, spatial hashing, or quadtrees. We can identify the chain triangles by computing a Voronoi cross-section for each MF triangle at a cost of $O(n \log n)$ time per triangle.

In $\mathbb{R}^3$, we identify the poles of every Voronoi cell and classify the critical points as manifold or medial in time linear in the size of $\operatorname{Vor} V$. For $d \geq 4$, we compute a three-dimensional Voronoi cell cross-section for each critical point at a cost of $O(n \log n)$ time each.

9 Conclusions

The main open problem we would like to solve is to efficiently find a fixed-point triangulation that is homeomorphic to Σ for $d \geq 4$. There are two very different approaches: one could find a "starter" triangulation Λ that approximates Σ well enough to apply fixed-point iterations; or one could find an efficient algorithm for extracting a 2-manifold from the MF complex. Both problems have stumped us in theory despite long efforts. As we noted in Section 3, we suspect that manifold extraction is hard in theory, but easy in practice.

It seems promising to consider what the MF complex looks like if we permit simplices of dimension greater than 2, even when Σ is a 2-manifold. In three dimensions, for instance, the MF complex would include sliver tetrahedra that contain their own circumcenters and perhaps other, chained, sliver tetrahedra. We conjecture that for $\epsilon \leq 0.143$, this full-dimensional MF complex is homotopy equivalent to Σ.

We are oddly optimistic that our ideas can be modified to find good representations of higher-dimensional manifolds, despite the problems we discuss at the end of Section 1. We observe that the simplices with the most terrible normals and tangents have Voronoi duals that lie close to Σ. Counterintuitively, we could perhaps better model Σ by taking a restricted Delaunay triangulation $\operatorname{Del}|_\Lambda V$ for a manifold Λ that approximates Σ but is displaced far enough away from Σ to dodge the Voronoi duals of the most troublesome simplices. Such a triangulation would not be a fixed point of the restricted Delaunay triangulation operator, but it might be found by similar methods.

Acknowledgments. The author J. Shewchuk began this work during a sabbatical at INRIA Sophia-Antipolis during the spring of 2010. He would like to thank the Geometrica Group for their kind reception, and he particularly thanks Jean-Daniel Boissonnat and Arijit Ghosh for extended discussions of the problem of manifold reconstruction.

This work was supported in part by INRIA Sophia-Antipolis, in part by the National Science Foundation under Awards CCF-0635381, IIS-0915462, and CCF-1423560, in part by the University of California Lab Fees Research Program under Grant 09-LR-01-118889-OBRJ, in part by a National Science Foundation Graduate Research Fellowship, and in part by an Alfred P. Sloan Research Fellowship. We thank our sponsors for their support.

References

1 Nina Amenta and Marshall Bern. Surface Reconstruction by Voronoi Filtering. *Discrete & Computational Geometry*, 22(4):481–504, June 1999.

2 Nina Amenta, Sunghee Choi, Tamal Krishna Dey, and Naveen Leekha. A Simple Algorithm for Homeomorphic Surface Reconstruction. *International Journal of Computational Geometry and Applications*, 12(1–2):125–141, 2002.

3 Nina Amenta, Sunghee Choi, and Günter Rote. Incremental Constructions con BRIO. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 211–219, San Diego, California, June 2003. Association for Computing Machinery.

4 Marshall Bern and David Eppstein. Mesh Generation and Optimal Triangulation. In *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, Singapore, 1992.

5 Jean-Daniel Boissonnat and Arijit Ghosh. Manifold Reconstruction Using Tangential Delaunay Complexes. *Discrete & Computational Geometry*, 51(1):221–267, January 2014.

6 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Sampling and Meshing of Surfaces. *Graphical Models*, 67(5):405–451, September 2005.

7 Jean-Daniel Boissonnat and Steve Oudot. Provably Good Sampling and Meshing of Lipschitz Surfaces. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, pages 337–346, Sedona, Arizona, June 2006.

8 Siu-Wing Cheng, Tamal Krishna Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver Exudation. *Journal of the ACM*, 47(5):883–904, September 2000.

9 Siu-Wing Cheng, Tamal Krishna Dey, and Edgar A. Ramos. Manifold Reconstruction from Point Samples. In *Proceedings of the Sixteenth Annual Symposium on Discrete Algorithms*, pages 1018–1027, Vancouver, British Columbia, Canada, January 2005. ACM–SIAM.

10 Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. CRC Press, Boca Raton, Florida, December 2012.

11 Kenneth L. Clarkson and Peter W. Shor. Applications of Random Sampling in Computational Geometry, II. *Discrete & Computational Geometry*, 4(1):387–421, December 1989.

12 Tamal K. Dey, Joachim Giesen, Edgar A. Ramos, and Bardia Sadri. Critical Points of Distance to an ϵ-Sampling of a Surface and Flow-Complex-Based Surface Reconstruction. *International Journal of Computational Geometry and Applications*, 18(1–2):29–62, 2008.

13 Tamal Krishna Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, New York, 2007.

14 Herbert Edelsbrunner. Surface Reconstruction by Wrapping Finite Sets in Space. In Boris Aronov, Saugata Basu, János Pach, and Micha Sharir, editors, *Discrete and Computational Geometry: The Goodman–Pollack Festschrift*, pages 379–404. Springer-Verlag, Berlin, 2003.

15 Herbert Edelsbrunner and Nimish R. Shah. Triangulating Topological Spaces. *International Journal of Computational Geometry and Applications*, 7(4):365–378, August 1997.

16 Julia Flötotto. *A Coordinate System Associated to a Point Cloud Issued from a Manifold: Definition, Properties and Applications*. PhD thesis, Université Nice Sophia Antipolis, 2003.

17 Joachim Giesen and Matthias John. The Flow Complex: A Data Structure for Geometric Modeling. *Computational Geometry: Theory and Applications*, 39:178–190, 2008.

Congruence Testing of Point Sets in 4-Space*

Heuna Kim[1] and Günter Rote[2]

1 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
heunak@mi.fu-berlin.de

2 **Institut für Informatik, Freie Universität Berlin, Berlin, Germany**
rote@inf.fu-berlin.de

Abstract

We give a deterministic $O(n \log n)$-time algorithm to decide if two n-point sets in 4-dimensional Euclidean space are the same up to rotations and translations. It has been conjectured that $O(n \log n)$ algorithms should exist for any fixed dimension. The best algorithms in d-space so far are a deterministic algorithm by Brass and Knauer (2000) [6] and a randomized Monte Carlo algorithm by Akutsu (1998) [1]. In 4-space, they take time $O(n^2 \log n)$ and $O(n^{3/2} \log n)$, respectively. Our algorithm exploits the geometric structure and the properties of 4-dimensional space, such as angles between planes, packing numbers, Hopf fibrations, and Plücker coordinates.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Congruence Testing Algorithm, Symmetry, Computational Geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.48

1 Introduction

Given two n-point sets $A, B \subset \mathbb{R}^4$, we want to test whether they are *congruent*, i. e., whether there exists an orthogonal matrix R and a translation vector t such that $RA + t = \{ Ra + t \mid a \in A \}$ equals B. Our main result is an optimal algorithm for this task.

▶ **Theorem 1.** *We can decide if two n-point sets A and B in 4-space are congruent in $O(n \log n)$ time and $O(n)$ space.*

The Computational Model. We use the Real Random-Access Machine (Real-RAM) model, which is common in Computational Geometry. We use square roots and basic operations from linear algebra and analytic geometry, such as orthonormal bases in 4 dimensions and eigenvectors of 2×2-matrices. We assume that we can compute exact results in constant time. Sines and cosines and their inverses are also used, but these operations can be eliminated in favor of purely algebraic operations. In order to test exact congruence, it is inevitable to use exact arithmetic with real numbers. If the model is restricted to rational arithmetic, the problem would be severely constrained. For example, five-fold symmetry is impossible in any dimension. The interesting and difficult inputs for congruence testing are the symmetric cases, and these instances appear only with irrational coordinates.

* This research was supported by the Deutsche Forschungsgemeinschaft within the research training group *Methods for Discrete Structures* (GRK 1408).

© Heuna Kim and Günter Rote;
licensed under Creative Commons License CC-BY

Pruning, Condensing, and Dimension Reduction. The conventional way of *pruning* is to classify points of A according to some invariant that is simply computable and to keep only the smallest class. For example, after we prune points by the distance to the origin, we obtain points on a sphere. We generalize this principle to a more general method: We try to *condense* a finite set A to a nonempty set $A' = F(A)$ of smaller size, not necessarily a subset of A. This should be done in an equivariant way, that is, $F(R \cdot A) = R \cdot F(A)$ for any rotation matrix R. For example, when we have a perfect matching of the points A that has been constructed by geometric criteria, we can replace the vertex set by the midpoints of the edges. Condensing can be repeated until it gets stuck, without affecting the time and space complexity, provided that we guarantee a size reduction by a factor $c < 1$.

At first glance, condensing looks dangerous because it *throws away information*. It could introduce new symmetries, and it might happen that the condensed sets are congruent, whereas the original sets are not. Therefore, the condensed set should be kept only temporarily. The prime goal of iterative condensing steps is to reduce point sets eventually to small enough sets A'' and B'' so that we can afford *dimension reduction*.

The standard dimension reduction procedure, which we call *1+3 dimension reduction*, is as follows: Choose $a_0 \in A''$ arbitrarily and successively match it to each $b \in B''$. We fix an arbitrary rotation R such that $Ra_0 = b$. Then the axis ℓ through $a_0 = b$ is fixed, and the problem becomes a three-dimensional problem of finding a rotation in the subspace $\ell^\perp$ perpendicular to ℓ. At this stage, we go back to the original sets A and B and project them onto $\ell^\perp$. To each projected point, we attach the signed distance from $\ell^\perp$ as a label. We then look for 3-dimensional congruences that respect this labeling information. We extend this to a new technique, called *2+2 dimension reduction*, for the case when the image of a two-dimensional plane is known, see Section 7.

Pruning and condensing are very powerful and versatile methods, because we can use them with any criterion or construction that one might think of (provided that it is not too hard to compute). They allow us to concentrate on the cases where condensing makes no progress, and these cases are highly structured and symmetric. The difficulty is to pick the right pruning criteria, and to decide how to proceed when pruning and condensing gets stuck.

Condensing and Pruning Convention. We will mostly describe the condensing steps only for the set A, but, whenever we apply any condensing on A, we will apply the same condensing to B in parallel. If A and B are congruent, B will undergo exactly the same sequence of steps as A. If at any point, a difference manifests itself, A and B are not congruent, and we can terminate. In the case of pruning, we choose the class of the smallest cardinality. To ensure that the results of A and B are identical, we use some lexicographic rule for tie-breaking. We will just say that we "prune by criterion X" without explicitly mentioning a tie-breaking rule.

Previous Work and Significance of the Results. Congruence is a fundamental geometric problem with a long history. A lower bound of $\Omega(n \log n)$ holds already in one dimension [3, 4]. Optimal algorithms with a running time of $O(n \log n)$ were developed in two dimensions [12, 3] (first in 1976), and in three dimensions [14, 4, 2]. It is believed by many researchers in the field that the problem is solvable in $O(n \log n)$ time in any *fixed* dimension. However, so far the best algorithms in general dimension d are exponential in d. The first approach in d-space by Alt, Mehlhorn, Wagener, and Welzl [2] reduces the d-dimensional problem to n instances in dimension $d-1$, leading to a running time of $O(n^{d-2} \log n)$. Matoušek (mentioned in [1]) improved this dimension reduction approach to an $O(n^{\lfloor d/2 \rfloor} \log n)$ algorithm by matching two points at a time. He used *closest pairs*, that is, pairs of points that achieve the minimum

distance. The number of closest pairs is $O(n)$ because each point can belong to at most $O(1)$ closest pairs, by a packing argument. Brass and Knauer [6] extended this idea to triplets and obtained an $O(n^{\lceil d/3 \rceil} \log n)$ algorithm. The extension to k-tuples with $k > 3$ is difficult, because a special treatment is required when k-tuples degenerate. Akutsu [1] gave a randomized Monte Carlo algorithm that uses Matoušek's idea. Its running time, not explicitly stated in [1], is $O(n^{\lfloor d/2 \rfloor/2} \log n)$ for $d \geq 6$ and $O(n^{3/2} \log n)$ for $d = 4, 5$. These are the best algorithms for general dimensions d to date.

The new algorithm. The techniques of closest pairs, pruning, and dimension reduction have been used for congruence testing before. We extend these ideas and apply them in a novel way. Our algorithm uses closest pairs not just as a convenient tool for matching, but it extracts helices around great circles on 3-sphere from the structure of the closest-pair graph.

Our algorithm uses 4-dimensional geometric tools: We measure angles between planes in 4-space and use Plücker coordinates, representing planes as antipodal points on the 5-sphere. As a result of symmetries, we will also encounter beautiful mathematical structures. In particular, the structure of Hopf bundles organizes the special case of isoclinic planes in a pleasant way. We will need the classification of Coxeter groups for special cases.

We have not yet succeeded in designing an $O(n \log n)$-time algorithm for an arbitrary fixed dimension. Our new algorithm for the four-dimensional case is already quite complicated and involves nontrivial mathematical properties of the 3-sphere, and therefore it is not straightforward to extend it. We view it as a major step towards a better understanding of the problem that would eventually lead to better solutions for higher dimensions.

The detailed analysis and background of the algorithm is given in the full version [11].

Preliminaries. In the problem, the translation vector t can be eliminated by translating A and B so that their centroids become the origin. Furthermore, we only consider orthogonal matrices R of determinant $+1$ (direct congruences). If we want to allow orthogonal matrices of determinant -1 (mirror congruence), we repeat the algorithm with a reflected copy of B.

For packing arguments, we denote by K_d the kissing number on $\mathbb{S}^d$: the maximum number of equal interior-disjoint balls on the d-dimensional sphere $\mathbb{S}^d$ that can simultaneously touch a ball of the same size. We use the known bounds $K_2 = 5$, $K_3 = 12$, and $40 \leq K_5 \leq 44$.

We will declare the problem to be trivially solvable if the closest-pair distance δ is large, i.e., $\delta > \delta_0 := 0.0005$. This implies that the input size $|A|$ is bounded by $n_0 < 3.016 \times 10^{11}$, and hence, by 1+3 dimension reduction, the problem can be reduced to at most n_0 instances of 3-dimensional congruence testing, taking $O(n \log n)$ time overall.

2 Overview of the Algorithm

We first give an overview of our algorithm, omitting details and some special cases. Figure 1 gives a schematic view. Our goal is to apply condensing repetitively until we can apply one of the two dimension reduction techniques to the original input point sets: 1+3 dimension reduction or 2+2 dimension reduction. We can afford 1+3 dimension reduction, as described in Section 1, as soon as the input has been reduced to small size, i.e., if $|A| \leq n_0$.

Similarly, if we find a set $\mathcal{P}$ of equivariant great circles such that $1 \leq |\mathcal{P}| \leq 829$, we can trigger 2+2 dimension reduction. By applying 2+2 dimension reduction, the problem boils down to congruence testing of labeled points on a 2-dimensional torus under translations. This can be solved in time $O(n \log n)$, see Section 7.

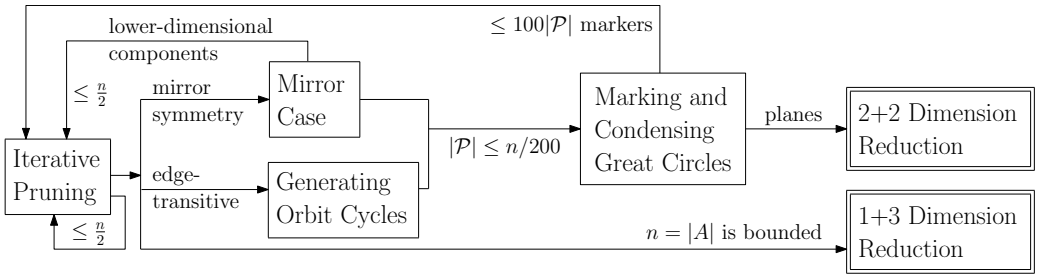


Figure 1 The general flow of the algorithm.

The algorithm begins with pruning by distance from the origin. We may thus assume without loss of generality that the resulting set A' lies on the unit 3-sphere $\mathbb{S}^3 \subset \mathbb{R}^4$.

In iterative pruning (Section 3), we first check if the minimum distance δ between points of A' is bigger than the threshold δ_0. If yes, we conclude that $|A'| \leq n_0$ and trigger 1+3 dimension reduction. Otherwise, we construct the closest-pair graph G on A'. We then prune the edges of G by the congruence type of their edge figures. An edge figure consists of two adjacent vertices and all their neighbors. We apply pruning with other criteria to the set A' until the resulting set A_0 cannot be further reduced. Then all edge neighborhoods in the graph G are congruent. This allows us to find either *orbit cycles* in G, or a subset of A_0 with mirror symmetry. An orbit cycle is a cyclic path $a_1a_2 \dots a_\ell a_1$ with a rotation R such that $Ra_i = a_{(i \bmod \ell)+1}$; in other words, it is the orbit of the point a_1 under a rotation R.

Mirror symmetry swaps the two endpoints for each edge of G and maps the whole neighborhood of the edge onto itself. If the point set A_0 has mirror symmetry, then by the classical classification of discrete reflection groups, A_0 must be related to one of the regular four-dimensional polyhedra, or A_0 (and G) is the Cartesian product of two regular polygons in orthogonal planes. The former case can be excluded, since $\delta < \delta_0$, and in the latter case, we can proceed to Section 6 for marking and condensing great circles by letting $\mathcal{P}$ as a set of circumscribing great circles of regular polygons, see Section 5.

Let us look at the case when we have found orbit cycles (Section 4). Geometrically, an orbit cycle lies on a helix around a great circle C. We associate this great circle C to each orbit cycle and get a collection $\mathcal{P}$ of great circles on $\mathbb{S}^3$. We treat these great circles as objects in their own right, and we construct the closest-pair graph on $\mathcal{P}$. For this, we use the Plücker embedding of the corresponding planes into $\mathbb{S}^5/\mathbb{Z}_2 = \mathbb{RP}^5$, mapping each plane to a pair of antipodal points on the 5-sphere $\mathbb{S}^5$. Then we construct the closest-pair graph of planes with respect to the Euclidean distance in the Plücker embedding.

For each closest pair (C, D) in $\mathcal{P}$, we look at the projection of C to the plane containing D. Generically, this is an ellipse, and the major axis of this ellipse marks two points on C. The set of all markers replaces the set A_0. This completes a successful condensing step, and we restart and continue as before. If all projected "ellipses" turn out to be circles, we will be able to find a subfamily of great circles in a special position: they must be part of a *Hopf bundle*. The circles of a Hopf bundle can be mapped one-to-one to points on the 2-sphere. We can thus use pruning and condensing procedures for a 2-sphere in three dimensions. This yields a small set $\mathcal{P}$ of at most 12 great circles. Then we can apply the 2+2 dimension reduction technique. The details are actually more complicated, since we might have a phase in which $\mathcal{P}$ is successively condensed, see Section 6.

This concludes the summary of the algorithm. The steps of the algorithm involve several different operations: We need closest-pair graphs in 4 and 6 dimensions. The closest-pair

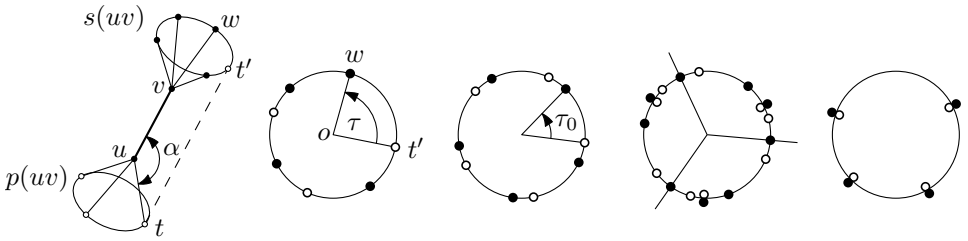


Figure 2 (a) Predecessors and successors of uv at angle α; t' is the reflected copy of t on the successor circle. (b) The corresponding predecessor-successor figure, and the torsion angle $\tau(tuvw)$. Predecessors are drawn white and successors black. (c) An edge-transitive predecessor-successor figure with torsion angle τ_0. (d) Canonical axes. (e) Mirror symmetry.

graph of k points can be calculated in $O(k \log k)$ time in any fixed dimension, by a classical divide-and-conquer approach [5]. We also need Voronoi regions in two dimensions and convex hulls in three dimensions. Finally, we need to sort lists of numbers lexicographically. In summary, we will be able to reduce the size of the current point set A' by a constant factor less than $1/2$, in $O(|A'| \log |A'|)$ time, until dimension reduction is possible.

3 Iterative Pruning Based on the Closest-Pair Graph: Algorithm C

After we construct the closest-pair graph G for a point set A on the 3-sphere, we try to condense it. We first explain the notion of a *predecessor-successor figure* in G.

We have a directed edge uv of G together with a set of *predecessor edges* $p(uv)$ incident to u and a set of *successor edges* $s(uv)$ incident to v, as in Fig. 2a. All edges have the same length, their endpoints lie on the 3-sphere $\mathbb{S}^3$, and all predecessor and successor edges form the same angle α with uv. Then the endpoints of these edges lie on two circles. If we reflect the predecessor circle at the bisecting hyperplane of uv, it comes to lie on the successor circle. This results in one circle with a succinct representation of the geometric situation. see Fig. 2a. We refer to the endpoints of the predecessor and successor edges as *predecessors* and *successors*.

The following algorithm will successively prune A until the closest distance δ is greater than δ_0, or it will exit to either Algorithm M or Algorithm O.

Algorithm C (Prune the closest-pair graph). We are given a set $A \subset \mathbb{R}^4$ of n points, equidistant from the origin. This set may already be the result of some previous condensing steps. Without loss of generality, we may assume that A lies on the unit sphere $\mathbb{S}^3$.

C1. [Well-separated points?] Compute the closest distance δ between points of A. If $\delta > \delta_0 = 0.0005$, apply 1+3 dimension reduction. ($|A|$ is bounded by a constant.)

C2. [Construct the closest-pair graph.] Construct the closest-pair graph G, and initialize the directed graph D with two opposite arcs for every edge of G. (This takes $O(n \log n)$ time, and the degrees in G are bounded by $K_3 = 12$.)

C3. [Prune by degree.] If the indegrees and outdegrees in D are not all the same, prune the vertices by degree, and return to C1, with the smallest class A' taking the role of A. (Otherwise, we now enter a loop in which we try to prune arcs from D.)

C4. [Prune by directed edge figure.] The *directed edge figure* of an arc $uv \in D$ consists of uv together with all arcs of D out of v and all arcs of D that lead into u. If the directed

edge figures are not all congruent, *prune the arcs of D* by congruence type of directed edge figures, and return to C3. (Here we apply the pruning principle not to points but to arcs and replace the edge set D by a smaller subset D'. Since the degrees are bounded, we can compare two directed edge figures in constant time.)

C5. [Mirror symmetry?] (Now all arcs have the same directed edge figure.) If the direct edge figure of uv is symmetric with respect to the bisecting hyperplane of uv, proceed to Algorithm R (Section 5).

C6. [Choose an angle α with no mirror symmetry.] Pick an angle α for which the predecessors and successors are not completely symmetric, that is, the predecessor-successor figure does not look like Figure 2e.

C7. [Initialize Successors.] For every arc $uv \in D$, set $s(uv) := \{vw : vw \in D, \angle uvw = \alpha\}$. (The size of $s(uv)$ is bounded by $K_2 = 5$. We will now enter an inner loop in which we try to prune arcs from the sets $s(uv)$.)

C8. [Update predecessors.] Define the predecessor edges by $p(uv) := \{\, tu : uv \in s(tu) \,\}$. Build the predecessor-successor figure for each arc, as explained in the text above.

C9. [Prune by predecessor-successor figures.] If there are arcs whose predecessor-successor figures are not congruent, prune the arcs of D accordingly, and return to C3.

C10. [Check regularity.] (Now all arcs have the same predecessor-successor figure. Each figure must contain the same number k of predecessors and successors, since the total number of predecessors in all figures must balance the total number of successors.) If the predecessor-successor figure consists of two regular k-gons, proceed to Algorithm O for generating orbit cycles, see Section 4. (We call this the *edge-transitive* case. See Figure 2c.)

C11. [Prune successors by canonical axes.] Prune $s(uv)$ to a proper nonempty subset by computing *canonical axes* as explained below, and return to C8.

The canonical-axes construction in Step C11 is a well-known procedure for detecting the rotational symmetries in a planar point configuration [12, 3]. We encode the points on a circle as a cyclic string alternating between k angular distances and k color labels. By standard string-processing techniques, we find the lexicographically smallest cyclic reordering of this string. The starting point of this string, together with the cyclic shifts which yield the same string, gives rise to a set of p equidistant rays starting from the origin (the *canonical axes*) as in Fig. 2d. These axes have the same p rotational symmetries as the original configuration. We know that $p < k$ because the maximally symmetric case of two regular k-gons (the *edge-transitive* case shown in Fig. 2c) has been excluded in Step C10. The loop from C8–C11 maintains the following loop invariant on entry to Step C10:

There is a position of a successor that is not occupied by a predecessor. (1)

For the reduction in Step C11, we rotate the canonical axes counterclockwise until they hit the first position of type (1). The successors that are intersected by the canonical axes form a nonempty proper subset $s' \subseteq s(uv)$. We thus replace $s(uv)$ for each edge uv by s' and return to Step C8. By construction, we have made sure that (1) still holds. After pruning all successor sets, the predecessor sets are reduced accordingly in Step C8, but this cannot invalidate (1). (The invariant (1) holds on first entry to the loop because of Step C6.)

The algorithm has three nested loops (indicated by indentation) and works its way up to higher and higher orders of regularity. After C3, all vertices have the same degree. After C4, we know that all *pairs* of adjacent vertices look the same. If we exit to Algorithm O in Step C10, wee will see that certain chains of *four* points can be found everywhere.

There is the *global loop* that leads back to C1 after each successful pruning of vertices by degree. Since the size of A is reduced to less than a half, we need not count the iterations of this loop. In addition, there is an outer loop that resumes working at C3 after pruning the edges of D, and an inner loop that starts at C8 and is repeated whenever the successor set $s(uv)$ is pruned. In these loops, we maintain that $D \neq \emptyset$ and $s(uv) \neq \emptyset$. In Step C3, if we have removed at least one edge from D, we will either be able to prune by degree, or the degree of all vertices has gone down by at least one. Since the degree is initially bounded by 12, Step C3 can be visited at most 12 times before exiting to C1. Similarly, Step C11 removes at least one element of $s(uv)$, so this loop is repeated at most 5 times before there is an exit to the outer loop in step C9. The most time-consuming step is the construction of the closest-pair graph in Step C2. All other operations take $O(n)$ time, not counting the exits to Algorithms M and O. Thus, the overall time is $O(|A| \log |A|)$.

4 Generating Orbit-Cycles: Algorithm O

We now describe how, in the edge-transitive case, the algorithm produces a set $\mathcal{P}$ of at most $|A|/200$ great circles. All predecessor-successor figures look like Fig. 2c. The *torsion angle* $\tau(tuvw)$ between a predecessor edge $tu \in p(uv)$ and a successor edge $vw \in s(uv)$ is the oriented angle $\angle(t'ow)$ in the predecessor-successor figure of uv. We define τ_0 as the smallest counterclockwise torsion angle that appears in the predecessor-successor figure. Let $t_0u_0v_0w_0$ be a fixed quadruple with this torsion angle.

▶ **Lemma 2.**

1. *For every triple $a_1a_2a_3$ with $a_2a_3 \in s(a_1a_2)$, there is a unique cyclic sequence $a_1a_2 \dots a_\ell$ such that $a_ia_{i+1}a_{i+2}a_{i+3}$ is congruent to $t_0u_0v_0w_0$ for all i. (Indices are taken modulo ℓ.)*
2. *Moreover, there is a unique rotation matrix R such that $a_{i+1} = Ra_i$. In other words, $a_1a_2 \dots a_\ell$ is the orbit of a_1 under the rotation R.*
3. *The points $a_1a_2 \dots a_\ell$ do not lie on a circle.*

Proof. We give only a sketch; the full proof is written out in [11, Lemma 17]. It is a straightforward sequence of arguments, once we have established that the three points t_0, u_0, v_0 cannot lie on a great circle, and therefore a rotation is uniquely determined by the images of these three points. If the points t_0, u_0, v_0 would lie on a great circle, this would imply a mirror symmetry in the predecessor-successor figure, contradicting the filtering steps C5 and C6 which have led to the invariant (1). ◀

We call the cyclic paths that are constructed in Lemma 2 *orbit cycles*. The following lemma gives a bound on the number of orbit cycles in terms of $|A|$ when the closest-pair distance is small enough.

▶ **Lemma 3.** *The number of orbit cycles is at most $|A|/200$ provided that the closest distance δ is smaller than $\delta_0 = 0.0005$.*

Proof. We have $a_{i+1} = R_{\varphi\psi}a_i$ for all i, with a rotation matrix that can be written as

$$R_{\varphi\psi} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & \cos\psi & -\sin\psi \\ 0 & 0 & \sin\psi & \cos\psi \end{pmatrix} \tag{2}$$

in an appropriate basis $x_1y_1x_2y_2$. We cannot have $\varphi = 0$ or $\psi = 0$, because otherwise the orbit would form a regular polygon $a_1a_2a_3a_4 \dots a_\ell$ in a plane, contradicting Lemma 2.3.

Thus, we know that $\varphi, \psi \neq 0$. To get a closed loop, we must have $|\varphi|, |\psi| \geq 2\pi/\ell$. If the projection of a_i to the x_1y_1-plane has norm r_1 and the projection to the x_2y_2-plane has norm r_2 with $r_1^2 + r_2^2 = 1$, then the squared distance is

$$\delta^2 = \|a_{i+1} - a_i\|^2 = (2r_1 \sin \tfrac{|\varphi|}{2})^2 + (2r_2 \sin \tfrac{|\psi|}{2})^2 \geq 4 \sin^2(\pi/\ell).$$

Thus, if $\delta \leq 2\sin(\pi/12000) \approx 0.000523$, it is guaranteed that $\ell \geq 12000$, that is, every orbit cycle contains at least 12000 points. On the other hand, each point $u \in A$ belongs to a bounded number of orbit cycles: it has at most $K_3 = 12$ incoming arcs tu, and each each arc tu has at most $K_2 = 5$ successor edges $uv \in s(tu)$. The triple tuv specifies a unique orbit cycle, and thus there are at most $12 \times 5 = 60$ orbit cycles through u. The total number of orbit cycles is therefore at most $|A| \cdot 60/12000 = |A|/200$. ◀

For each orbit cycle, we can find an appropriate rotation matrix $R_{\varphi,\psi}$. If $\varphi = \pm\psi$, then the orbit of any point under this rotation lies on a great circle, contradicting Lemma 2.3. Thus, we get a unique invariant plane that rotates by the smaller angle in itself. We replace each orbit cycle by the great circle in this invariant plane, yielding the desired set $\mathcal{P}$ of great circles with $|\mathcal{P}| \leq |A|/200$.

5 The Mirror Case: Algorithm R

Algorithm R (Treat mirror-symmetric closest-pair graphs). We are given a nonempty directed subgraph D of the closest-pair graph on a point set $A \subset \mathbb{S}^3$ with closest-pair distance $\delta \leq \delta_0$. All directed edges in D have equal edge-figures and exhibit perfect mirror-symmetry, as established in Steps C4 and C5 of Algorithm C. Algorithm R will produce, in an equivariant way, either

(a) a set A' of at most $|A|/2$ points, or
(b) a set $\mathcal{P}$ of at most $|A|/200$ great circles.

R1. [Make D undirected.] Construct the undirected version of D and call it G.
R2. [Check for eccentric centers of mass.] Compute the center of mass of each connected component of G. If these centers are not in the origin, return the set A' of these centers.
R3. [Two-dimensional components?] If each component is a regular polygon with center at the origin, return the set $\mathcal{P}$ of circumcircles of each polygon.
R4. [Three-dimensional components?] If each component spans a 3-dimensional hyperplane H, replace the component by two antipodal points perpendicular to H. Return the set A' of these points.
R5. [Toroidal grid.] Now each component of D is the product $P \times Q$ of a regular p-gon P and a regular q-gon Q in orthogonal planes, with $p, q \geq 3$. Pick a vertex u from each component. There are four incident edges, and the plane spanned by two incident edges is orthogonal to the plane spanned by the other two edges. Represent the component of G by these two orthogonal planes shifted to the origin. Return the set $\mathcal{P}$ of great circles in these planes (two per component).

The algorithm takes linear time. We still need to show that these cases are exhaustive. After we make D undirected in Step R1, for every edge uv of G, the bisecting hyperplane of uv acts as a mirror that exchanges u and v together with their neighborhoods. Since the reflected mirror hyperplanes act again as mirrors, it follows that every component of the graph is the orbit of some point u_0 under the group generated by reflections perpendicular to the edges incident to u_0. Thus, the incident edges of a single point determine the geometry of the whole component. The graph may have several components, all of which are congruent.

The discrete groups generated by reflections (finite Coxeter groups) have been classified in any dimension, cf. [7, Table IV on p. 297]. In four dimensions, they were first enumerated by Édouard Goursat in 1899: There are five *irreducible groups*, which are the symmetry groups of the five regular 4-dimensional polytopes, and the *reducible* groups, which are a direct product of lower-dimensional reflection groups. There are infinitely-many reducible groups of the form $D_2^p \times D_2^q$, where D_2^p is the dihedral group of order $2p$, the symmetry group of the regular p-gon. These groups are treated in Step R5. Except this infinite family, there are only eight other groups, which can be excluded by the following lemma.

▶ **Lemma 4.** *If every component of G is the orbit of a point under a four-dimensional symmetry group which is not $D_2^p \times D_2^q$, then the minimum distance δ is at least* 0.07*, and therefore bigger than δ_0.*

Proof. We analyze each of the eight groups case by case. The arrangement of all mirror hyperplanes of a reflection group cuts the 3-sphere into equal cells, which can be taken as the *fundamental regions* of the group. (These fundamental regions are not necessarily equal to the Voronoi regions of the point set; the Voronoi regions are usually cut into smaller cells by mirrors passing through the centers of the regions.) It is known that the fundamental region of a reflection group is a spherical *simplex*, by a theorem of Cartan (1928), see [7, Theorem 11.23]. Thus, in 4-space, the fundamental region is a spherical tetrahedron T. We must now consider all possibilities how the orbit of a point $u_0 \in T$ might give rise to a component of G. We have to place u_0 on some facets F of T and equidistant from the remaining facets of T. If the point is not chosen equidistant from the other faces, the closest-pair graph would not contain edges that generate all four mirrors. The minimum distance is achieved when u_0 lies in the interior of T and $F = \emptyset$. This value is approximately 0.0782. See [11, Section 10.1] for more details. ◀

We go through the steps one by one to check if the algorithm achieves the claimed results. Step R2 treats the case when a (lower-dimensional) component does not go through the origin. This includes the cases when G is a matching or a union of "small" regular polygons. Every component contains at least 2 points, and thus $|A'| \le |A|/2$. Steps R3 and R4 treat the two- and three-dimensional components that are centered at the origin. (The one-dimensional case of a matching cannot be centered at the origin, because then we would have $\delta = 2$.) If we have a regular k-gon inscribed in a great circle (Step R3), from $\delta < \delta_0$, $k \ge 12000$. Thus $|C| \le |A|/12000 \le |A|/200$. A three-dimensional component (Step R4) contains at least four points and is reduced to two antipodal points. Again we have $|A'| \le |A|/2$.

Steps R2–R4 have treated all cases of Coxeter groups which are not full-dimensional, and Lemma 4 excludes all full-dimensional groups which are not of the form $D_2^p \times D_2^q$. Thus, in Step R5, each component of D is the product $P \times Q$ of a regular p-gon P and a regular q-gon Q in orthogonal planes, with $p, q \ge 3$. Such a component forms a toroidal $p \times q$ grid. Each vertex has four neighbors. The two polygons P and Q have equal side lengths δ, because otherwise the four neighbors would not be part of the closest-pair graph G. The two incident edges of a vertex u that come from P are orthogonal to the two edges that come from Q, so we can distinguish the two edge classes. (The case when all four edges are perpendicular is the 4-cube with $p = q = 4$ and with reflection group $D_2^4 \times D_2^4$, which has 16 vertices and minimum distance $\delta = 1$ and is therefore excluded.) All copies of P in the grid lie in parallel planes, and so do the copies of Q. Accordingly, Step R5 represents each connected component by two orthogonal planes through the origin. We need to show that the component is large. As $P \times Q$ lies on the unit 3-sphere, the circumradii r_P and r_Q of the two polygons satisfies $r_P^2 + r_Q^2 = 1$. Thus, the larger circumradius, let us say r_P, is at least $1/\sqrt{2}$. Since the

closest-pair distance is $\delta = 2r_P \sin\frac{\pi}{p} \leq \delta_0 = 0.0005$, we get $\sqrt{2}\sin\frac{\pi}{p} \leq \delta_0$, which implies $p \geq 8886$. Each component contains $pq \geq p$ points and is reduced to two circles. Thus, the algorithm achieves the claimed reduction.

6 Marking and Condensing of Great Circles: Algorithm M

We have extracted a set $\mathcal{P}$ of at most $|A|/200$ great circles from the point set A, either from the mirror case (Algorithm R in Section 5) or from orbit cycles (Algorithm O in Section 4). Algorithm M obtains a small set A' of *marker points* on each circle so that we can resume Algorithm C, or it exits to Algorithm T for 2+2 dimension reduction (Section 7). For this purpose, we look at pairs of circles $C, D \in \mathcal{P}$ and the angle between the 2-planes P, Q in which they lie. The angle of two planes P, Q in 4-space consists of *two* numbers α, β, with $0 \leq \alpha, \beta \leq \pi/2$ [13]. One way to define the angle is by the orthogonal projection C' of the unit circle C in plane P onto Q. This projection is an ellipse, and its axes are $\cos\alpha$ and $\cos\beta$. If $\alpha \neq \beta$, we can use the major axis of the projection ellipse C' to mark the two points of D on this axis, which are closest points on D to C. We similarly mark two points on C.

Doing this for all pairs of circles would lead to a quadratic blowup. Therefore we construct the closest-pair graph on the *set of circles*. We use Plücker coordinates to map great circles of $\mathbb{S}^3$ to points on $\mathbb{S}^5$. We can then compute the closest-pair graph in six dimensions, and every circle has at most K_5 closest neighbors.

For a plane P spanned by two vectors u, v, Plücker coordinates are the sixtuples of the 2×2 determinants of the 2×4 matrix with rows u and v. Plücker coordinates are usually regarded as points of projective 5-space: they are unique up to scaling. We thus normalize them, and represent every plane as a pair of antipodal points on the sphere $\mathbb{S}^5$. We define the *Plücker distance* between two planes as the smallest Euclidean distance among any two of the four representative points in $\mathbb{R}^6$. The following lemma shows that this distance is geometrically meaningful and does not depend on the choice of a coordinate system.

▶ **Lemma 5.**
1. *Two planes P, Q with angles α, β have Plücker distance $\sqrt{2(1 - \cos\alpha\cos\beta)}$.*
2. *A circle can have at most K_5 closest neighbors on the Plücker sphere $\mathbb{S}^5$.*

Proof. Property 1 is a lengthy calculation [11, Lemma 4]. Property 2 is straightforward. ◀

The above approach fails for planes with $\alpha = \beta$. Such planes or circles are called *isoclinic*. They come in two variations, left-isoclinic and right-isoclinic, which are distinguished as follows: Let v_1, v_2 be a basis of P, and let v_1', v_2' be the projection of v_1, v_2 to Q. Then P and Q are right-isoclinic if the vectors v_1, v_2, v_1', v_2' form a positively oriented basis of $\mathbb{R}^4$. This classification as left or right is called the *chirality* of a pair of isoclinic planes. When $\alpha = \beta = \pi/2$ (completely orthogonal planes) or $\alpha = \beta = 0$ (identical planes), the two planes are both left-isoclinic and right-isoclinic.

Isoclinic pairs manifest a very rigid structure on a sphere. The following proposition summarizes some beautiful properties of such pairs. They are formulated for right-isoclinic circles, but they hold equally with left and right exchanged.

▶ **Proposition 6.**
1. *The relation of being right-isoclinic is transitive (as well as reflexive and symmetric). An equivalence class is called a* right Hopf bundle.
2. *For each right Hopf bundle, there is a* right Hopf map *h that maps the circles of this bundle to points on $\mathbb{S}^2$.*

3. *By this map, two isoclinic circles with angle α, α (and with Euclidean distance $\sqrt{2}\sin\alpha$ on the Plücker sphere $\mathbb{S}^5$) are mapped to points at angular distance 2α on $\mathbb{S}^2$.*
4. *A circle can have at most $K_2 = 5$ closest neighbors on the Plücker sphere $\mathbb{S}^5$ that are right-isoclinic.*

Proof Sketch. Transitivity (Property 1) and the preservation of distances (Property 3) can be established by calculations, see [11, Corollary 11 and Lemma 12]. The right Hopf map in Property 2 is obtained as follows: Choose a positively oriented coordinate system (x_1, y_1, x_2, y_2) for which some circle C_0 of the bundle lies in the x_1y_1-plane. Then the map $h\colon \mathbb{S}^3 \to \mathbb{S}^2$ defined by $h(x_1, y_1, x_2, y_2) = \big(2(x_1y_2 - y_1x_2),\ 2(x_1x_2 + y_1y_2),\ 1 - 2(x_2^2 + y_2^2)\big)$ maps all points on a circle of the bundle to the same point on $\mathbb{S}^2$. A different choice of C_0 would lead to a different map, but by Property 3, the images are related by an isometry of $\mathbb{S}^2$. Property 4 is a direct consequence of Properties 2 and 3. ◀

We need an auxiliary Algorithm K, defined later, for condensing on the 2-sphere:

▶ **Lemma 7.** *Algorithm K condenses a nonempty set F of points on the 2-sphere $\mathbb{S}^2$ in an equivariant way to a nonempty set F' of at most $\min\{12, |F|\}$ representative points on $\mathbb{S}^2$, in $O(|F| \log |F|)$ time. These points are either the vertices of a regular tetrahedron, a regular octahedron, a regular icosahedron, a single point, or a pair of antipodal points.*

Algorithm M (Mark and condense great circles). Given a set $\mathcal{P}$ of great circles on $\mathbb{S}^3$, this algorithm will produce, in an equivariant way, either a nonempty set A' of at most $100|\mathcal{P}|$ points on $\mathbb{S}^3$, or a nonempty set $\mathcal{P}'$ of at most 829 great circles.

The algorithm will update the set $\mathcal{P}$ and maintain an equivalence relation $\sim$ on $\mathcal{P}$ such that all circles in the same equivalence class belong to a common (left or right) Hopf bundle. The common chirality of all bundles is indicated by a variable *chirality* $\in \{None, Left, Right\}$, where *None* is chosen when the equivalence relation is trivial and all classes are singletons. The size of the equivalence classes is bounded by 12.

M1. [Initialize.] Let every circle $C \in \mathcal{P}$ form a singleton equivalence class.
M2. [Prune by size.] If equivalence classes are of different sizes, choose the size that occurs least frequently. Throw away all equivalence classes that are not of this size, together with the circles they contain.
M3. [Few circles?] If $|\mathcal{P}| \le 829$, return the set $\mathcal{P}$.
M4. [Singletons?] If all equivalence classes are singletons, set *chirality* := *None*.
M5. [Construct closest pairs.] Represent each circle $C \in \mathcal{P}$ by two antipodal points on $\mathbb{S}^5$, using Plücker coordinates. Construct the closest-pair graph H on $\mathcal{P}$ with respect to the distances on $\mathbb{S}^5$.
M6. [Classify edges.] Partition the edges of H into $E_L \cup E_R \cup E_N$, representing pairs of circles that are left-isoclinic, right-isoclinic, and not isoclinic.
M7. [Use non-isoclinic edges.] If $E_N \neq \emptyset$, let $\mathcal{N} := E_N$ and go to Step M12.
M8. [Find non-isoclinic pairs from equivalent circles.] If $E_L \neq \emptyset$ and *chirality* = *Right*, set $\mathcal{N} := \big\{\{C', D\} \mid \{C, D\} \in E_L,\ C' \text{ is closest to } C \text{ among the circles } C' \sim C\ \big\}$, and go to Step M12.
M9. (Same as M8, with left and right exchanged.)
M10. [Merge classes.] If $E_L \neq \emptyset$ and *chirality* $\in \{Left, None\}$, merge equivalence classes that contain circles that are connected in E_L. Use Algorithm K to condense each resulting class F to a set F' of at most 12 *representative* circles. Set $\mathcal{P}$ to the set of all representatives

circles, and put them into the same equivalence class if and only if they come from the same set F'. Set *chirality* := *Left*, and return to M2.

M11. (Same as M10, with left and right exchanged. Since all possibilities are exhausted, the only remaining possibility in this step is to merge classes and return to M2.)

M12. [Mark points on circles.] (Each pair in $\mathcal{N}$ is now a non-isoclinic pair of circles.) For each pair of circles $\{C, D\} \in \mathcal{N}$, mark the two points on C that are closest to D, and likewise, mark two points on D. Return the set A' of all marked points.

The crucial properties for the correctness and the running time are formulated in a lemma, implying that the algorithm produces indeed at most $100|\mathcal{P}|$ points, 4 for each pair in $\mathcal{N}$.

▶ **Lemma 8.**

1. *After the algorithm returns to Step M2 from step M10 or M11, the number of equivalence classes is reduced at least by half.*
2. *Algorithm M terminates in $O(|\mathcal{P}| \log |\mathcal{P}|)$ time.*
3. *In Step M12, $\mathcal{N}$ is a nonempty set of at most $25|\mathcal{P}|$ non-isoclinic pairs.*

Proof.

1. The only possibility for the algorithm to stall is that the edges of H are *within* classes, and no merging takes place in Step M10 or M11. Each class must be one of the five configurations listed in Lemma 7, and the smallest possible angular distance $\delta_{\text{ico}} \approx 1.107$ occurs for two adjacent vertices of the icosahedron. Translating this to the distance on $\mathbb{S}^5$ by Lemma 3, we get that the closest-pair distance is at least $\delta_{\min} \approx 0.7435$, and a volume packing argument on $\mathbb{S}^5$ yields that there can be at most 829 circles with this distance, see [11, Section 11.3]. Then the algorithm exits in Step M3.
2. The most expensive step is the construction of the closest-pair graph in Step M5, which takes $O(|\mathcal{P}| \log |\mathcal{P}|)$ time. The algorithm contains one loop, when returning from M10 or M11 to M2. Since the number of equivalence classes decreases geometrically and each class contains at most 12 circles, the overall time is also bounded by $O(|\mathcal{P}| \log |\mathcal{P}|)$.
3. When $\mathcal{N}$ is constructed in Step M7, there can be at most $22|\mathcal{P}|$ such pairs, because the degree in H is bounded by $K_5 \leq 44$. In Step M8, the constructed set $\mathcal{N}$ is nonempty: Every pair $\{C, D\} \in E_L$ produces at least one element of $\mathcal{N}$, since all equivalence classes contain at least two elements, by M2 and M4. When a pair $\{C', D\}$ in $\mathcal{N}$ is formed, we have a left-isoclinic pair $\{C, D\}$ and a right-isoclinic pair $\{C, C'\}$. If $\{C', D\}$ were also isoclinic, we would get a contradiction to transitivity (Lemma 1). Each circle $C \in \mathcal{P}$ gives rise to at most $5 \cdot 5 = 25$ pairs, since a circle can have at most 5 isoclinic neighbors in H by Lemma 4. ◀

Constructing a Canonical Set on the 2-Sphere: Algorithm K. This algorithm is a variant of a standard 3-dimensional congruence testing algorithm [4]. It condenses a set $F \subset \mathbb{S}^2$ to a nonempty set $F' \subset \mathbb{S}^2$ of at most $\min\{12, |F|\}$ representative points, in $O(|F| \log |F|)$ time.

We start by computing the convex hull of F. If it does not contain the origin, we output the vector pointing to the centroid of the points as a representative. If the hull is one- or two-dimensional, we get two antipodal points as canonical representatives. We can thus assume that the hull is a three-dimensional polytope. Now we prune by vertex degree, prune and condense by face degree (replacing the point set by the centroids of some faces), and prune and condense by edge length (replacing the point set by the midpoints of some edges). This is repeated until the vertices form a regular tetrahedron, octahedron, or icosahedron. The details are given in [11, Section 20]. The most expensive part is the computation of the convex hull, and each condensing reduces the size by a constant factor. Thus, the overall running time is $O(|F| \log |F|)$.

7 2+2 Dimension Reduction: Algorithm T

▶ **Theorem 9.** *Given two sets A, B of n points, and two planes P and Q, it can be checked in $O(n \log n)$ time if there exists a rotation that maps A to B, and P to Q.*

Proof. We begin with choosing a coordinate system (x_1, y_1, x_2, y_2) for A so that P becomes the x_1y_1-plane, and similarly for B and Q. We then look for rotations R that leave the x_1y_1-plane invariant. Such rotations have the form $R = \left(\begin{smallmatrix} R_1 & 0 \\ 0 & R_2 \end{smallmatrix}\right)$ with two orthogonal 2×2 matrices R_1 and R_2. Since $\det R = \det R_1 \cdot \det R_2 = 1$, we try two cases: (a) R_1 and R_2 are planar rotations; (b) R_1 and R_2 are planar reflections. We can reduce (b) to (a) by applying the rotation $\left(\begin{smallmatrix} 0&1&0&0 \\ 1&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{smallmatrix}\right)$ to A. Thus, it suffices to describe (a), where R has the form $R_{\varphi\psi}$ in (2), i. e., a combination of a rotation by φ in the x_1y_1-plane and an independent rotation by ψ in the x_2y_2-plane. We use polar coordinates in these two planes by setting

$$\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} r_1 \cos\alpha_1 \\ r_1 \sin\alpha_1 \\ r_2 \cos\alpha_2 \\ r_2 \sin\alpha_2 \end{pmatrix} \qquad \text{with } r_1 = \sqrt{x_1^2 + y_1^2} \text{ and } r_2 = \sqrt{x_2^2 + y_2^2}.$$

The rotation $R_{\varphi\psi}$ changes (α_1, α_2) by adding (φ, ψ) modulo 2π, and leaves r_1 and r_2 unchanged. In other words, $R_{\varphi\psi}$ acts as a translation on the torus $\mathbb{T}^2 = [0, 2\pi) \times [0, 2\pi)$. We attach the distances (r_1, r_2) to each point $(\alpha_1, \alpha_2) \in \mathbb{T}^2$ as a *label*. The problem becomes therefore a *two-dimensional problem* of testing *congruence under translation* for *labeled* points on a torus. We denote the two labeled point sets as $\tilde{A}$ and $\tilde{B}$, and a point of $\tilde{A}$ can only be mapped to a point of $\tilde{B}$ with the same label.

Points in the x_1y_1-plane should be considered separately, because α_2 is not unique when $r_2 = 0$, and the same problem occurs for the points in the x_2y_2-plane. We will defer the treatment of these points to the end of this section, and start with other points first.

We now give an algorithm for the following problem: given two labeled point sets $\tilde{A}$ and $\tilde{B}$ on the torus $\mathbb{T}^2$, test if $\tilde{A}$ and $\tilde{B}$ are the same up to translations. We will find a canonical set of $\tilde{A}$ (and $\tilde{B}$) which is similar to a condensed set. In contrast to a condensed set, we add no new symmetries to a canonical set, by updating labels to preserve complete information. Let $\mathrm{Sym}(A)$ for a set $A \subset \mathbb{T}^2$ denote translational symmetry group of A, i.e., the set of translations that map A to itself and preserve labels, if A is a labeled set.

Algorithm T (Reduce a labeled point set on the torus to a canonical set). The input is a labeled point set $\tilde{A}$ on the torus $\mathbb{T}^2$. We assume that two labels can be compared in constant time. The output is an unlabeled *canonical set* $\hat{A}$ with the following two properties:

(a) $\mathrm{Sym}\,\hat{A} = \mathrm{Sym}\,\tilde{A}$.

(b) The group $\mathrm{Sym}\,\hat{A}$ acts transitively on $\hat{A}$: For any two points $x, y \in \hat{A}$, the translation from x to y leaves A invariant.

In addition, $\hat{A}$ should be obtained from $\tilde{A}$ without making any arbitrary decisions.

T1. [Prune by labels.] Choose the label that occurs least frequently in $\tilde{A}$, and let A' be the set of points with this label. (For a while we will now do ordinary pruning, using only the geometry of the point set A'.)

T2. [Compute the Voronoi diagram.] Compute the Voronoi diagram of A' on the torus $\mathbb{T}^2$. This can be reduced to a Voronoi diagram in the plane by replicating the square region representing the torus together with the set A' 9 times in a 3×3 tiling, see for

example [9]. Clipping the result to the central tile yields the Voronoi diagram on the torus in $O(|A'| \log |A'|)$ total time.

T3. [Prune by shape.] Translate each point $a \in A'$ to the origin together with its Voronoi cell. If the translated cells are not all equal, replace A' by the subset of points whose cell shape occurs least frequently, and return to T2.

T4. [Restore information from the original set $\tilde{A}$ by labeling the points in A'.] (Now all Voronoi cells are translated copies of the same hexagon or rectangle.) Assign each point of $\tilde{A}$ to its Voronoi cell. A point on the boundary is assigned to all incident cells. Now for each Voronoi site $x \in A'$, collect the points in its cell and translate them so that x lies at the origin. Represent each point as a triple (φ-coordinate, ψ-coordinate, label). Concatenate these triples in lexicographic order into a string of numbers that represents the cell contents, and attach the string of each cell as a label to the point x. (This string representation is obtained equivariantly, since two points $x, y \in A'$ get the same string if and only if the two Voronoi cells are exact translated copies of each other, including all points in the cells with their original labels. We have thus preserved complete information about the symmetries of $\tilde{A}$.)

T5. [Compress labels.] Sort the label strings and replace each label with its rank in sorted order.

T6. [Finalize.] If there are at least two labels, return to T1. Otherwise, return the set of points A', without labels, as the canonical set $\hat{A}$.

▶ **Lemma 10.** *Algorithm T computes a canonical set $\hat{A}$ of a labeled set $\tilde{A}$ on the torus in time $O(|\tilde{A}| \log |\tilde{A}|)$.*

Proof. We first check that $\hat{A}$ has the claimed properties. Property (a) consists of two inclusions: We have $\mathrm{Sym}(\tilde{A}) \subseteq \mathrm{Sym}(\hat{A})$, because $\hat{A}$ is obtained in a equivariant way from $\tilde{A}$. None of the operations which are applied to $\tilde{A}$ to obtain $\hat{A}$ destroys any translational symmetries. The other inclusion $\mathrm{Sym}(\hat{A}) \subseteq \mathrm{Sym}(\tilde{A})$ is ensured by Step T4. (This would work for any set A'.) To see property (b), note that the Voronoi cell of a point fixes the relative positions of its neighbors. Starting from any point $a \in \hat{A}$ we can reconstruct the whole set $\hat{A}$ if we know the shape of each Voronoi cell. Step T3 ensures that all Voronoi cells are equal, and therefore the reconstructed set $\hat{A} - a$ is the same, no matter from which point a we start.

Let us analyze the running time. Each iteration of the loop T2–T3 takes $O(|A'| \log |A'|)$ time and reduces the size of A' to half or less. Thus, the total running time of this loop is $O(|\tilde{A}| \log |\tilde{A}|)$, since the initial size of A' is bounded by the size of $\tilde{A}$.

Steps T4 and T5 involve point location in Voronoi diagrams and sorting of strings of numbers of total length $O(|\tilde{A}|)$. These operations can be carried out in $O(|\tilde{A}| \log |\tilde{A}|)$ time. Thus, each iteration of the whole loop T1–T6 takes $O(|\tilde{A}| \log |\tilde{A}|)$ time. Step T1 reduces the size of $\tilde{A}$ to a half or less after each iteration. Thus, the total running time is $O(|\tilde{A}| \log |\tilde{A}|)$. ◀

We use this procedure for comparing two labeled sets $\tilde{A}$ and $\tilde{B}$ on the torus as follows. We run Algorithm T in parallel for $\tilde{A}$ and $\tilde{B}$. We must make sure that all steps run identically. In particular, the sorted strings in Step T5 must be the same for $\tilde{A}$ and $\tilde{B}$. If the algorithm runs to the end without finding a difference between $\tilde{A}$ and $\tilde{B}$, and if the Voronoi cells of the canonical sets $\hat{A}$ and $\hat{B}$ are equal, we know that $\tilde{A}$ and $\tilde{B}$ are congruent by translation.

We still have to deal with points on coordinate planes. Let $A_1 \subseteq A$ be the points in the x_1y_1-plane, and let $A_2 \subseteq A$ be the points in the x_2y_2-plane. If $A_1 \neq \emptyset$, we compute the canonical axes of A_1, as described for Step C11 in Section 3. They form $k \geq 1$ equally spaced

angular directions $\bar{\alpha} + j\frac{2\pi}{k}$ modulo 2π, $j \in \mathbb{Z}$. We know that R must map the canonical axes of A_1 to the canonical axes for the corresponding set B_1. We incorporate this restriction into an additional label for each point $u \in A \setminus A_1 \setminus A_2$. If u has a polar angle α_1 in the x_1y_1-plane, we attach to it the difference to the nearest smaller canonical angle:

$$\min\{\, \alpha_1 - (\bar{\alpha} + j\tfrac{2\pi}{k}) \mid j \in \mathbb{Z},\ \alpha_1 - (\bar{\alpha} + j\tfrac{2\pi}{k}) \geq 0 \,\}$$

We also have to test if A_1 and B_1 are congruent. If $A_2 \neq \emptyset$, we treat this in the same way and attach another additional label to the points in $A \setminus A_1 \setminus A_2$. (In fact, Algorithm T is an extension of the canonical axes construction from the one-dimensional torus $\mathbb{S}^1$ to the two-dimensional torus $\mathbb{T}^2$.) ◀

8 Concluding Remarks

The running times of the algorithms for general dimensions are improved by incorporating our algorithm when the dimension is reduced to 4. The deterministic algorithm of Brass and Knauer runs now in $O(n^{\lceil (d-1)/3 \rceil} \log n)$ time, and the randomized algorithm of Akutsu takes time $O(n^{\lfloor (d-1)/2 \rfloor/2} \log n)$ for $d \geq 9$ and $O(n^{3/2} \log n)$ for $d \leq 8$. It is likely that our algorithm can be simplified, and the constants in the bounds are certainly too pessimistic.

Approximate Congruence Testing. It makes sense to test for congruence with an error tolerance ε, but this problem is known to be NP-hard even in the plane [8, 10]. However, the problem becomes polynomial if the input points are sufficiently separated compared to ε. We are confident that our algorithm, when implemented with standard floating-point arithmetic and with appropriate error margins to shield equality tests from round-off errors, would decide approximate congruence in the following weak sense. Given a tolerance ε that is large compared to the machine precision, and two sets A and B whose points are separated by more than, say 10ε, the algorithm would correctly distinguish the instances that are congruent with a tolerance of ε from the instances that are not even congruent with a tolerance of, say 100ε. Between ε and 100ε, the algorithm is allowed to make errors. Such a result will require a thorough analysis of the numerical stability of the operations in our algorithm.

Regularity. The general theme in our algorithm is whether *local* regularity implies *global* regularity; in other words, when sufficiently large neighborhoods of all points look the same, does a symmetry group have to act transitively on the point set? Our techniques might also shed light on symmetries on the 3-sphere, for example the classification of finite subgroups of the orthogonal group $O(4)$ of 4×4 orthogonal matrices, or on regular and semiregular tilings of $\mathbb{S}^3$.

References

1. Tatsuya Akutsu. On determining the congruence of point sets in d dimensions. *Computational Geometry: Theory and Applications*, 4(9):247–256, 1998.
2. Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete & Computational Geometry*, 3(1):237–256, 1988.
3. M. J. Atallah. On symmetry detection. *IEEE Trans. Computers*, 100(7):663–666, 1985.
4. M. D. Atkinson. An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8(2):159–172, 1987.

5 Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *Proc. 8th Ann. ACM Symp. Theory of Computing (STOC)*, pages 220–230. ACM, 1976.

6 Peter Brass and Christian Knauer. Testing the congruence of d-dimensional point sets. *International Journal of Computational Geometry and Applications*, 12(1–2):115–124, 2002.

7 Harold S. M. Coxeter. *Regular Polytopes*. Dover Publications, 3rd edition, 1973.

8 C. Dieckmann. *Approximate Symmetries of Point Patterns*. PhD thesis, FU Berlin, 2012.

9 N. P. Dolbilin and D. H. Huson. Periodic Delone tilings. *Per. Math. Hung.*, 34:57–64, 1997.

10 Sebastian Iwanowski. Testing approximate symmetry in the plane is NP-hard. *Theoretical Computer Science*, 80(2):227–262, 1991.

11 H. Kim and G. Rote. Congruence testing of point sets in 4 dimensions. arXiv:1603.07269.

12 Glenn Manacher. An application of pattern matching to a problem in geometrical complexity. *Information Processing Letters*, 5(1):6–7, 1976.

13 Henry P. Manning. *Geometry of Four Dimensions*. Macmillan, 1914.

14 Kōkichi Sugihara. An $n \log n$ algorithm for determining the congruity of polyhedra. *Journal of Computer and System Sciences*, 29(1):36–47, 1984.

On the Complexity of Minimum-Link Path Problems*

Irina Kostitsyna[†1], Maarten Löffler[‡2], Valentin Polishchuk[§3], and Frank Staals[¶4]

1 **Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
i.kostitsyna@tue.nl
2 **Dept. of Computing and Information Sciences, Utrecht University, The Netherlands**
m.loffler@uu.nl
3 **Communications and Transport Systems, ITN, Linköping University, Sweden**
valentin.polishchuk@liu.se
4 **MADALGO, Aarhus University, Denmark**
f.staals@cs.au.dk

Abstract

We revisit the minimum-link path problem: Given a polyhedral domain and two points in it, connect the points by a polygonal path with minimum number of edges. We consider settings where the min-link path's vertices or edges can be restricted to lie on the boundary of the domain, or can be in its interior. Our results include bit complexity bounds, a novel general hardness construction, and a polynomial-time approximation scheme. We fully characterize the situation in 2D, and provide first results in dimensions 3 and higher for several versions of the problem.

Concretely, our results resolve several open problems. We prove that computing the minimum-link *diffuse reflection path*, motivated by ray tracing in computer graphics, is NP-hard, even for two-dimensional polygonal domains with holes. This has remained an open problem [16] despite a large body of work on the topic. We also resolve the open problem from [25] mentioned in the handbook [17] (see Chapter 27.5, Open problem 3) and The Open Problems Project [9] (see Problem 22): "What is the complexity of the minimum-link path problem in 3-space?" Our results imply that the problem is NP-hard even on terrains (and hence, due to discreteness of the answer, there is no FPTAS unless P=NP), but admits a PTAS.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases minimum-link path, diffuse reflection, terrain, bit complexity, NP-hardness

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.49

1 Introduction

The minimum-link path problem is fundamental in computational geometry [32, 15, 21, 23, 25, 26, 5, 19]. It concerns the following question: given a polyhedral domain D and two

* A preliminary version of this paper was presented at the 23rd Spanish Meeting on Computational Geometry in July 2015.
† I. K. is supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.023.208.
‡ M. L. is supported by the Netherlands Organisation for Scientific Research (NWO) under grant 639.021.123.
§ V. P. is supported by grant 2014-03476 from the Sweden's innovation agency VINNOVA.
¶ F. S. is supported by the Danish National Research Foundation under grant nr. DNRF84.

© Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals;
licensed under Creative Commons License CC-BY

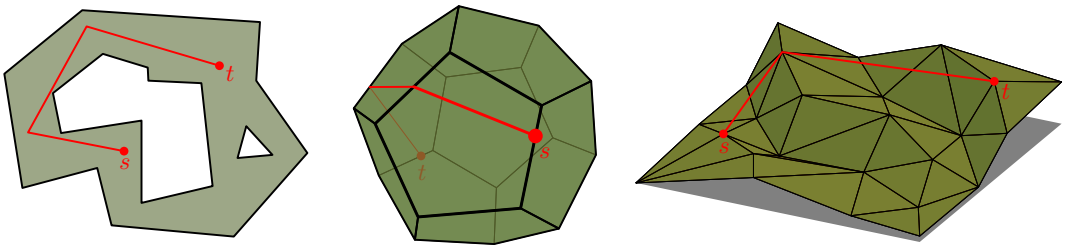


Figure 1 Left: $\mathsf{MinLinkPath}_{2,2}$ in a polygon with holes. Middle: $\mathsf{MinLinkPath}_{1,2}$ on a polyhedron. Right: $\mathsf{MinLinkPath}_{0,3}$ on a polyhedral terrain.

points s and t in D, find a polygonal path connecting s to t that lies in D and has as few links as possible.

In this paper, we revisit the problem in a general setting which encompasses several specific variants that have been considered in the literature. First, we nuance and tighten results on the bit complexity involved in optimal min-link paths. Second, we present and apply a novel generic NP-hardness construction. Third, we extend a simple polynomial-time approximation scheme.

Concretely, our results resolve several open problems. We prove that computing a min-link *diffuse reflection path* in a polygon with holes [16] is NP-hard, and show that the min-link path problem in 3-space [17] (Chapter 27.5, Open problem 3) is NP-hard as well (even for terrains). In both cases, there is no FPTAS unless P=NP, but there is a PTAS.

We use terms *links* and *bends* for edges and vertices of the path, saving the terms *edges* and *vertices* for those of the domain.

1.1 Problem Statement, Domains and Constraints

Due to their diverse applications, many different variants of min-link paths have been considered in the literature. These variants can be categorized by two aspects. Firstly, the *domain* can take very different forms. We consider several common domains, ranging from a simple polygon in 2D to complex scenes in full 3D or even in higher dimensions. Secondly, the links of the paths can be *constrained* to lie on the boundary of the domain, or bends may be restricted to vertices, edges, or higher-dimensional subcomplexes of the domain.

Problem Statement. Let D be a closed connected d-dimensional polyhedral domain. For $0 \leq a \leq d$ we denote by $D|^a$ the a-skeleton of D, i.e., its a-dimensional subcomplex. For instance, $D|^{d-1}$ is the boundary of D, and $D|^0$ is the set of vertices of D. Note that $D|^a$ is not necessarily connected.

▶ **Definition 1.** We define $\mathsf{MinLinkPath}_{a,b}(D, s, t)$, for $0 \leq a \leq b \leq d$ and $1 \leq b$, to be the problem of finding a min-link polygonal path in D between two given points s and t, where the bends of the solution (and s and t) are restricted to lie in $D|^a$ and the links of the solution are restricted to lie in $D|^b$. Fig. 1 illustrates several instances of the problem.

Domains. We recap the various settings that have been singled out for studies in computational geometry. We remark that we will not survey the rich field of path planning in rectilinear, or more generally, C-oriented worlds [1]; all our paths will be assumed to be unrestricted in terms of orientations of their links.

One classical distinction between working setups in 2D is *simple polygons* vs. *polygonal domains*. The former are a special case of the latter: simple polygons are domains without holes. Many problems admit more efficient solutions in simple polygons—loosely speaking, the golden standard is running time of $O(n)$ for simple polygons and of $O(n \log n)$ for polygonal domains of complexity n. This is the case, e.g., for the shortest path problem [18, 20]. For min-link paths, $O(n)$-time algorithms are known for simple polygons [32, 15, 21], but for polygonal domains with holes the fastest known algorithm runs in nearly quadratic time [25], which may be close to optimal due to 3SUM-hardness of the problem [26].

In 3D, a *terrain* is a polyhedral surface (often restricted to a bounded region in the xy-projection) that is intersected only once by any vertical line. Terrains are traditionally studied in GIS applications and are ubiquitous in computational geometry. Min-link paths are closely related to visibility problems, which have been studied extensively on terrains [13, 31, 22]. One step up from terrains, we may consider *simple* polyhedra (surfaces of genus 0), or *full 3D* scenes. Visibility has been studied in full 3D as well [27, 11, 33]. To our knowledge, min-link paths in higher dimensions have not been studied before (with the exception of [7] that considered rectilinear paths).

Constraints. In path planning on polyhedral surfaces or terrains, it is standard to restrict paths to the terrain.

Min-link paths, on the other hand, have various geographic applications, ranging from feature simplification [19] to visibility in terrains [13]. In some of these applications, paths are allowed to live in free space, while bends are still restricted to the terrain. In the GIS literature, out of simplicity and efficiency concerns, it is common to constrain bends even further to vertices of the domain (or, even more severely, the vertices of the terrain itself may be restricted to a grid, as in the *digital elevation map* (DEM) model).

In a vanilla min-link path problem the location of vertices (bends) of the path are unconstrained, i.e., they can occur anywhere in the free space. In the *diffuse reflection* model [16, 29, 3, 5] the bends are restricted to occur on the boundary of the domain. Studying this kind of paths is motivated by ray tracing in realistic rendering of 3D scenes in graphics, as light sources that can reach a pixel with fewer reflections make higher contributions to intensity of the pixel [14, 8]. Despite the 3D graphics motivation, all work on diffuse reflection has been confined to 2D polygonal domains, where the path bends are restricted to edges of the domain (and even in 2D, the complexity of the problem was open before).

1.2 Representation and Computation

In computational geometry, the standard model of computation is the *real RAM*, which represents the memory as an infinite sequence of storage cells, each of which can store any real number or integer. The real RAM is prefered for its elegance, but may not always be the best representation of physical computers. In contrast, the *word RAM* stores a sequence of w-bit words, where $w \geq \log n$ (and n is the problem size). The word RAM is much closer to reality, but complicates the analysis of geometric problems.

This difference is often insignificant, as the real numbers involved in solving many geometric problems are in fact algebraic numbers of low degree in a bounded domain, which can be described exactly with constantly many w-bit words. Path planning is notoriously different in this respect. Indeed, in the real RAM both the Euclidean shortest paths and the min-link paths in 2D can be found in optimal times. On the contrary, much less is known about the complexity of the problems in other models. For L_2-shortest paths the issue is that their length is represented by the sum of square roots and it is not known whether comparing the sum to a number can be done efficiently (if yes, one may hope that the difference between

Table 1 Computational complexity of $\mathsf{MinLinkPath}_{a,b}$ for $a \leq b \leq 3$. Results with citations are known, results marked with $\star$ are from this paper. Results without marks are trivial.

$\mathsf{MinLinkPath}_{a,b}$	$b=1$	$b=2$	$b=3$
$a=0$	$O(n)$	$O(n^2)$	$O(n^2)$
$a=1$	$O(n)$	Simple Polygon: $O(n^9)$[5] Full 2D: NP-hard$\star$ PTAS$\star$	NP-hard$\star$ (even in terrains) PTAS$\star$
$a=2$	N/A	Simple Polygon: $O(n)$[32] Full 2D: $O(n^2\alpha(n)\log^2 n)$[25] PTAS$\star$	NP-hard$\star$ (even in terrains) PTAS$\star$
$a=3$	N/A	N/A	Terrains: $O(1)$ Full 3D: NP-hard$\star$ PTAS$\star$

the models vanishes). Slightly more is known about min-link paths, for which the models are *provably* different: Snoeyink and Kahan [23] observed that the region of points reachable by k-link paths may have vertices needing $\Omega(k \log n)$ bits to be represented. One of the results we present in this paper is the matching upper bound on the bit complexity of min-link paths in 2D.

Relatedly, when studying the computational complexity of geometric problems, it is often not trivial to show a problem is in NP. Even if a potential solution can be verified in polynomial time, if such a solution requires real numbers that cannot be described succinctly, the set of solutions to try may be too large. Recently, there has been some interest in computational geometry in showing problems are in NP [12] (see also [30]).

A common practical approach to avoiding bit complexity issues is to approximate the problem by restricting solutions to use only vertices of the input. In min-link paths, this corresponds to $\mathsf{MinLinkPath}_{0,b}$. Although such paths can be computed efficiently, it can be shown with a simple example (refer to the full version [24]) that the number of links in such a setting may be a linear factor higher than when considering geometric versions.

1.3 Results

We give hardness results and approximation algorithms for various versions of the min-link path problem (see also Table 1). Specifically,

- In Section 2 we show a general lower bound on the bit complexity of min-link paths of $\Omega(n \log n)$ bits. (This was previously claimed, but not proven, by Snoeyink and Kahan [23].) We show that the bound is tight in 2D and we argue that this implies that $\mathsf{MinLinkPath}_{a,2}$ is in NP. In Section 5, we argue that in 3D the boundary of the k-link reachable region can consist of $2k$-th order algebraic curves, potentially leading to exponential bit complexity.
- In Section 3.1 we present a blueprint for showing NP-hardness of min-link path problems. We apply it to prove NP-hardness of the diffuse reflection path problem ($\mathsf{MinLinkPath}_{1,2}$) in 2D polygonal domains with holes in Section 3.2. In Section 6, we use the same blueprint to prove that all non-trivial versions, defined above, of min-link path problems in 3D are weakly NP-hard. We also note that the min-link path problems have no FPTAS and no additive approximation (unless P=NP).

- In Section 4, we extend the 2-approximation algorithm from [17, Ch. 27.5], based on computing weak visibility between sets of potential locations of the path's bends, to provide a simple PTAS for $\mathsf{MinLinkPath}_{2,2}$, which we also adapt to $\mathsf{MinLinkPath}_{1,2}$. In Section 7, we give simple constant-factor approximation algorithms for higher-dimensional versions of min-link path problems, and use them to show that all versions admit PTASes.
- In Section 7.3, we focus on diffuse reflection ($\mathsf{MinLinkPath}_{2,3}$) in 3D on terrains—the version that is most important in practice. We give a 2-approximation algorithm that runs faster than the generic algorithm from [17, Ch. 27.5]. We also present an $O(n^4)$-size data structure encoding visibility between points on a terrain and argue that the size of the structure is asymptotically optimal.

Omitted proofs can be found in the full version of the paper [24].

2 Algebraic Complexity in $\mathbb{R}^2$

2.1 Lower bound on the Bit complexity

Snoeyink and Kahan [23] claim to "give a simple instance in which representing path vertices with rational coordinates requires $\Theta(n^2 \log n)$ bits". In fact, they show that, there exists a simple polygon (whose vertices have integer coordinates encoded with $O(\log n)$ bits), such that the region reachable from one of its vertices s with k links has vertices whose coordinates have bit complexity $k \log n$. Note however, that this does not directly imply that a min-link path from s to another point t with low-complexity (integer) coordinates must necessarily have such high-complexity bends (i.e., if t itself is not a high-complexity vertex of a k-reachable region, one potentially could hope to avoid placing the internal vertices of an s-t min-link path on such high-complexity points). Below we present a construction where the intermediate vertices actually require $\Omega(k \log n)$ bits to be described, even if s and t can be represented using only $\log n$ bits each. We first prove this for the $\mathsf{MinLinkPath}_{1,2}$ variant of the problem, and then extend our results to paths that may bend anywhere within the polygon, i.e., $\mathsf{MinLinkPath}_{2,2}$.

▶ **Lemma 2.** *There exists a simple polygon P, and points s and t in P such that:* (i) *all the coordinates of the vertices of P and of s and t can be represented using $O(\log n)$ bits, and* (ii) *any s-t min-link path that bends only on the edges of P has vertices whose coordinates require $\Omega(k \log n)$ bits, where k is the length of a min-link path between s and t.*

Proof. We will refer to numbers with $O(\log n)$ bits as *low-complexity*.

The general idea in our construction is as follows. We start with a low-complexity point $s' = b_0$ on an edge e_0 of the polygon. We then consider the furthest point b_{i+1} on the boundary of P that is reachable from b_i. More specifically, we require that any point on the boundary of P between s' and b_i is reachable by a path of at most i links. We will obtain b_{i+1} by projecting b_i through a vertex c_i. Each such step will increase the required number of bits for b_{i+1} by $\Theta(\log n)$. Eventually, this yields a point b_k on edge e_k. Let t' be the k-reachable point on e_k closest to b_k that has low complexity. Since all points along the boundary from s' to b_k are reachable, and the vertices of P have low complexity, such a point is guaranteed to exist. We set $a_k = t'$ and project a_i through c_{i-1} to a_{i-1} to give us the furthest point (from t') reachable by $k - i$ links. See Fig. 2 for an illustration.

The points in the interval $I_i = [a_i, b_i]$, with $1 \leq i < k$, are reachable from s' by exactly i links, and reachable from t' by exactly $k - i$ links. So, to get from s' to t' with k links, we need to choose the i^{th} bend of the path to be within the interval $[a_i, b_i]$. By construction,

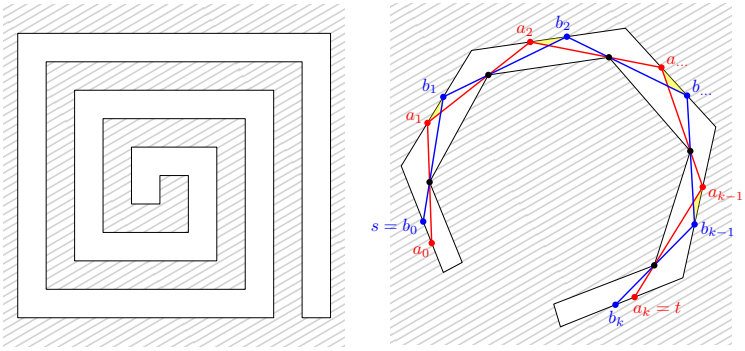


Figure 2 (a) A spiral, as used in the construction by Kahan and Snoeyink. It uses integer coordinates with $O(\log n)$ bits. (b) The general idea.

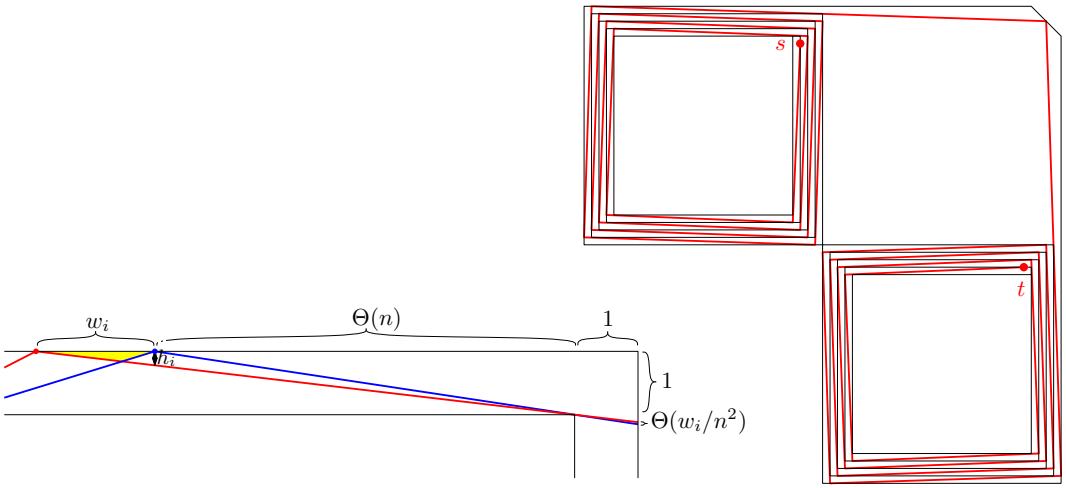


Figure 3 Left: The interval I_i of length w_i produces an interval I_{i+1} of length at most $w_{i+1} = h_i/\Theta(n) = \Theta(w_i/n^2)$, where $h_i = w_i/(w_i + \Theta(n))$. When the i^{th} link can be anywhere in region R_i (shown in yellow), it follows that R_i has height at most h_i, and width at most w_i. Right: An overview of our polygon P and the minimum-link path that has high-complexity coordinates.

the intervals for i close to one or close to k must contain low-complexity points. We now argue that we can build the construction such that $I_{k/2}$ contains no low-complexity points.

Observe that, if an interval contains no points that can be described with fewer than m bits, its length can be at most 2^{-m}. So, we have to show that $I_{k/2}$ has length at most $2^{-k \log n}$. By construction, the interval I_k has length at most one. Similarly, the length of I_0 can be chosen to be at most one (if it is larger, we can adjust $s' = b_0$ to be the closest integer point to a_0). Now observe that in every step, we can reduce the length w_i of the interval I_i by a factor $\Theta(n^2)$, using a construction like in Fig. 3 (left). Our overall construction is then shown in Fig. 3 (right).

It follows that $I_{k/2}$ cannot contain two low-complexity points that are close to each other. Note however, that it may still contain one such point. It is easy to see that there is a sub-interval $J_{k/2} = [\ell_{k/2}, r_{k/2}] \subseteq I_{k/2}$ of length $w_{k/2}/2$ that contains no points with fewer than $k \log n$ bits. By choosing $J_{k/2}$ we have restricted the interval that must contain the $(k/2)^{\text{th}}$ bend. This also restricts the possible positions for the i^{th} bend to an interval $J_i \subseteq I_i$. We find these intervals by projecting $\ell_{k/2}$ and $r_{k/2}$ through the vertices of P. Note that s'

and t' may not be contained in J_0 and J_k, respectively, so we pick a new start point $s \in J_0$ and end point $t \in J_k$ as follows. Let $m_{k/2}$ be the mid point of $J_{k/2}$ and project m_i through the vertices of P. Now choose s to be a low-complexity point in the interval $[m_0, r_0]$, and t to be a low-complexity point in the interval $[\ell_k, m_k]$. Observe that $[m_0, r_0]$ and $[\ell_k, m_k]$ have length $\Theta(1)$—as $[\ell_{k/2}, m_{k/2}]$ and $[m_{,k/2}, r_{k/2}]$ have length $w_{k/2}/4$—and thus contain low complexity points. Furthermore, observe that t is indeed reachable from s by a path with $k-1$ bends (and thus k links), all of which much lie in the intervals J_i, $1 \leq i < k$. For example using the path that uses all points m_i. Thus, we have that t is reachable from s by a min-link path of k links, and we need $\Omega(k \log n)$ bits to describe the coordinates of some vertices in such a path. ◀

Next, we extend the construction from Lemma 2 to the case in which the bends may also lie in the interior of P.

▶ **Lemma 3.** *There exists a simple polygon P, and points s and t in P such that:* (i) *all the coordinates of the vertices of P and of s and t can be represented using $O(\log n)$ bits, and* (ii) *any s-t min-link path has vertices whose coordinates require $\Omega(k \log n)$ bits, where k is the length of a min-link path between s and t.*

2.2 Upper bound on the Bit complexity

In this section we show that the bound of Snoeyink and Kahan [23] on the complexity of k-link reachable regions is tight. Consider a polygon P and a point s in it. Let $\mathcal{R} = \{R_1, R_2, R_3, \dots\}$, where R_1 is the set of all points in P that see s, and R_i is the set of points in P that see some point in R_{i-1} for $i \geq 2$, i.e., region R_i consists of all the points in P that are illuminated by region R_{i-1}. Representing $\mathcal{R}$ as polygons with rational coordinates requires $O(n^2 \log n)$ for any polygon P, assuming that representation of the coordinates of any vertex of P requires at most $c_0 \log n$ bits for some constant c_0. Thus, we have matching lower and upper bounds on the bit complexity of a min-link path in $\mathbb{R}^2$.

▶ **Theorem 4.** *Representing a vertex of region R_i requires $O(i \log n)$ bits. Representing the regions in $\mathcal{R}$ as polygons with rational coordinates requires $O(n^2 \log n)$ bits each.*

▶ **Corollary 5.** *If there exists a solution with k links, there also exists one in which the coordinates of the bends use at most $O(k \log n)$ bits.*

▶ **Theorem 6.** *MinLinkPath$_{a,2}$ is in NP.*

Proof. We need to show that a candidate solution can be verified in polynomial time. A potential solution needs at most n links. By Corollary 5, we only need to verify candidate solutions that consist of bends with $O(n \log n)$-bit coordinates. Given such a candidate, we need to verify pairwise visibility between at most n pairs of points with $O(n \log n)$-bit coordinates, which can be done in polynomial time. ◀

3 Computational Complexity in $\mathbb{R}^2$

In this section we show that MinLinkPath$_{1,2}$ is NP-hard. To this end, we first provide a blueprint for our reduction in Section 3.1. In Section 3.2 we then show how to "instantiate" this blueprint for MinLinkPath$_{1,2}$ in a polygon with holes.

3.1 A Blueprint for Hardness Reductions

We reduce from the 2-Partition problem: Given a set of integers $A = \{a_1, \ldots, a_m\}$, find a subset $S \subseteq A$ whose sum is equal to half the sum of all the numbers. The main idea behind all the hardness reductions is as follows. Consider a 2D construction in Fig. 4 (left). Let point s have coordinates $(0, 0)$, and t (not in the figure) have coordinates $(\sum a_i/2, 4m - 2)$. For now, in this construction, we will consider paths from s to t that are only allowed to bend on horizontal lines with even y-coordinates. Moreover, we will count an intersection with each such horizontal line as a bend. We will place fences along the lines with odd y-coordinates in such a way that an s-t path with $2m - 1$ links exists (that bends only on horizontal lines with even y-coordinates) if and only if there is a solution to the 2-Partition instance.

Call the set of horizontal lines $\ell_0 : y = 0$, $\ell_i : y = 4i - 2$ for $1 \leq i \leq m$ *important* (dashed lines in Fig. 4), and the set of horizontal lines $\ell'_i : y = 4i - 4$ for $2 \leq i \leq m$ *intermediate* (dash-dotted lines in Fig. 4). Each important line ℓ_i will "encode" the running sums of all subsets of the first i integers $A_i = \{a_1, \ldots, a_i\}$. That is, the set of points on ℓ_i that are reachable from s with $2i - 1$ links will have coordinates $(\sum_{a_j \in S_i} a_j, 4i - 2)$ for all possible subsets $S_i \subseteq A_i$.

Call the set of horizontal lines $f_1 : y = 1$, $f_i : y = 4i - 5$ for $2 \leq i \leq m$ *multiplying*, and the set of horizontal lines $f'_i : y = 4i - 3$ for $2 \leq i \leq m$ *reversing*. Each multiplying line f_i contains a fence with two zero-width slits that we call 0-slit and a_i-slit. The 0-slit with x-coordinate 0 corresponds to not including integer a_i into subset S_i, and the a_i-slit with x-coordinate $\sum_1^i a_j - a_i/2$ corresponds to including a_i into S_i. Each reversing line f'_i contains a fence with two zero-width slits (reversing 0-slit and reversing a_i-slit) with x-coordinates 0 and $\sum_1^i a_j$ that "put in place" the next bends of potential min-link paths, i.e., into points on ℓ_i with x-coordinates equal to the running sums of S_i. We add a vertical fence of length 1 between lines ℓ'_i and f'_i at x-coordinate $\sum_1^i a_j/2$ to prevent the min-link paths that went through the multiplying 0-slit from going through the reversing a_i-slit, and vice versa.

As an example, consider (important) line ℓ_2 in Fig. 4. The four points on line ℓ_2 that are reachable from s with 3 links have x-coordinates $\{0, a_1, a_2, a_1 + a_2\}$. The points on line ℓ'_3 that are reachable from s with a path (with 4 links) that goes through the 0-slit of line f_3 have x-coordinates $\{0, -a_1, -a_2, -(a_1 + a_2)\}$, and the points on ℓ'_3 that are reachable from s through the a_3-slit have x-coordinates $\{2a_1+2a_2+a_3, a_1+2a_2+a_3, 2a_1+a_2+a_3, a_1+a_2+a_3\}$. The reversing 0-slit of line f'_3 places the first four points into x-coordinates $\{0, a_1, a_2, a_1 + a_2\}$ on line ℓ_3, and the reversing a_3-slit of line f'_3 places the second four points into x-coordinates $\{a_3, a_1 + a_3, a_2 + a_3, a_1 + a_2 + a_3\}$ on ℓ_3.

In general, consider some point p on line ℓ_{i-1} that is reachable from s with $2i - 3$ links. The two points on ℓ'_i that can be reached from p with one link have x-coordinates $-p_x$ and $2\sum_1^i a_j - a_i - p_x$, where p_x is the x-coordinate of p. Consequently, the two points on ℓ_i that can be reached from p with two links have x-coordinates p_x and $p_x + a_i$. Therefore, for every line ℓ_i, the set of points on it that are reachable from s with a min-link path have x-coordinates equal to $\sum_{a_j \in S_i} a_j$ for all possible subsets $S_i \subseteq A_i$. Consider line ℓ_m and the destination point t on it. There exists an s-t path with $2m - 1$ links if and only if the x-coordinate of t is equal to $\sum_{a_j \in S} a_j$ for some $S \subseteq A$. The complexity of the construction is polynomial in the size of the 2-Partition instance. Therefore, finding a min-link path from s to t in our 2D construction is NP-hard.

▶ Remark. Instead of 0-width slits, we could use slits of positive width $w = o(\frac{1}{4m})$; since the width of the light beam grows by $2w$ between two consecutive important lines, the maximum shift of the path on ℓ_m due to the positive width of slits will be at most $(2m - 1) \times 2w < 1$.

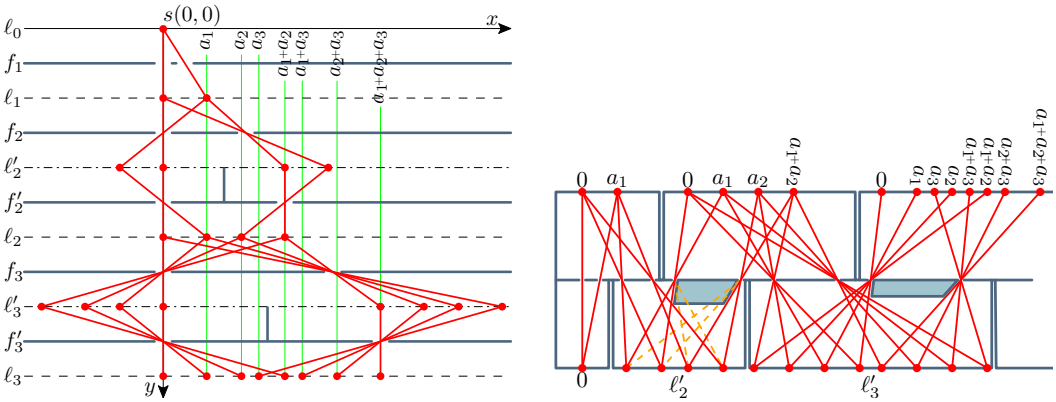


Figure 4 Left: The first few lines of a 2D construction depicting the general idea behind the hardness proofs: important lines ℓ_0–ℓ_3, intermediate lines ℓ'_1–ℓ'_3, multiplying lines f_1–f_3, and reversing lines f'_1–f'_3. The slits in the fences on multiplying and reversing lines are placed in such a way that the locations on ℓ_i that are reachable from s with $2i-1$ links correspond to sums formed by all possible subsets of $\{a_1, \ldots, a_i\}$. Right: There exists an s-t diffuse reflection path with $2m-1$ links iff 2-Partition instance is feasible.

3.2 Hardness of MinLinkPath$_{1,2}$

To show the hardness of the diffuse reflection problem in 2D, we turn our construction from Section 3.1 into a "zigzag" polygon (Fig. 4 (right)); the fences are turned into obstacles within the corresponding corridors, and slits remain slits—the only free space through which it is possible to go with one link between the polygon edges that correspond to consecutive lines ℓ'_i and ℓ_i (or ℓ_{i-1} and ℓ'_i). This retains the crucial property of 2D construction: locations reachable with fewest links on the edges of the polygon correspond to sums of numbers in the subsets of A. We conclude:

▶ **Theorem 7.** *MinLinkPath$_{1,2}$ in a 2D polygonal domain with holes is NP-hard.*

Overall our reduction bears resemblance to the classical *path encoding* scheme [6] used to prove hardness of 3D shortest path and other path planning problems, as we also repeatedly double the number of path homotopy types; however, since we reduce from 2-Partition (and not from 3SAT, as is common with path encoding), our proof(s) are much less involved than a typical path-encoding one.

No FPTAS & No additive approximation. Obviously, problems with a discrete answer (in which a second-best solution is separated by at least 1 from the optimum) have no FPTAS. We can slightly amplify the hardness results, showing that for any constant K it is not possible to find an additive-K approximation for our problems: Concatenate K instances of the construction from the hardness proof, aligning s in the instance $k+1$ with t from the instance k. Then there is a path with $K(2m-1)$ links through the combined instance if the 2-Partition is feasible; otherwise $K(2m-1)+K-1$ links are necessary. Thus an algorithm, able to differentiate between instances in which the solution has $K(2m-1)$ links and those with $K(2m-1)+K-1$ links in poly(mK) = poly(m) time, would also be able to solve 2-Partition in the same time.

4 Algorithmic Results in $\mathbb{R}^2$

4.1 Constant-factor Approximation

$\mathsf{MinLinkPath}_{2,2}$ in 2D can be solved exactly [25]. For $\mathsf{MinLinkPath}_{1,2}$, [16] gives a 3-approximation.

4.2 PTAS

We describe a $(1+\varepsilon)$-approximation scheme for $\mathsf{MinLinkPath}_{1,2}$, based on building a graph of edges of D that are *k-link weakly visible*.

Consider the set F of all edges of D (i.e., $\bigcup F = D|^1$). To avoid confusion between edges of D and edges of the graph we will build, we will call elements of F *features* (this will also allow us to extend the ideas to higher dimensions later). Two features $f, f' \in F$ are *weakly visible* if there exist mutually visible points $p \in f$ and $p' \in f'$; more generally, we say f and f' are *k-link weakly visible* if there is a k-link path from p to p' (with links restricted to $D|^1$).

For any constant $k \geq 1$, we construct a graph $G^k = (F, E_k)$, where E_k is the set of pairs of k-link weakly visible features. Let $\pi^k = \{f_0, f_1, \ldots, f_\ell\}$, with $f_0 \ni s$ and $f_\ell \ni t$ be a shortest path in G from the feature containing s to the feature containing t. We describe how to transform π^k into a solution π^k_* for the $\mathsf{MinLinkPath}_{1,2}$ problem. Embed edges of π^k into D as k-link paths. This does not yet connect s to t since the endpoint of edge $f_{i-1}f_i$ inside feature f_i does not necessarily coincide with the endpoint of edge $f_i f_{i+1}$; to create a connected path, we observe that the two endpoints can always be connected by two extra links via some feature that is mutually visible from both points (or a single extra link within f_i if we allow links to coincide within the boundary of D).

▶ **Lemma 8.** *The number of links in π^k_* is at most $(1+1/k)\mathsf{opt}$.*

We now argue that the weak k-link visibility between features can be determined in polynomial time using "staged illumination"—the predominant technique for finding min-link paths (see Chapters 26.4 and 27.3 in the handbook [17]): starting from each feature f, find the set $W(f)$ of points on other features weakly visible from f, then find the set weakly visible from $W^2(f) = W(W(f))$, repeat k times to obtain the set $W^k(f)$ reachable from f with k links; feature f' can be reached from f with k links iff $W^k(f) \cap f' \neq \varnothing$. For constant k, building $W^k(f)$ takes time polynomial in n, although possibly exponential in k (in fact, explicit bounds on the complexity of $W^k(f)$ for diffuse reflection problem were obtained in [4, 3, 5]). This can be seen by induction: Partition the set $W^{i-1}(f)$ into the polynomial number of constant-complexity pieces. For each piece p, each element e of the boundary of the domain, and each feature f', compute the part of f' shadowed by e from the light sources on p—this can be done in constant time analogously to determining weak visibility between two features above (by considering the part of $p \times f'$ carved out by the occluder e). The part of f' weakly seen from $W^{i-1}(f)$ is the union, over all parts p, of the complements of the sets occluded by all elements e; since there is a polynomial number of parts, elements and features, it follows that $W^i(f)$ can be constructed in polynomial time.

▶ **Theorem 9.** *For a constant k the path π^k_*, having at most $(1+1/k)\mathsf{opt}$ links, can be constructed in polynomial time.*

5 Algebraic Complexity in $\mathbb{R}^3$

Consider the bit complexity of $\mathsf{MinLinkPath}_{a,b}$ for a polyhedral domain D in $\mathbb{R}^3$. The lower bounds on the bit complexity for $a \leq b \leq 2$ in $\mathbb{R}^2$ obviously extend to $\mathbb{R}^3$. In this section we focus on the upper bound; we characterize region R_i reachable from a given point s with i links and discuss the upper bound on its bit complexity.

Order of the boundary curves. Assume that representations of the coordinates of any vertex of D and s require at most $c_0 \log n$ bits for some constant c_0. Analogous to Section 2, we define a sequence of regions $\mathcal{R} = \{R_1, R_2, R_3, \dots\}$, where R_1 is the set of all points in D that see s, and R_i is the region of points in D that see some point in R_{i-1} for $i \geq 2$, i.e., region R_i consists of all the points of D that are illuminated by region R_{i-1}. Note, that R_i is a union of subsets of faces of D. Therefore, when we will speak of the boundaries (in the plural form of the word) of R_i, that we will denote as ∂R_i, we will mean the illuminated sub-intervals of edges of D as well as the frontier curves interior to the faces of D.

Unlike in 2D, the boundaries of R_i interior to the faces of D do not necessarily consist of straight-line segments. Observe, that a union of all lines intersecting three given lines in 3D forms a hyperboloid, and therefore, a straight-line segment on the boundaries of R_{i-1} forces the corresponding part of ∂R_i to be an intersection of a hyperboloid and a plane, i.e., a hyperbola. Moreover, the order of the curves of ∂R_i will grow with i, but at most linearly.

▶ **Theorem 10.** *The boundaries of region R_i are curves of order at most $2i + 1$ for $i \geq 2$, and at most 2 for $i = 1$.*

Bit complexity. The fact that the order of the curves on the boundaries of R_i grows linearly may give hope that the bit complexity of representation of R_i can be bounded from above similarly to Section 2.2. However, following similar calculations allowed us to obtain only an exponential upper bound for the space required to store the coordinates of vertices of R_i.

▶ **Lemma 11.** *The coordinates of a vertex of R_i can be stored in $O(9^i)$ space.*

6 Computational Complexity in $\mathbb{R}^3$

We will show now how to use our blueprint from Section 3.1 to build a terrain for the $\mathsf{MinLinkPath}_{1,2}$ problem such that a path from s to t with $2n - 1$ links will exist if and only if there exists a subset $S \subseteq A$ whose sum is equal to half the sum of all integers $A = \{a_1, \dots, a_m\}$. Take the 2D construction and bend it along all the lines ℓ_i and ℓ'_i, except ℓ_0 and ℓ_m (refer to Fig. 5). Let the angles between consecutive faces be $\pi - \delta$ for some small angle $\delta < \pi/4m$ (so that the sum of the bends between the first face (between ℓ_0 and ℓ_1) and the last face (between ℓ'_m and ℓ_m) is less than π). Build a fence of height $\tan(\delta/4)$ on each face according to the 2D construction. The height of the fences is small enough so that no two points on consecutive fences see each other. Therefore, for two points s and t placed on ℓ_0 and ℓ_m as described above, an s-t path with $2m - 1$ links must bend only on ℓ_i and ℓ'_i and pass through the slits in the fences. Finding a min-link path on such a terrain is equivalent to finding a min-link path (with bends restricted to ℓ_i and ℓ'_i) in the 2D construction. Therefore,

▶ **Theorem 12.** *$\mathsf{MinLinkPath}_{1,2}$ on a terrain is NP-hard.*

Observe that bending in the interior of a face cannot reduce the link distance between s and t. Hence, our reduction also shows that $\mathsf{MinLinkPath}_{2,2}$ is NP-hard. Furthermore,

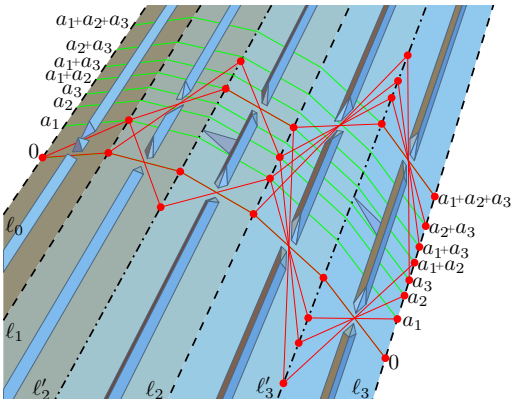


Figure 5 The terrain obtained by bending the 2D construction along the important and intermediate lines. The height of the fences is low enough that no two points on consecutive fences can see each other.

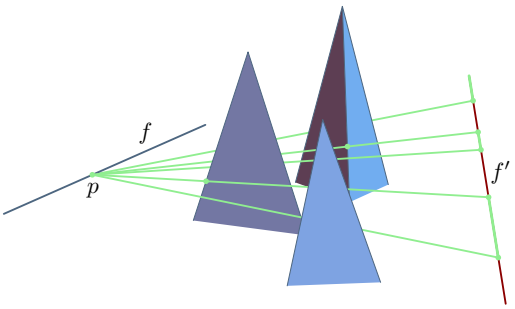


Figure 6 The weak visibility $W(f)$ restricted to edge f' is the union of all visible intervals (green) over all points $p \in f$. If this region is non-empty, f and f' are weakly visible.

lifting the links from the terrain surface into $\mathbb{R}^3$ also does not reduce link distance; we can make sure that the fences are low in height, so that fences situated on different faces do not see each other. Therefore, jumping onto the fences is useless. Hence, $\mathsf{MinLinkPath}_{1,3}$ and $\mathsf{MinLinkPath}_{2,3}$ are also NP-hard.

$\mathsf{MinLinkPath}_{a,b}$ in general polyhedra. Since a terrain is a special case of a 3D polyhedra, it follows that $\mathsf{MinLinkPath}_{1,2}$, $\mathsf{MinLinkPath}_{2,2}$, $\mathsf{MinLinkPath}_{1,3}$, and $\mathsf{MinLinkPath}_{2,3}$ are also NP-hard for an arbitrary polyhedral domain in $\mathbb{R}^3$. Our construction does not immediately imply that $\mathsf{MinLinkPath}_{3,3}$ is NP-hard. However, we can put a copy of the terrain slightly above the original terrain (so that the only free space is the thin layer between the terrains). When this layer is thin enough, the ability to take off from the terrain, and bend in the free space, does not help in decreasing the link distance from s to t. Thus, $\mathsf{MinLinkPath}_{3,3}$ is also NP-hard.

▶ **Corollary 13.** *$\mathsf{MinLinkPath}_{a,b}$ with $a \geq 1$ and $b \geq 2$ in a 3D domain D is NP-hard. This holds even if D is just a terrain.*

7 Algorithmic Results in $\mathbb{R}^3$

7.1 Constant-factor Approximation

Our approximations refine and extend the 2-approximation for min-link paths in higher dimensions suggested in Chapter 26.5 (section Other Metrics) of the handbook [17] (see also Ch. 6 in [28]); since the suggestion is only one sentence long, we fully quote it here:

> Link distance in a polyhedral domain in $\mathbb{R}^d$ can be approximated (within factor 2) in polynomial time by searching a weak visibility graph whose nodes correspond to simplices in a simplicial decomposition of the domain.

Indeed, consider $D|^a$, the set of all points where the path is allowed to bend, decompose $D|^a$ into a set F of small-complexity convex pieces, and call each piece a *feature*. Similar to Section 4.2, we say that two features f and f' are weakly visible if there exist mutually

visible points $p \in f$ and $p' \in f'$; more generally, the weak visibility region $W(f)$ is the set of points that see at least one point of f, so f' is weakly visible from f iff $f' \cap W(f) \neq \varnothing$ (in terms of illumination, $W(f)$ is the set of points that get illuminated when light sources are placed at every point of f). See Fig. 6 for an illustration.

Weak visibility between two features f and f' can be determined straightforwardly by building the set of pairs of points (p, p') in the parameter space $f \times f'$ occluded by (each element of) the obstacles. To be precise, $f \times f'$ is a subset of $\mathbb{R}^{2a}$. Now, consider $D|^{d-1}$, which we also decompose into a set of constant-complexity *elements*. Each element e defines a set $B(e) = \{(p, p') \in f \times f' : pp' \cap e \neq \varnothing\}$ of pairs of points that it blocks; since e has constant complexity, the boundary of $B(e)$ consists of a constant number of curved surfaces, each described by a low degree polynomial. Since there are $O(n)$ elements, the union (and, in fact, the full arrangement) of the sets $B(e)$ for all e can be built in $O(n^{4a-3+\varepsilon})$ time, for an arbitrarily small $\varepsilon > 0$, or $O(n^2)$ time in case $a = 1$ [2]. We define the *visibility map* $M(f, f') \subseteq f \times f'$ to be the complement of the union of the blocking sets, i.e., the map is the set of mutually visible pairs of points from $f \times f'$. We have:

▶ **Lemma 14.** *$M(f, f')$ can be built in $O(n^{\max(2,4a-3+\varepsilon)})$ time, for an arbitrary $\varepsilon > 0$.*

The features f and f' weakly see each other iff $M(f, f')$ is not empty. Let G be the graph on features whose edges connect weakly visible features; s and t are added as vertices of G, connected to features (weakly) seen from them. Let $\pi = \{f_0, f_1, \ldots, f_\ell\}$, with $f_0 = s$ and $f_\ell = t$ be a shortest s-t path in G; ℓ is the length of π. Embed the edges of π into the geometric domain, putting endpoints of the edges arbitrarily into the corresponding features. This does not yet connect s to t since the endpoint of edge $f_{i-1}f_i$ inside feature f_i does not necessarily coincide with the endpoint of edge f_if_{i+1}; to create a connected path, connect the two endpoints by an extra link within f_i (this is possible since the features are convex).

Bounding the approximation ratio of the above algorithm is straightforward: Let opt denote a min-link s-t path and, abusing notation, also the number of links in it. Consider the features to which consecutive bends of opt belong; the features are weakly visible and hence are adjacent in G. Thus $\ell \leq \mathsf{opt}$. Adding the extra links inside the features adds at most $\ell - 1$ links. Hence the total number of links in the produced path is at most $2\ell - 1 < 2\mathsf{opt}$.

Since G has $O(n)$ nodes and $O(n^2)$ edges, Dijkstra's algorithm will find the shortest path in it in $O(n^2)$ time.

▶ **Theorem 15** (cf. [17, Ch. 27.5])**.** *A 2-approximation to $\mathsf{MinLinkPath}_{a,b}$ can be found in $O(n^{2+\max(2,4a-3+\varepsilon)})$ time, where $\varepsilon > 0$ is an arbitrarily small constant.*

Interestingly, the running time in Theorem 15 depends only on a, and not on b or d, the dimension of D (of course, $a \leq d$, so the runtime is bounded by $O(n^{2+\max(2,4d-3+\varepsilon)})$ as well).

7.2 PTAS

To get a $(1 + 1/k)$-approximation algorithm for any constant $k \geq 1$, we expand the above handbook idea by searching for a shortest s-t path π^k in the graph G^k whose edges connect features that are k-link weakly visible. Similarly to Section 4.2, we obtain the following.

▶ **Theorem 16.** *For a constant k the path π^k_*, having at most $(1 + 1/k)\mathsf{opt}$ links, can be constructed in polynomial time.*

7.3 The global visibility map of a terrain

Using the result from Theorem 15 for $\mathsf{MinLinkPath}_{2,3}$ on terrains, we get a 2-approximate min-link path in $O(n^{7+\varepsilon})$ time (since the path can bend anywhere on a triangle of the

terrain, the features are the triangles and intrinsic dimension $d = 2$). In this section we show that a faster, $O(n^4)$-time 2-approximation algorithm is possible. We also consider encoding visibility between all points on a terrain (not just between features, as the visibility map from Section 7 does): we give an $O(n^4)$-size data structure for that, which we call the terrain's *global visibility map*, and provide an example showing that the size of the structure is worst-case optimal.

Specifically, the following results are proved in the full version of the paper [24]:

▶ **Theorem 17.** *A 2-approximation for* $\mathsf{MinLinkPath}_{2,3}$ *in a terrain can be found in* $O(n^4)$ *time.*

▶ **Theorem 18.** *Determining weak visibility between a pair of edges in a polygonal domain with holes is 3SUM-hard.*

▶ **Theorem 19.** *Determining weak visibility between a pair of edges in a terrain is 3SUM-hard.*

▶ **Theorem 20.** *The complexity of the global visibility map, encoding all pairs of mutually visible points on a terrain (or on a set of obstacles in 3D) of complexity* n*, is* $\Theta(n^4)$*.*

8 Conclusion

We considered minimum-link path problems in 3D, showing that most versions are NP-hard but admit PTASes; we also obtained similar results for the diffuse reflection problem in 2D polygonal domains with holes. The biggest remaining open problem is whether pseudopolynomial-time algorithms are possible for the problems: our reductions are from 2-Partition, and hence do not show strong hardness. A related question is exploring bit complexity of min-link paths in 3D (note that finding a min-link path with integer vertices is already weakly NP-hard for the case of simple polygons in 2D [10]).

Acknowledgments. We thank Joe Mitchell and Jean Cardinal for fruitful discussions on this work and the anonymous reviewers for their helpful comments.

References

1 John Adegeest, Mark H. Overmars, and Jack Snoeyink. Minimum-link c-oriented paths: Single-source queries. *Int. J. of Computational Geometry and Applications*, 4(1):39–51, 1994.

2 Pankaj K Agarwal and Micha Sharir. Arrangements and their applications. *Handbook of computational geometry*, pages 49–119, 2000.

3 Boris Aronov, Alan R. Davis, Tamal K. Dey, Sudebkumar Prasant Pal, and D. Chithra Prasad. Visibility with multiple reflections. *Discrete & Computational Geometry*, 20(1):61–78, 1998. doi:10.1007/PL00009378.

4 Boris Aronov, Alan R. Davis, Tamal K. Dey, Sudebkumar Prasant Pal, and D. Chithra Prasad. Visibility with one reflection. *Discrete & Computational Geometry*, 19(4):553–574, 1998. doi:10.1007/PL00009368.

5 Boris Aronov, Alan R. Davis, John Iacono, and Albert Siu Cheong Yu. The complexity of diffuse reflections in a simple polygon. In *Proc. 7th Latin American Symposium on Theoretical Informatics*, pages 93–104, 2006.

6 J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th Annual Symposium on Foundations of Computer Science*, pages 49–60, 1987.

7 M. de Berg, M. J. van Kreveld, B. J. Nilsson, and M. H. Overmars. Shortest path queries in rectilinear worlds. *Int. J. of Computational Geometry and Applications*, 3(2):287–309, 1992.

8 Mark de Berg. Generalized hidden surface removal. *Computational Geometry*, 5(5):249–276, 1996. doi:10.1016/0925-7721(95)00008-9.

9 Erik D. Demaine, Joseph S. B. Mitchell, and Joseph O'Rourke. The open problems project. http://maven.smith.edu/~orourke/TOPP/.

10 Wei Ding. On computing integral minimum link paths in simple polygons. In *EuroCG*, 2008.

11 Alon Efrat, Leonidas J. Guibas, Olaf A. Hall-Holt, and Li Zhang. On incremental rendering of silhouette maps of a polyhedral scene. *Computational Geometry: Theory and Applications*, 38(3):129–138, 2007.

12 Dania El-Khechen, Muriel Dulieu, John Iacono, and Nikolaj van Omme. Packing 2×2 unit squares into grid polygons is np-complete. In *CCCG 2009*.

13 Leila De Floriani and Paola Magillo. Algorithms for visibility computation on terrains: a survey. *Environment and Planning B: Planning and Design*, 30(5):709–728, 2003. URL: http://EconPapers.repec.org/RePEc:pio:envirb:v:30:y:2003:i:5:p:709-728.

14 James D. Foley, Richard L. Phillips, John F. Hughes, Andries van Dam, and Steven K. Feiner. *Introduction to Computer Graphics*. Addison-Wesley Longman Publishing Co., Inc., 1994.

15 Subir Kumar Ghosh. Computing the visibility polygon from a convex set and related problems. *Journal of Algorithms*, 12(1):75–95, 1991.

16 Subir Kumar Ghosh, Partha P. Goswami, Anil Maheshwari, Subhas C. Nandy, Sudebkumar Prasant Pal, and Swami Sarvattomananda. Algorithms for computing diffuse reflection paths in polygons. *The Visual Computer*, 28(12):1229–1237, 2012. doi:10.1007/s00371-011-0670-z.

17 J. E. Goodman and J. O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC Press series on discrete mathematics and its applications. Chapman & Hall/CRC, 2004.

18 Leonidas J. Guibas, J. Hershberger, D. Leven, Micha Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

19 Leonidas J. Guibas, John Hershberger, Joseph S. B. Mitchell, and Jack Snoeyink. Approximating polygons and subdivisions with minimum link paths. In *Proceedings of the 2nd International Symposium on Algorithms*, ISA'91, pages 151–162, London, UK, UK, 1991. Springer-Verlag.

20 J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4:63–98, 1994.

21 John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Computational Geometry: Theory and Applications*, 4:63–97, 1994.

22 Ferran Hurtado, Maarten Löffler, Inês Matos, Vera Sacristán, Maria Saumell, Rodrigo I Silveira, and Frank Staals. Terrain visibility with multiple viewpoints. In *Proc. 24th International Symposium on Algorithms and Computation*, pages 317–327. Springer, 2013.

23 Simon Kahan and Jack Snoeyink. On the bit complexity of minimum link paths: Superquadratic algorithms for problems solvable in linear time. *Computational Geometry: Theory and Applications*, 12(1-2):33–44, 1999.

24 Irina Kostitsyna, Maarten Löffler, Valentin Polishchuk, and Frank Staals. On the complexity of minimum-link path problems. http://arxiv.org/abs/1603.06972, 2016.

25 J. S. B. Mitchell, G. Rote, and G. Woeginger. Minimum-link paths among obstacles in the plane. *Algorithmica*, 8(1):431–459, 1992.

26 Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Computational Geometry: Theory and Applications*, 47(6):651–667, 2014. doi:10.1016/j.comgeo.2013.12.005.

27 Esther Moet. *Computation and complexity of visibility in geometric environments.* PhD thesis, Utrecht University, 2008.

28 Christine Piatko. *Geometric bicriteria optimal path problems.* PhD thesis, Cornell University, 1993.

29 D. Prasad, S. P. Pal, and T. Dey. Visibility with multiple diffuse reflections. *Computational Geometry: Theory and Applications*, 10:187–196, 1998.

30 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Recognizing string graphs in NP. *J. Comput. Syst. Sci.*, 67(2):365–380, 2003. doi:10.1016/S0022-0000(03)00045-X.

31 A. James Stewart. Hierarchical visibility in terrains. In *Eurographics Rendering Workshop*, June 1997.

32 Subhash Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Computer Vision, Graphics and Image Processing*, 35(1):99–110, 1986. doi:10.1016/0734-189X(86)90127-1.

33 Giovanni Viglietta. Face-guarding polyhedra. In *CCCG 2011*.

A Quasilinear-Time Algorithm for Tiling the Plane Isohedrally with a Polyomino

Stefan Langerman[*1] **and Andrew Winslow**[2]

1 **Département d'Informatique, Université Libre de Bruxelles, Bruxelles, Belgium**
stefan.langerman@ulb.ac.be

2 **Département d'Informatique, Université Libre de Bruxelles, Bruxelles, Belgium**
andrew.winslow@ulb.ac.be

Abstract

A plane tiling consisting of congruent copies of a shape is *isohedral* provided that for any pair of copies, there exists a symmetry of the tiling mapping one copy to the other. We give a $O(n \log^2 n)$-time algorithm for deciding if a polyomino with n edges can tile the plane isohedrally. This improves on the $O(n^{18})$-time algorithm of Keating and Vince and generalizes recent work by Brlek, Provençal, Fédou, and the second author.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Plane tiling, polyomino, boundary word, isohedral

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.50

1 Introduction

The 18th of Hilbert's 23 famous open problems posed in 1900 [20] concerned *isohedral* tilings of polyhedra where every pair of copies in the tiling has a symmetry of the tiling that maps one copy to the other (see Figure 1). Hilbert asked for an example of an *anisohedral* polyhedron that admits a tiling, but no isohedral tilings.

Reinhardt [30] was the first to give an example of an anisohedral polyhedron. Along with this example, Reinhardt also stated that a proof that no anisohedral polygons exist was forthcoming, a claim thought to be supported by Hilbert [15]. In fact, Reinhardt (and Hilbert?) were mistaken: no such proof is possible and Heesch provided the first counterexample in 1935 [18] (see Figure 2).

In the 1963, Heesch and Kienzle [19] provided the first complete classification of isohedral tilings. This classification was given as nine *boundary criteria*: conditions on a polygon's boundary that, if satisfied, imply an isohedral tiling and together form a necessary condition for isohedral polygons. Each boundary criterion describes a factorization of the boundary into a specific number of intervals with given properties, e.g., an interval is rotationally symmetric or two intervals are translations of each other. Special cases of this classification have been rediscovered since, including the criterion of Beauquier and Nivat [3] and Conway's criterion, attributed to John H. Conway by Gardner [11, 32] (see Figure 3).

While a complete classification of isohedral tilings exists, many problems in tiling classification and algorithmics remain open. For instance, complete classifications of pentagons

[*] Directeur de recherches du F.R.S.-FNRS.


© Stefan Langerman and Andrew Winslow;
licensed under Creative Commons License CC-BY

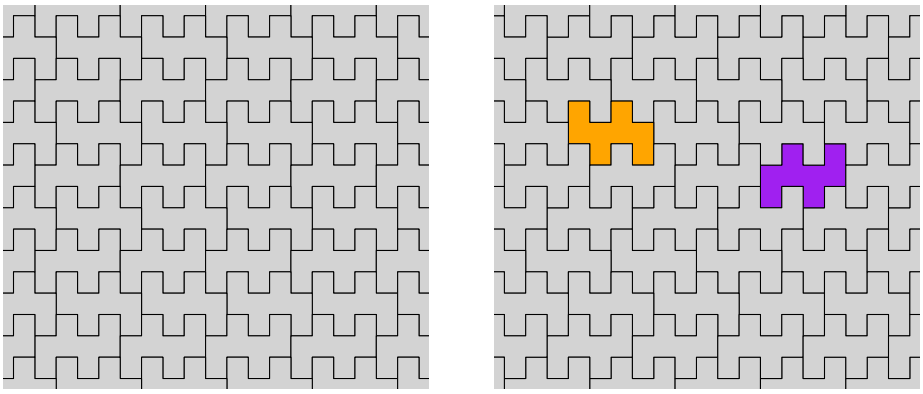


Figure 1 Isohedral (left) and anisohedral (right) tilings of a polyomino. There is no symmetry of the right tiling mapping one colored tile to the other.

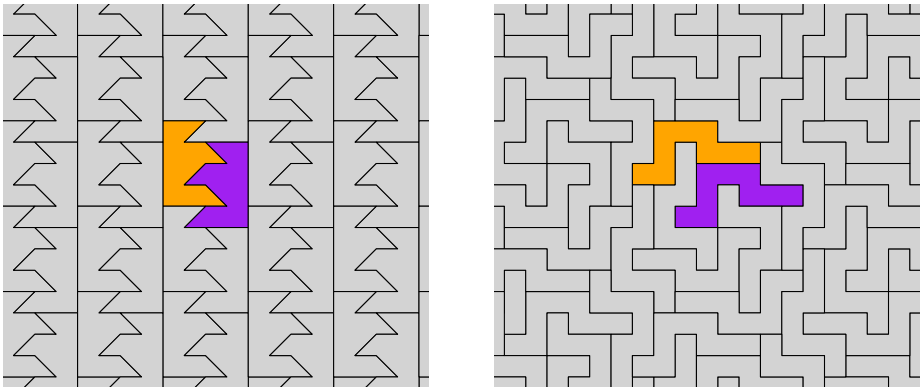


Figure 2 The anisohedral polygon of Heesch [18] and an anisohedral polyomino of Rhoads [31]. There is no symmetry of either tiling mapping one colored tile to the other.

that tile the plane were claimed as early as 1968 [23], but additional pentagons have been discovered as recently as 2015 [26]. The existence of an algorithm for deciding if a polyomino tiles the plane is a longstanding open problem [12, 13], as is the existence of a polygon that tiles only without symmetry [34].

One of the most successful lines of work in tiling algorithmics was initiated by Wijshoff and van Leeuwen [35], who considered tiling the plane using *translated* copies of a polyomino (isohedrally or otherwise). They proved that deciding whether a polyomino admits such a tiling is possible in polynomial time. Their algorithm was subsequently improved by Beauquier and Nivat [3], who gave a simple boundary criterion for polyominoes that admit such a tiling. Subsequent application of more advanced algorithmic techniques led to a sequence of improved algorithms by Gambini and Vuillon [10], Provençal [29], Brlek, Provençal, and Fédou [5], and the second author [36], who gave an optimal $O(n)$-time algorithm, where n is the number of edges on the polyomino's boundary.

The boundary criterion of Beauquier and Nivat matches one of the criteria of Heesch and Kienzle, implying that this problem is a special case of deciding if a polyomino is isohedral. The general problem of isohedrality was proved decidable in 1999 by Keating and Vince [22],

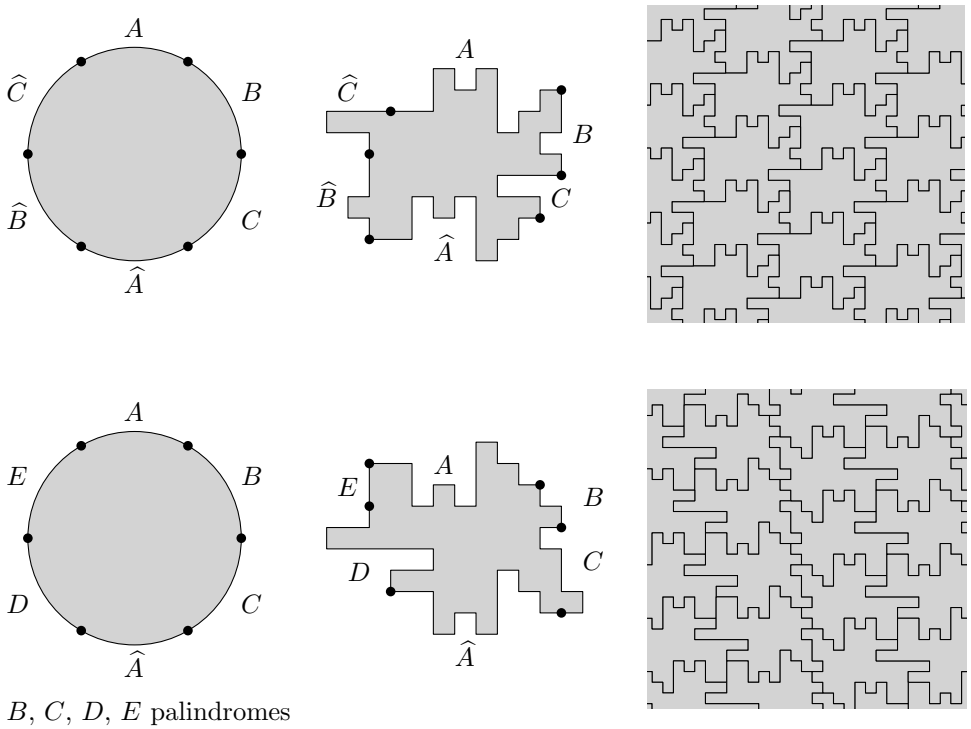


Figure 3 Two of seven boundary criteria characterizations of isohedral tilings. These criteria were given by Beauquier and Nivat [3] (top) and John H. Conway [11] (bottom). Precise definitions are given in Sections 2.

who gave a matrix-based algorithm running in $O(n^{18})$ time. Their algorithm does not make use of boundary criteria, which we note yields a straightforward $O(n^6)$-time algorithm.

Here we give a $O(n \log^2 n)$-time algorithm for deciding if a polyomino is isohedral. The algorithm uses the original boundary characterization of Heesch and Kienzle [19] to decompose the problem into seven subproblems, each of recognizing whether a polyomino's boundary admits a factorization with a specific form. Structural and algorithmic results on a variety of word problems are used, extending the approach of [36] to factorizations of six additional forms. The algorithm also finds a witness tiling and is easily extended to other classes of lattice shapes, e.g. polyhexes and polyiamonds.

2 Definitions

Although the main result of the paper concerns geometric tilings, the proof is entirely described using *words*, also called *strings*. We use the term "word" for consistency with terminology in previous work on tilings of polyominoes.

Polyomino and Tiling

A *polyomino* is a simply connected polygon whose edges are unit length and parallel to one of two perpendicular lines. Let $\mathcal{T} = \{T_1, T_2, \dots\}$ be an infinite set of finite simply connected closed sets of $\mathbb{R}^2$. Provided the elements of $\mathcal{T}$ have pairwise disjoint interiors and cover the Euclidean plane, then $\mathcal{T}$ is a *tiling* and the elements of $\mathcal{T}$ are called *tiles*. Provided every $T_i \in \mathcal{T}$ is congruent to a common shape T, then $\mathcal{T}$ is *monohedral* and T is the *prototile* of $\mathcal{T}$.

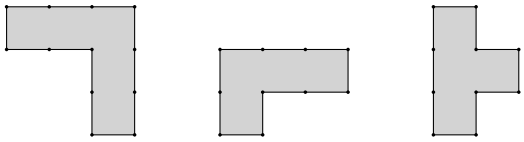


Figure 4 Polyominoes with (circular) boundary words $\mathbf{ur^3d^3lu^2l^2}$, $\mathbf{u^2r^3dl^2dl}$, and $\mathbf{u^3(rd)^2ldl}$ (from left to right).

In this case, T is said to *have* a tiling. A monohedral tiling is also *isohedral* provided, for every pair of elements $T_i, T_j \in \mathcal{T}$, there exists a symmetry of $\mathcal{T}$ that maps T_i to T_j. Otherwise the tiling is *anisohedral*.

Letter

A *letter* is a symbol $x \in \Sigma = \{\mathbf{u}, \mathbf{d}, \mathbf{l}, \mathbf{r}\}$ representing the directions up, down, left and right. The $\Theta°$*-rotation* of a letter x, written $t_\Theta(x)$, is defined as the letter obtained by rotating x counterclockwise by $\Theta°$, e.g., $t_{270}(\mathbf{u}) = \mathbf{r}$. A special case of $\Theta°$-rotations is the *complement* of a letter, written $\overline{x}$ and defined as $\overline{x} = t_{180}(x)$.

Word and Boundary Word

A *word* is a sequence of letters and the *length* of a word W, denoted $|W|$, is the number of letters in W. For an integer $i \in \{1, 2, \ldots, |W|\}$, $W[i]$ refers to the ith letter of W and $W[-i]$ refers to the ith from the last letter of W. The notation W^i denotes the repetition of a word i times. In this work two kinds of words are used: non-circular words and circular words (defining the boundaries of polyominoes). A word is *non-circular* if it has a first letter, and *circular* otherwise. For a circular word W, an arbitrary but fixed assignment of the letter $W[1]$ may be used, resulting in a non-circular *shift* of W. The *boundary word* of a polyomino P, denoted $\mathcal{B}(P)$, is the circular word of letters corresponding to the sequence of directions traveled along cell edges during a clockwise traversal of the polyomino's boundary (see Figure 4).

Rotation and complement

The rotation (or complement) of a word W, written $t_\Theta(W)$ (or $\overline{W}$), is the word obtained by replacing each letter in W with its rotation (or complement). The *reverse* of a word W, written $\widetilde{W}$, are the letters of W in reverse order. The *backtrack* of a word W is denoted $\widehat{W}$ and defined as $\widehat{W} = \widetilde{\overline{W}}$.

Factor

A *factor of* W is a contiguous sequence X of letters in W, written $X \preceq W$. For integers $1 \leq i, j \leq |W|$ with $i \leq j$, $W[i..j]$ denotes the factor of W from $W[i]$ to $W[j]$, inclusive. A factor X *starts* or *ends* at $W[i]$ if $W[i]$ is the first or last letter of X, respectively. Two factors $X, Y \preceq W$ may refer the same letters of W or merely have the same letters in common. In the former case, X and Y are *equal*, written $X = Y$, while in the latter, X and Y are *congruent*, written $X \equiv Y$. For instance, if $W = \mathbf{uuulruuu}$ then $W[1..3] \equiv W[6..8]$. A *factorization* of W is a partition of W into consecutive factors F_1 through F_k, written $W = F_1 F_2 \ldots F_k$.

Prefix, suffix, and center

A factor $X \preceq W$ is a *prefix* if X starts at $W[1]$, written $X \preceq_{\text{pre}} W$. Similarly, $X \preceq W$ is a *suffix* if X ends at $W[-1]$, written $X \preceq_{\text{suff}} W$. The factor $X \preceq W$ such that $W = UXV$, $|U| = |V|$, and $|X| \in \{1, 2\}$ is the *center of* W. Similar definitions for words are defined equivalently, e.g., a word is a *prefix* of another word provided it is congruent to a prefix factor of that word.

Period, composite, and primitive

A word X *is a period of* W provided W is congruent to a prefix of X^k for some $k \geq 0$ (introduced by [24]). Alternatively, X is a prefix of W and $W[i] = W[i + |X|]$ for all $1 \leq i \leq |W| - |X|$. A word X is *composite* provided there exists a subword Y such that $X = Y^k$ for some $k \geq 2$, and otherwise is *primitive*.

Θ-drome and gapped mirror.

A word X is a Θ-*drome* provided $X = Y t_{\Theta+180}(\widetilde{Y})$, e.g., a *palindrome* is a 180-drome. Such a factor is *admissible* provided $W = XU$ with $U[-1] \neq t_{\Theta+180}(U)[1]$.[1]

A pair of disjoint factors $X, Y \preceq W$ is a *gapped mirror* provided $X \equiv \widehat{Y}$. Such a pair X, Y is *admissible* provided $W = XUYV$ with $U[1] \neq \overline{U[-1]}$, $V[1] \neq \overline{V[-1]}$.

3 Proof Overview

The remainder of the paper is dedicated to proving the following theorem:

▶ **Theorem 1.** *Let P be a polyomino with $|\mathcal{B}(P)| = n$. It can be decided in $O(n \log^2 n)$ time if P has an isohedral tiling (of the plane).*

Here we survey some of the ideas involved in the proof. The proof starts with a list of the boundary word factorization forms that together characterize the polyominoes capable of tiling isohedrally. These are found in the bordered subregion[2] of Table 10 of [19][3] excluding the two types of isohedral tilings that use 60° and 120° rotations of the shape. The factorizations can be cross-verified using the incidence and adjacency symbols of a more detailed classification of isohedral tiling types of Grünbaum and Shephard [14], and correspond to the isohedral types IH 1, 4, 28, 2, 3, 5, and 6 in this classification. The factorization forms are:

- Translation: $ABC\widehat{A}\widehat{B}\widehat{C}$.
- Half-turn: $W = ABC\widehat{A}DE$ with B, C, D, E palindromes.
- Quarter-turn: $W = ABC$ with A a palindrome and B, C 90-dromes.
- Type-1 reflection: $W = AB f_\Theta(B)\widehat{A}C f_\Phi(C)$ for some Θ, Φ.
- Type-2 reflection: $W = ABC\widehat{A} f_\Theta(C) f_\Theta(B)$.
- Type-1 half-turn-reflection: $W = ABC\widehat{A}D f_\Theta(D)$ with B, C palindromes.
- Type-2 half-turn-reflection: $W = ABCD f_\Theta(B) f_\Phi(D)$ with A, C palindromes and $\Theta° - \Phi° = \pm 90°$.

[1] In this work, several types of factors and pairs of factors have restricted "admissible" versions. Intuitively, admissible versions are maximal in a natural sense for each type. For instance, a Θ-drome is admissible if it is the longest Θ-drome with its center.

[2] A translation of the caption of Table 10: "The strong border contains 9 major types, from which the others can be thought of as emerging by shrinking lines or line pairs."

[3] Reproduced on page 326 of [33].

The second author [36] gave a $O(n)$-time algorithm for deciding if a boundary word of length n has a translation factorization. Section 4 gives a $O(n \log^2 n)$-time algorithm for deciding if a boundary word of length n has a half-turn factorization. For the remaining boundary word factorization forms, $O(n \log n)$-time or faster algorithms also exist. Due to space constraints, these algorithms are omitted. Brlek, Koskas, and Provençal [4] provide a $O(n)$-time algorithm for deciding if a given circular word is the boundary word of a polyomino, and we assume for the remainder of the paper that the input is guaranteed to be the boundary word of a polyomino and thus simple. This assumption of simplicity is used to prove that factors and pairs of factors in a factorization are *admissible*: maximal in a natural sense for each factor (pair) type. E.g., for half-turn factorizations:

▶ Lemma 17 (restated). Let P be a polyomino and $\mathcal{B}(P) = ABC\widehat{A}DE$ with B, C, D, E palindromes. Then the gapped mirror pair A, $\widehat{A}$ and palindromes B, C, D, E are admissible.

For various factors and factor pairs, this implies there are $O(n)$ or $O(n \log n)$ candidate factors for these elements of a factorization, and they can be computed in similar time. Note that this alone is not sufficient to solve these problems in quasi-linear time. Without additional structural results, attempting to combine a quasi-linear number of factors into a factorization with one of the 6 forms is at least as hard as searching for cycles of fixed length between 3 and 6 in a graph with n vertices and $O(n)$ edges, only known to admit a $O(n^{1.67})$-time algorithm [1]. Additional structural results must be used, such as the following for half-turn factorizations:

▶ Lemma 7 (restated). *The prefix palindrome factorization of a word W has the form $W = X_1^{r_1} X_2^{r_2} \dots X_m^{r_m} Q$ with:*

- *Every X_i primitive.*
- *For all $3 \le i \le m$, $\sum_{j=1}^{i-2} |X_j^{r_j}| \le |X_i|$ and thus $m = O(\log |W|)$.*

Such results allow more efficient "batch processing" of factors to achieve quasi-linear running time. It can also be seen by cursory examination that each algorithm returns affirmatively only once witness factorization is found. Witness factorizations define the set of boundary intervals shared by pairs of neighboring tiles in an isohedral tiling, thus the algorithm can also return a witness isohedral tiling if desired.

4 Half-Turn Factorizations

▶ **Definition 2.** A *half-turn factorization* of a boundary word W has the form $W = ABC\widehat{A}DE$ with B, C, D, E palindromes.

4.1 Prefix palindrome factorizations

▶ **Definition 3.** Let W be a word. A factorization $W = F_1 F_2 \dots F_{n+1}$ is a *prefix palindrome factorization* of W provided that the set of prefix palindromes of W is $\{F_1 F_2 \dots F_i : 1 \le i \le n\}$.

▶ **Lemma 4.** *Let $W = PX$ with P, W palindromes and $0 < |P| < |W|$. Then W has a period of length $|X|$. Furthermore, if X is composite, then W has a prefix palindrome longer than P.*

Proof. Since $W = PX$ and P, W are palindromes, $W = \widetilde{W} = \widetilde{PX} = \widetilde{X}\widetilde{P} = \widetilde{X}P$. So P is a prefix of $\widetilde{X}P$ and so $\widetilde{X}$ is a period of P and of $W = \widetilde{X}P$. Since $\widetilde{X}$ is a period of W and $|\widetilde{X}| < |W|$, there exist words Y, Z such that $\widetilde{X} = ZY$, $W = (ZY)^p Z$, and $P = (ZY)^{p-1} Z$ for some $p \ge 1$. So $X = YZ$ and since W is a palindrome, Y and Z are palindromes.

If X is composite, then $X = G^k$ for some $k \geq 2$. So there exist words G_1, G_2 such that $G = G_1G_2$, $Y = (G_1G_2)^iG_1$, $Z = G_2(G_1G_2)^{k-i-1}$. Since Y and Z are palindromes, G_1 and G_2 are palindromes.

Now we construct a prefix palindrome of W, called Q, that is longer than P. Without loss of generality, assume $|G_2| > 0$. Then there are two possibilities for the values of $|G_2|$ and $|Z|$:

1. $|G_2| < |Z|$ and we let $Q = \widetilde{X}^pG_2$.
2. $|G_2| = |Z|$ and we let $Q = PG_2$.

In the first case, $Q = \widetilde{X}^pG_2 = (G_1G_2)^pG_2 = (G_2G_1)^{kp}G_2$, so Q is a palindrome. Also, $W = (ZY)^pZ$ and G_2 is a prefix of Z, so Q is a prefix of W and $|P| = |W| - |X| = |W| - |ZY| = p|ZY| - |Y| < |Q| = p|ZY| + |G_2| < p|ZY| + |Z| = |W|$.

In the second case, $Y = G_2^i$ and $Z = G_2^{k-i}$. So $Q = PG_2 = (ZY)^{p-1}ZG_2 = G_2^{kp-k}G_2^{k-i}G_2 = G_2^{kp-i+1}$ and Q is a palindrome. Also, $W = PX$ and G_2 is a prefix of X, so Q is a prefix of W and $|P| < |Q| = |P| + |G_2| < |P| + |X| = |W|$. ◀

The following is a well-known result; see Chapter 2 of Crochemore and Rytter [6].

▶ **Lemma 5** (Fine and Wilf's theorem [8]). *Let W be a word with periods of length p and q. If $p + q \leq |W|$, then W also has a period of length $\gcd(p, q)$.*

▶ **Lemma 6.** *Let $P_1, P_2, \ldots, P_m$ be the set of prefix palindromes of a word with $0 < |P_1| < |P_2| < \cdots < |P_m|$. Then for any $1 \leq i \leq m - 2$, either $|P_{i+1}| - |P_i| = |P_{i+2}| - |P_{i+1}|$ or $|P_i| + |P_{i+1}| < |P_{i+2}|$.*

Proof. Let $P_{i+1} = P_iX_i$ and $P_{i+2} = P_iX_iX_{i+1}$.

Since there are no prefix palindromes of length between $|P_i|$ and $|P_{i+1}|$ or $|P_{i+1}|$ and $|P_{i+2}|$, Lemma 4 implies X_i and X_{i+1} are primitive and P_{i+1} and P_{i+2} have periods of length $|X_i|$ and $|X_{i+1}|$, respectively. Since P_{i+1} is a prefix of P_{i+2}, it also has a period of length $|X_{i+1}|$.

The lemma permits $|X_i| = |X_{i+1}|$, so assume $|X_i| \neq |X_{i+1}|$. If $|X_{i+1}| \leq |P_i|$, then P_{i+1} has periods of length $|X_i|$ and $|X_{i+1}|$ with $|X_{i+1}| + |X_i| \leq |P_{i+1}|$. Then by Lemma 5, P_{i+1} has a period of length $\gcd(|X_i|, |X_{i+1}|)$. This length must be at least $|X_i|$ and $|X_{i+1}|$, otherwise X_i or X_{i+1} is not primitive. So $|X_i| = |X_{i+1}|$. Otherwise, $|X_{i+1}| > |P_i|$ and so $|P_i| + |P_{i+1}| < |X_{i+1}| + |P_{i+1}| = |P_{i+2}|$. ◀

The next lemma is a strengthening of similar prior results by Apostolico, Breslauer, and Galil [2], I et al. [21], and Matsubara et al. [27].

▶ **Lemma 7** (Prefix Palindrome Factorization Lemma). *The prefix palindrome factorization of a word W has the form $W = X_1^{r_1}X_2^{r_2}\ldots X_m^{r_m}Q$ with:*

- *Every X_i primitive.*
- *For all $3 \leq i \leq m$, $\sum_{j=1}^{i-2} |X_j^{r_j}| \leq |X_i|$ and thus $m = O(\log |W|)$.*

Proof. We give a constructive proof. Let $P_1, P_2, \ldots, P_n$ be the set of prefix palindromes of W with $|P_1| < |P_2| < \cdots < |P_n|$.

Let W_i be the word such that $P_{i+1} = P_iW_i$ and let Q be the word such that $W = P_nQ$. So W has a prefix palindrome factorization $W_1W_2\ldots W_nQ$. By Lemma 4, every W_i is primitive. Moreover, by Lemma 6 either $|W_i| = |W_{i+1}|$ or $|P_i| < |W_{i+1}|$ for every $1 \leq i \leq n - 2$.

Suppose $|W_i| = |W_{i+1}|$. By Lemma 6, P_i and P_{i+1} have a common period and thus $W_i = W_{i+1}$. More generally, if $|W_i| = |W_{i+1}| = \cdots = |W_{i+c}|$, then $W_i = W_{i+1} = \cdots = W_{i+c}$. If $|P_i| < |W_{i+1}|$, then $|W_{i+1}| > |P_i| = \sum_{j=1}^{i-1} |W_j|$. So the factorization $W_1W_2\ldots W_nQ$ can

be rewritten as $X_1^{r_1}X_2^{r_2}\dots X_m^{r_m}Q$ with the property that $|X_i| \geq \sum_{j=1}^{i-2}|X_j|^{r_j}$. So for all $i \geq 4$, $2|X_{i-3}| < |X_i|$ and thus $m = O(\log|W|)$. ◀

Such a factorization can be stored using $O(\log|W|)$ space by simply storing $|X_i|$ and r_i for each i. Additional observations can be used prove that $|W|$ prefix palindrome factorizations of the suffixes of a word W can be computed in optimal time:

▶ **Lemma 8.** *The prefix palindrome factorizations of all shifts of a circular word W can be computed in $O(|W|\log|W|)$ total time.*

Proof. Lemma 9 of [21] states that the prefix palindrome factorization of a non-circular word xY can be computed in $O(\log|Y|)$ time given the factorization of Y. Thus the factorizations of non-circular word WW can be enumerated in $O(|W|\log|W|)$ time, beginning with $Y = W[-1]$. Every shift of word W is a subword of the non-circular word WW, and the computed factorizations can be trimmed in $O(\log|W|)$-time per factorization to be the factorizations of shifts of W. ◀

Identical results, including a suffix palindrome factorization lemma, clearly hold for suffix palindromes as well.

4.2 Algorithm

The main idea is to iterate over all pairs of adjacent letters and guess the form of the palindromes D and E in both directions from that location. Specifically, guess what repeated factor $X_i^{r_i}$ terminates them in their prefix and suffix palindrome factorizations. Then try to complete the factorization using Lemma 16, which decides if it is possible to rewrite a given portion of the boundary as $L^bABC\widehat{A}R^c$ with B, C palindromes and b, c in some range. The results leading up to Lemma 16 provide the necessary structure to achieve this goal. In particular, Lemma 14 shows how to decompose a word into two palindromes, and Lemmas 9 through 11 yield fast detection of a factorization of the form BCR^k with B, C palindromes.

▶ **Lemma 9.** *Let W be a word with subwords L, R such that $W = LR^r$ and $R \not\preceq_{\text{suff}} L$. Let P_1, P_2 be palindromes such that $W = P_1P_2R^k$ with $|L| \leq |P_1|$. Then there exists a palindrome P_2' and integer k' such that $W = P_1P_2'R^{k'}$ with $|P_2'| < |R|$.*

Proof. Since $|L| \leq |P_1|$, P_2 is a a suffix of R^i for some minimal i. If $i \leq 1$, then either $|P_2| = |R|$ (and $|P_2'| = 0$, $k' = k+1$) or $|P_2| < |R|$ and the claim is satisfied. If $i \geq 2$, then there exist words Y, Z with $|Y| > 0$ such that $R = YZ$ and $P_2 = Z(YZ)^{i-1}$.

Let $P_2' = Z$ and $k' = i - 1 + k$. So $W = P_1P_2R^k = P_1P_2'(YP_2')^{i-1}R^k = P_1P_2'R^{k'}$. Since $|YZ| = |YP_2'| = |R|$ and $|Y| > 0$, it follows that $|P_2'| < |R|$. Since $P_2 = Z(YZ)^{i-2}YZ$, $Z = P_2'$ is a palindrome. ◀

▶ **Lemma 10** (Lemma C4 of [9])**.** *If a word $X_1X_2 = Y_1Y_2 = Z_1Z_2$ with X_2, Y_1, Y_2, Z_1 palindromes. Then X_1 and Z_2 are palindromes.*

▶ **Lemma 11.** *Let R be a primitive word and let $W = LR^r$. There is a set of integers H with $|H| = O(\log|W|)$ such that $W = P_1P_2R^k$ if and only if it does so with $|LR^h| \leq |P_1| \leq |LR^{h+1}|$ for some $h \in H$. Moreover, given $|R|$ and the prefix palindrome factorization of W, H can be computed in $O(\log|W|)$ time.*

Proof. We may assume $|P_2| < |R|$ by Lemma 9. Consider the prefix palindrome factorization of W as described in Lemma 7. Any solution P_1 ends with one of the repeating subwords X_i of the factorization. There are three cases: $|X_i| < |R|$, $|X_i| > |R|$, and $|X_i| = |R|$.

Case 1: $|X_i| < |R|$. We claim that if $|X_i| < |R|$, then $X_i^{r_i}$ overlaps R^r in at most two repetitions of R (and there are at most three values of h). Assume, for the sake of contradiction, that R^2 is a subword of $|X_i^{r_i}|$. Then R^2 is a word of length at least $|X_i| + |R|$ with periods of length $|X_i|$ and $|R|$. So by Lemma 5, R has a period of length $\gcd(|X_i|, |R|) \leq |X_i| < |R|$, a contradiction.

Case 2: $|X_i| > |R|$. We claim that if $|X_i| > |R|$, then X_i cannot repeat in R^r (and there are at most two values of h). Assume, for the sake of contradiction, that X_i^2 is a subword of R^r. Then by Lemma 5, X_i has a period of length $\gcd(|X_i|, |R|) \leq |R| < |X_i|$. So by Lemma 4, the factorization given was not a prefix palindrome factorization, a contradiction.

Case 3: $|X_i| = |R|$. We claim that if $|X_i| = |R|$, then $h = 0$ suffices. Suppose $|LR^h| \leq |P_1| \leq |LR^{h+1}|$ for some $h \geq 1$. By Lemma 4, P_1 has a period of length $|X_i| = |R|$. Let Y, Z be words such that YZ is a period of P_1 and $|R| - |Z| = |P_2|$. So $P_1 = (YZ)^pY$ for some $p \geq 1$ and Y, Z are palindromes.

Since $LR^{h+1} = (YZ)^pYP_2$ and $|YZYP_2| = 2|R|$, $YZ = YP_2 = R$. So $LR = (YZ)^{p-h}YP_2 = P_1'P_2$, where $P_1' = (YZ)^{p-h}Y$ and thus is a palindrome. So there exists a P_1' with $|LR^0| \leq |P_1'| \leq |LR^1|$.

Computing H. The value of h in case 3 is always 0. For case 1, use the values of h such that $X_i^{r_i}$ contains the last letter of LR^h or LR^{h+1}. For case 2, use the values of h such that the prefix palindrome ending at the unique repetition of X_i has length between $|LR^h|$ and $|LR^{h+1}|$. ◀

▶ **Lemma 12.** *Let R be a primitive word and let $W = LR^r$. Assume that $|R|$ and the prefix and suffix palindrome factorizations of W are given. Then it can be decided in $O(\log |W|)$ time if $W = P_1P_2R^k$ with P_1, P_2 palindromes and $|L| \leq |P_1|$.*

Proof. We may assume $|P_2| < |R|$ by Lemma 9. First, use Lemma 11 to compute a $O(\log |W|)$-sized candidate set of integers H such that a solution exists if and only if $LR^{h+1} = P_1P_2$ with $|LR^h| \leq |P_1| \leq |LR^{h+1}$ for some $h \in H$. By Lemma 10, it suffices to check for such solutions with at least one of the following types of palindromes:

- The longest prefix palindrome of LR^{h+1} with length at least $|LR^h|$.
- The longest suffix palindrome of LR^{h+1} with length less than $|R|$.

Compute the longest prefix palindromes of LR^{h+1} for all values of h in $O(\log |W|)$ total time using a two-finger scan of (1) the prefix palindrome factorization of W and (2) the values of h. Use a second two-finger scan of (1) these prefix palindromes and (2) the suffix palindrome factorization of the last $|R|$ letters of the suffix palindrome factorization of LR^r to search for a solution P_1, P_2.

The longest suffix palindrome of LR^{h+1} (with length less than R) is invariant for h and can be computed in $O(\log |W|)$ time, using the last $|R|$ letters of the suffix palindrome factorization of LR^r. Call the length of this palindrome λ. Use a scan of the prefix palindrome factorization to determine if a prefix palindrome of W has length $|LR^h| - \lambda$ for some value of h. ◀

Next, we develop a second result that is combined with the previous lemma to obtain Lemma 16.

▶ **Lemma 13.** *Let L and R be words such that $L \not\preceq_{\text{pre}} R$. Let A_i be the longest common prefix of $L^{l-i}R$ and a word U. Let $k = l - \lceil |A_0|/|L| \rceil$. Then:*

- *For all i with $0 \le i \le k$, $A_i = A_0$ and $|A_i| < |L^{l-k}|$.*
- *For all i with $k+2 \le i \le l$, $|A_i| - |L^{l-i}| = |A_{k+2}| - |L^{l-(k+2)}|$.*

Proof. Let k be the maximum k such that $|A_k| < |L^{l-k}|$. We show that this value of k has the desired properties, including that $k = l - \lceil |A_0|/|L| \rceil$.

Property 1. Let $0 \le i \le k$. Since A_i is a prefix of $L^{l-i}R$ and $|A_i| < |L^{l-i}|$, A_i is the longest common prefix of L^l and U. This is true for all choices of A_i, and thus all A_i are equal.

Property 2. By definition, $|A_{k+1}| \ge |L^{l-(k+1)}|$ and so $L^{l-(k+1)}$ is a prefix of U. Since $L \not\preceq_{\text{pre}} R$, the length of A_{k+2}, the longest common prefix of $L^{l-(k+2)}R$ and U, must be less than $|L^{l-(k+1)}|$. So $|A_{k+2}| < |L^{l-(k+1)}|$ and L is a period of A_{k+2}.

Let R_1, R_2 be words such that $R = R_1R_2$ and $A_{k+2} = L^{l-(k+2)}R_1$. Then $|R_1| < |L|$ and so R_1 is the longest common prefix of R and L.

So for all $i \ge k+2$, the longest common prefix of $L^{l-i}R$ and $L^{l-(k+1)}$ is $L^{l-i}R_1$. Moreover, since $|L^{l-i}R_1| < |L^{l-(k+1)}|$ and $L^{l-(k+1)}$ is a prefix of U, the longest common prefix of $L^{l-i}R$ and U is also $L^{l-i}R_1$.

Property 3. Finally, we prove that $k = l - \lceil |A_0|/|L| \rceil$. Since $|A_0| = |A_i| < |L^{l-i}|$ for all $0 \le i \le k$, it follows that $|A_0| < |L^{l-k}| = |L|(l-k)$. Then by algebra, $k < l - |A_0|/|L|$.

Since $A_{k+1} \ge |L^{l-(k+1)}|$, it must be that U has a prefix $L^{l-(k+1)}$. So $|A_0| = |A_k| \ge |L^{l-(k-1)}| = |L|(l-k-1)$. Then by algebra $k+1 \ge l - |A_0|/|L|$. So $k < l - |A_0|/|L| \le k+1$. ◀

▶ **Lemma 14.** *Let W be a word and l_1, l_2 integers. Assume the prefix and suffix palindrome factorizations of W are given. Then it can be decided in $O(\log |W|)$ time if there exist palindromes P_1, P_2 such that $W = P_1P_2$ with $|P_1| \ge l_1$, $|P_2| \ge l_2$.*

Proof. By Lemma 10, such a pair of palindromes exist if and only if there exists such a pair such that either P_1 is the longest prefix palindrome of W with $|P_1| \le |W| - l_2$ or P_2 is the longest suffix palindrome of W with $|P_2| \le |W| - l_1$. Scan each factorization in $O(\log |W|)$ time to find these specific palindromes, and then scan the opposite factorizations for a second palindrome to complete W. ◀

The following result comes from a trivial modification of Theorem 9.1.1 of [17] to allow for circular words, namely giving the concatenation of two copies of a corresponding non-circular word as input, and returning ∞ if the output has length more than $\text{lcm}(|X|, |Y|)$.

▶ **Lemma 15** (Theorem 9.1.1 of [17])**.** *Two circular words X, Y can be preprocessed in $O(|X| + |Y|)$ time to support the following queries in $O(1)$-time: what is the longest common factor of X and Y starting at $X[i]$ and $Y[j]$?*

▶ **Lemma 16.** *Let W be a circular word. Let $W = L^lZ$ and $W = YR^r$ such that $L \not\preceq_{\text{pre}} Z$, $R \not\preceq_{\text{suff}} Y$, and L, R are primitive. Assume that the prefix and suffix palindrome factorizations of every shift of W are given. It can be decided in $O((l+r)\log |W|)$ time if there exist positive integers b, c such that $W = L^bAP_1P_2\widehat{A}R^c$ with A, $\widehat{A}$ admissible and P_1, P_2 palindromes.*

Proof. The approach is to iteratively search for a solution for each value of b, carrying out the same algorithm on each value. Then performing an identical, symmetric iteration through the values of c. First assume b is a fixed value and c is not. Let A_i be the longest word such that L^bA_i and $\widehat{A_i}R^i$ are a prefix and suffix of W, respectively. Lemma 13 implies that there exists an integer k such that:

- For all i with $0 \leq i \leq k$, $|L^b A_i|$ is fixed and $|\widehat{A_i} R^i| = |A_0| + |R|i$.
- For all i with $k+2 \leq i \leq l$, $|\widehat{A_i} R^i|$ is fixed and $|L^b A_i| = |L^b| + |A_{k+2}| - |R|(i-(k+2))$.
- k can be computed in $O(1)$ time assuming a data structure allowing $O(1)$ time longest common prefix queries for suffixes of W and $\widehat{W}$ is given.

In other words, k is an efficiently-computable integer that partitions the values of i into three parts: one with a single value ($i = k+1$) and two others where either $|L^b A_i|$ or $|\widehat{A_i} R^i|$ is fixed and the other is a linear set. Handle the case of $i = k+1$ individually by using Lemma 14 to check if the word between $L^b A_{k+1}$ and $\widehat{A_{k+1}} R^{k+1}$ has a factorization into two palindromes. Next, check that all A_i, $\widehat{A_i}$ except $i = 0$ are admissible by verifying $L^b[-1] \neq \overline{R^i}[1]$. Also handle $i = 0$ individually, including checking admissibility. Lemma 12 is used to handle the remaining two cases in $O(\log |W|)$ time each.

Case 1: $0 \leq i \leq k$. In this case, $|L^b A_i|$ is fixed and $|\widehat{A_i} R^i| = |A_0| + |R|i$. If $k-1 < 3$, then handle all three cases individually in $O(\log |W|)$ time using Lemma 14. Otherwise handle only $i = k$ similarly.

By Lemma 13, $A_i = A_0$ for all $0 \leq i \leq k-1$ and $|A_0| < |R^{r-k}|$. So $\widehat{A_0} R^i$ for all $0 \leq i \leq k-1$ and R^r are suffixes of W. Also, for all $i \leq k-1$, $|\widehat{A_0} R^i| \leq |\widehat{A_0} R^{k-1}| \leq |\widehat{A_0} R^k| - |R| \leq |R^r| - |R|$. So for some R' with $|R'| = |R|$, $\{\widehat{A_0} R^i : 0 \leq i \leq k-1\} = \{(R')^i \widehat{A_0} : 0 \leq i \leq k-1\}$. Let L' be such that $W = L^b A_i L' (R')^k \widehat{A_0}$. Then a solution factorization exists if and only if there exist palindromes $P_1 P_2 = L'(R')^{k-i}$ for some $0 \leq i \leq k-1$.

First, search for solutions with $|P_1| \geq |L'|$. We first prove that R' is primitive, allowing Lemma 12 to be invoked. Suppose, for the sake of contradiction, that R' has a period of length p with $p < |R'|$. So $(R')^2$ has periods of length $|R'|$ and p such that $|R'| + p < |(R')^2|$ and so by Lemma 5 has a period of length $\gcd(|R'|, p) < |R'|$. Then since $(R')^2$ contains R as a subword, R also has a period of length p and thus is not primitive, a contradiction.

For solutions with $|P_1| < |L'|$, Lemma 10 implies that it suffices to check for solutions with the longest possible P_1 (longest possible P_2 is handled when performing the symmetric iteration over values of c). To do so, scan the prefix palindrome factorization starting at $L'[|P_1| + 1]$ for a palindrome of length $|L'| - |P_1| + |(R')^{k-i}|$ with $0 \leq i \leq k-1$.

Case 2: $k+2 \leq i \leq l$. In this case, $|\widehat{A_i} R^i|$ is fixed and $|L^b A_i| = |L^b| + |A_{k+2}| - |R|(i-(k+2))$. Then there exists a word R' such that $W = L^b (\widehat{R})^{r-i} A_r R' \widehat{A_i} R^i$ for all $k+2 \leq i \leq l$. Let L' be the suffix of $\widehat{R} A_r$ of length $|R|$. Then $W = L^b A_r (L')^{r-(k+2)} R' \widehat{A_r} R^r$. So there exists a pair of palindromes P_1, P_2 with $W = L^b A_i P_1 P_2 \widehat{A_i} R^i$ for some $k+2 \leq i \leq l$ if and only if $(L')^{r-i} R' = P_1 P_2$ for some $k+2 \leq i \leq l$. This situation is identical to that encountered in the previous case – handle in the same way.

Handling overlap. The description of the algorithm so far has ignored the possibility that $|L^b A_i| + |\widehat{A_i} R^i| > |W|$, i.e., that $L^b A_i$ and $\widehat{A_i} R^i$ "overlap". For the case of $0 \leq i \leq k$, this occurs when $|L^b A_0| + |\widehat{A_0} R^i| > |W|$. Restricting the values of i to satisfy $0 \leq i \leq \min(k, \lfloor (|W| - b|L| - 2|A_0|)/|R| \rfloor)$ ensures that W can be decomposed as claimed. For the case of $k+2 \leq i \leq l$, this occurs when $|L^b R^{r-i} A_r| + |\widehat{A_r} R^r| > |W|$. Restricting the values of i to satisfy $\max(\lceil (2|A_r| + 2r|R| + b|L| - |W|)/|R| \rceil, k+2) \leq i \leq l$ ensures that W can be decomposed as claimed. Check the individually handled cases, namely $i = k, k+1$, for overlap individually.

Running time. The running time of this algorithm is $O((l+r)\log |W|)$, since the amount of time spent for each value of b and c is $O(\log |W|)$ to handle individual values of i and

$O(\log |W|)$ to handle each large case by Lemma 12. However, this assumes a data structure enabling $O(1)$ time common prefix queries on W and $\widehat{W}$. Compute such a data structure in $O(|W|)$ time using Lemma 15. Since $\Omega(|W|)$ time must be spent to decide if a boundary word has a half-turn factorization, such a computation has no additional asymptotic cost. ◀

▶ **Lemma 17.** *Let P be a polyomino and $\mathcal{B}(P) = ABC\widehat{A}DE$ with B, C, D, E palindromes. Then the gapped mirror pair $A, \widehat{A}$ and palindromes B, C, D, E are admissible.*

Proof.

$A, \widehat{A}$ is admissible. It cannot be that $|B| = |C| = 0$, since then $ABC\widehat{A} = A\widehat{A}$ is non-simple. If $BC[1] = \overline{BC}[-1]$, then $|B|, |C| > 0$ and thus $B[-1] = BC[1] = \overline{BC}[-1] = \overline{C}[1]$ and BC is non-simple. So $BC[1] \neq \overline{BC}[-1]$ and by symmetry, $DE[1] \neq \overline{DE}[-1]$. So $A, \widehat{A}$ are admissible.

B, C, D, E are admissible. Consider the pairs of non-equal consecutive letters in W. These pairs come from sets $\mathcal{R} = \{\mathbf{lu}, \mathbf{ur}, \mathbf{rd}, \mathbf{dl}\}$ and $\mathcal{L} = \{\mathbf{ul}, \mathbf{ld}, \mathbf{dr}, \mathbf{ru}\}$, and Proposition 6 of [7] states that the number of pairs from $\mathcal{R}$ is four more than the number from $\mathcal{L}$. Also, any palindrome contains an equal number of consecutive letter pairs from $\mathcal{L}$ and $\mathcal{R}$.

If $|A| = 0$, then W has factorization $W = BCDE$ with the four consecutive-letter pairs from $\mathcal{R}$ not contained in any factor, i.e., for each factor $X \in \{B, C, D, E\}$, $W = XY$ with $Y[-1]X[1], X[-1]Y[1] \in \mathcal{R}$. Since X is a palindrome, $X[-1]Y[1] = X[1]Y[1] \in \mathcal{R}$ and so $Y[1] \neq Y[-1]$. Thus X is admissible.

If $|A| > 0$, then $|BC| > 0$, since otherwise $A[-1]\widehat{A}[1] = A[-1]\overline{A[-1]}$ is a subword and W is non-simple. Without loss of generality, $|B| > 0$. If $|C| = 0$, then $W = BY$ with $Y[1] = \widehat{A}[1] = \overline{A}[-1] \neq A[-1] = Y[-1]$ and B is admissible. If $|C| > 0$, then $C[1] = C[-1] \neq \widehat{A}[1] = A[-1]$. So $W = BY$ with $Y[-1] = A[-1] \neq C[1] = Y[1]$ and so B is admissible. By symmetry, it is also the case that C, D and E are also admissible. ◀

▶ **Theorem 18.** *Let P be a polyomino with $|\mathcal{B}(P)| = n$. It can be decided in $O(n \log^2 n)$ time if $\mathcal{B}(P)$ has a half-turn factorization.*

Proof. First, compute the prefix palindrome factorizations of each shift of W by computing and truncating the prefix palindrome factorizations of the $|W|$ longest suffixes of W using Lemma 8. Similarly compute the suffix palindrome factorization of every shift of W. By Lemma 8, this takes $O(|W| \log |W|)$ total time.

Next, compute the admissible factors (including zero-length factors), i.e., palindromes maximal about their center, by computing and truncating the maximal palindromes of WW output by Manacher's $O(|W|)$-time algorithm [25]. Each admissible factor F is contained in a prefix palindrome factorization as $X_1^{r_1} X_2^{r_2} \dots X_i^j$ with either $1 \leq i \leq m$ and $0 \leq j < r_i$, or $i = m$ and $j = r_m$ if F is the longest prefix palindrome of the word. If $j < r_i$, call $X_i^{r_i}$ the *terminator* of F, otherwise call Q^1 the terminator of F. Either all or none of the prefix palindromes with a given terminator are admissible. Similar definitions and observations apply to suffix palindrome factorizations.

For each admissible factor W, mark the two terminators (one prefix, one suffix) of the factor. Locating the prefix and suffix terminators for each of the $2|W|$ admissible factors takes $O(|W| \log |W|)$ total time.

Without loss of generality, every solution half-turn factorization has $|E| > 0$. Search for half-turn factorizations $ABC\widehat{A}DE$ by iterating over possible first letters of E. By Lemma 17, only solutions with admissible D and E must be considered. This corresponds

to a palindromes D and E starting and ending with a marked terminators $X_s^{r_s}$ and $X_p^{r_p}$, respectively. For each such terminator pair $X_s^{r_s}, X_p^{r_p}$, use Lemma 16 with $L = X_p$, $l = r_p$, $R = \widetilde{X_s}$, $r = r_s$ to check for a partial factorization $ABC\widehat{A}$ to complete the factorization along with D and E with the marked terminator pair. By Lemma 7, X_s and X_p are primitive and by definition of palindrome factorizations, l and r are maximal.

Checking for a partial factorization using Lemma 16, each pair $X_s^{r_s}, X_p^{r_p}$ takes $O((r_s + r_p)\log|W|)$ time, $O(\log|W|)$ time for each admissible factor involved. Moreover, each admissible factor is involved in $O(\log|W|)$ pairs of terminators: $O(\log|W|)$ prefix (suffix) terminators when E (D). So $O(\log^2|W|)$ total time is spent per admissible factor and in total the the algorithm takes $O(|W|\log^2|W|)$ time. ◀

5 Conclusion

This work demonstrates that not just polynomial, but quasilinear-time algorithms exist for deciding tiling properties of a polyomino. It remains to be seen if a linear-time algorithm exists, or whether a super-linear lower bound for one of the factorization forms exists. The slowest algorithm is for half-turn factorizations, so it seems natural to attack this special case first.

▶ **Open Problem 1.** *Can it be decided in $o(n\log^2 n)$-time if a polyomino P with $|\mathcal{B}(P)| = n$ has a half-turn factorization?*

▶ **Open Problem 2.** *Can it be decided in $O(n)$-time if a polyomino P with $|\mathcal{B}(P)| = n$ has an isohedral tiling of the plane?*

For monohedral tilings containing only translations of the prototile, a polyomino has such a tiling only if it has one that is also isohedral [3, 35]. Does this remain true for tilings using other sets of transformations of the prototile? Modifying the anisohedral tile of Heesch [18] (see [16]) proves that the answer is "no" for tilings with reflected tiles, while an example of Rhoads [31] proves that the answer is "no" for tilings with 90° rotations of tiles. This leaves one possibility open:

▶ **Open Problem 3.** *Does there exist a polyomino P that has a tiling containing only translations and 180° rotations of P and every such tiling is anisohedral?*

As mentioned in Section 3, there are isohedral tiling types (characterized by boundary factorizations) that cannot be realized by polyominoes due to angle restrictions. Moreover, the boundary factorization forms here also apply to general polygons, under appropriate definitions of "boundary word". Extending the algorithms presented here to polygons, along with developing algorithms for the remaining boundary factorizations is a natural goal. However, significant challenge remains in efficiently converting a polygon's boundary into a word that can be treated with the approach used here.

▶ **Open Problem 4.** *Can it be decided in $O(n\log^2 n)$ time if a polygon with n vertices has an isohedral tiling of the plane?*

Observe that pairs of tiles in a tiling that can be mapped to each other via a symmetry of the tiling induces a partition of the tiles. Define a tiling to be k-isohedral if the partition has k parts, e.g., an isohedral tiling is 1-isohedral. Thus k-isohedral tilings are a natural generalization of isohedral tilings that allow increasing complexity; specifically, they cannot be characterized by a single boundary factorization. A natural generalization of the problem considered here is as follows:

▶ **Open Problem 5.** *Can it be decided efficiently if a polyomino has a k-isohedral tiling?*

An approach described by Joseph Myers [28] achieves a running time of approximately $n^{O(k^2)}$, though a precise analysis of the running time has not been performed. A fixed-parameter tractable algorithm also may be possible. On the other hand, a proof of NP-hardness is unlikely, since it implies, for each $c \in \mathbb{N}$, the existence of prototiles whose only tilings are k-isohedral for $k \geq c$. Such tiles are only known to exist for $c \leq 10$ [28].

Acknowledgements. The authors wish to thank anonymous reviewers for comments that improved the correctness of the paper.

References

1. N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
2. A. Apostolico, D. Breslauer, and Z. Galil. Parallel detection of all palindromes in a string. *Theoretical Computer Science*, 141:163–173, 1995.
3. D. Beauquier and M. Nivat. On translating one polyomino to tile the plane. *Discrete & Computational Geometry*, 6:575–592, 1991.
4. S. Brlek, M. Koskas, and X. Provençal. A linear time and space algorithm for detecting path intersection. In *DGCI 2009*, volume 5810 of *LNCS*, pages 397–408. Springer Berlin Heidelberg, 2009.
5. S. Brlek, J.-M. X. Provençal, and Fédou. On the tiling by translation problem. *Discrete Applied Mathematics*, 157:464–475, 2009.
6. M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, 1994.
7. A. Daurat and M. Nivat. Salient and reentrant points of discrete sets. *Discrete Applied Mathematics*, 151:106–121, 2005.
8. N. J. Fine and H. S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965.
9. Z. Galil and J. Seiferas. A linear-time on-line recognition algorithm for "Palstar". *Journal of the ACM*, 25(1):102–111, 1978.
10. L. Gambini and L. Vuillon. An algorithm for deciding if a polyomino tiles the plane by translations. *RAIRO – Theoretical Informatics and Applications*, 41(2):147–155, 2007.
11. M. Gardner. More about tiling the plane: the possibilities of polyominoes, polyiamonds, and polyhexes. *Scientific American*, pages 112–115, August 1975.
12. C. Goodman-Strauss. Open questions in tiling. Online, published 2000. URL: `http://comp.uark.edu/~strauss/papers/survey.pdf`.
13. C. Goodman-Strauss. Can't decide? undecide! *Notices of the American Mathematical Society*, 57(3):343–356, 2010.
14. B. Grünbaum and G. C. Shephard. The eighty-one types of isohedral tilings in the plane. *Mathematical Proceedings of the Cambridge Philosophical Society*, 82(2):177–196, 1977.
15. B. Grünbaum and G. C. Shephard. Isohedral tilings of the plane by polygons. *Commentarii Mathematici Helvetici*, 53(1):542–571, 1978.
16. B. Grünbaum and G. C. Shephard. Tilings with congruent tiles. *Bulletin of the American Mathematical Society*, 3:951–973, 1980.
17. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.
18. H. Heesch. Aufbau der ebene aus kongruenten bereichen. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen*, 1:115–117, 1935.
19. H. Heesch and O. Kienzle. *Flächenschluss: System der Formen lückenlos aneinanderschliessender Flachteile*. Springer, 1963.

20 D. Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8(10):437–479, 1902.

21 T. I, S. Sugimoto, S. Inenaga, H. Bannai, and M. Takeda. Computing palindromic factorizations and palindromic covers on-line. In A. S. Kulikov, S. O. Kuznetsov, and P. Pevzner, editors, *CPM 2014*, volume 8486 of *LNCS*, pages 150–161. Springer, Switzerland, 2014.

22 K. Keating and A. Vince. Isohedral polyomino tiling of the plane. *Discrete and Computational Geometry*, 21(4):615–630, 1999.

23 R. B. Kershner. On paving the plane. *The American Mathematical Monthly*, 75(8):839–844, 1968.

24 D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

25 G. K. Manacher. A new linear-time "on-line" algorithms for finding the smallest initial palindrome of a string. *Journal of the ACM*, 22(3):346–351, 1975.

26 C. Mann, J. McLoud-Mann, and D. Von Derau. Convex pentagons that admit i-block transitive tilings. Technical report, arXiv, 2015. URL: `http://arxiv.org/abs/1510.01186`.

27 W. Matsubara, S. Inenaga, A. Ishino, A. Shinohara, T. Nakamura, and K. Hashimoto. Efficient algorithms to compute compressed longest common substrings and compressed palindromes. *Theoretical Computer Science*, 410:900–913, 2009.

28 J. Myers. Polyomino, polyhex, and polyiamond tiling. Online, updated February 2012. URL: `http://www.polyomino.org.uk/mathematics/polyform-tiling/`.

29 X. Provençal. *Combinatoire des mots, géométrie discrète et pavages*. PhD thesis, Université du Québec à Montréal, 2008.

30 K. Reinhardt. Zur zerlegung der euklidischen räume in kongruente polytope. *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, pages 150–155, 1928.

31 G. C. Rhoads. Planar tilings polyominoes, polyhexes, and polyiamonds. *Journal of Copmutational and Applied Mathematics*, 174:329–353, 2005.

32 D. Schattschneider. Will it tile? try the Conway criterion! *Mathematics Monthly*, 53(4):224–233, 1980.

33 D. Schattschneider. *Visions of Symmetry: Notebooks, Periodic Drawings, and Related Work of M. C. Escher*. W. H. Freeman and Company, 1990.

34 J. E. S. Socolar and J. M. Taylor. An aperiodic hexagonal tile. *Journal of Combinatorial Theory, Series A*, 118(8):2207–2231.

35 H. A. G. Wijshoff and J. van Leeuwen. Arbitrary versus periodic storage schemes and tessellations of the plane using one type of polyomino. *Information and Control*, 62:1–25, 1984.

36 A. Winslow. An optimal algorithm for tiling the plane with a translated polyomino. In *26th International Symposium on Algorithms and Computation (ISAAC)*, 2015.

Eliminating Higher-Multiplicity Intersections, II. The Deleted Product Criterion in the r-Metastable Range*

Isaac Mabillard[1] and Uli Wagner[2]

1 **IST Austria, Klosterneuburg, Austria**
imabillard@ist.ac.at
2 **IST Austria, Klosterneuburg, Austria**
uli@ist.ac.at

Abstract

Motivated by Tverberg-type problems in topological combinatorics and by classical results about embeddings (maps without double points), we study the question whether a finite simplicial complex K can be mapped into $\mathbb{R}^d$ without higher-multiplicity intersections. We focus on conditions for the existence of *almost r-embeddings*, i.e., maps $f\colon K \to \mathbb{R}^d$ such that $f(\sigma_1) \cap \cdots \cap f(\sigma_r) = \emptyset$ whenever $\sigma_1, \ldots, \sigma_r$ are pairwise disjoint simplices of K.

Generalizing the classical Haefliger-Weber embeddability criterion, we show that a well-known necessary *deleted product condition* for the existence of almost r-embeddings is sufficient in a suitable *r-metastable range* of dimensions: If $rd \geq (r+1)\dim K + 3$, then there exists an almost r-embedding $K \to \mathbb{R}^d$ if and only if there exists an equivariant map $(K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$, where $(K)^r_\Delta$ is the deleted r-fold product of K, the target $S^{d(r-1)-1}$ is the sphere of dimension $d(r-1)-1$, and $\mathfrak{S}_r$ is the symmetric group. This significantly extends one of the main results of our previous paper (which treated the special case where $d = rk$ and $\dim K = (r-1)k$ for some $k \geq 3$), and settles an open question raised there.

1998 ACM Subject Classification G. Mathematics of Computing

Keywords and phrases Topological Combinatorics, Tverberg-Type Problems, Simplicial Complexes, Piecewise-Linear Topology, Haefliger–Weber Theorem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.51

1 Introduction

Let K be a finite simplicial complex, and let $f\colon K \to \mathbb{R}^d$ be a continuous map.[1] Given an integer $r \geq 2$, we say that $y \in \mathbb{R}^d$ is an **r-fold point** or **r-intersection point** of f if it has r pairwise distinct preimages, i.e., if there exist $y_1, \ldots, y_r \in K$ such that $f(y_1) = \ldots = f(y_r) = y$ and $y_i \neq y_j$ for $1 \leq i < j \leq r$. We will pay particular attention to r-fold points that are **global**[2] in the sense that their preimages lie in r *pairwise disjoint simplices* of K, i.e., $y \in f(\sigma_1) \cap \ldots \cap f(\sigma_r)$, where $\sigma_i \cap \sigma_j = \emptyset$ for $1 \leq i < j \leq r$.

* Research supported by the Swiss National Science Foundation (Project SNSF-PP00P2-138948). We would like to thank Arkadiy Skopenkov for many helpful comments.

[1] For simplicity, throughout most of the paper we use the same notation for a simplicial complex K and its *underlying topological space*, relying on context to distinguish between the two when necessary.

[2] In our previous paper [17], we used the terminology "*r-Tverberg point*" instead of "*global r-fold point*."

© Isaac Mabillard and Uli Wagner;
licensed under Creative Commons License CC-BY

We say that a map $f\colon K \to \mathbb{R}^d$ is an **r-embedding** if it has no r-fold points, and we say that f is an **almost r-embedding** if it has no *global* r-fold points.[3]

The most fundamental case $r = 2$ is that of **embeddings** (=2-embeddings), i.e., injective continuous maps $f\colon K \to \mathbb{R}^d$. Finding conditions for a simplicial complex K to be embeddable into $\mathbb{R}^d$ — a higher-dimensional generalization of graph planarity — is a classical problem in topology (see [27, 33] for surveys) and has recently also become the subject of systematic study from a viewpoint of algorithms and computational complexity (see [23, 22, 9]).

Here, we are interested in necessary and sufficient conditions for the existence of almost r-embeddings. One motivation are *Tverberg-type problems* in topological combinatorics (see the corresponding subsection below). Another motivation is that, in the classical case $r = 2$, embeddability is often proved in two steps: in the first step, the existence of an **almost embedding** (=almost 2-embedding) is established; in the second step this almost embedding is transformed into a honest embedding, by removing *local* self-intersections. Similarly, we expect the existence of an almost r-embedding to be not only an obvious necessary condition but a useful stepping stone towards the existence of r-embeddings and, in a further step, towards the existence of embeddings in certain ranges of dimensions.

The Deleted Product Criterion for Almost r-Embeddings

There is a natural *necessary condition* for the existence of almost r-embeddings. Given a simplicial complex K and $r \geq 2$, the (combinatorial) **deleted r-fold product**[4] of K is defined as

$$(K)^r_\Delta := \{(x_1, \dots, x_r) \in \sigma_1 \times \cdots \times \sigma_r \mid \sigma_i \text{ a simplex of } K, \sigma_i \cap \sigma_j = \emptyset \text{ for } 1 \leq i < j \leq r\}.$$

The deleted product is a regular polytopal cell complex (a subcomplex of the cartesian product), whose cells are products of r-tuples of pairwise disjoint simplices of K.

▶ **Lemma 1** (Necessity of the Deleted Product Criterion). *Let K be a finite simplicial complex, and let $d \geq 1$ and $r \geq 2$ be integers. If there exists an almost r-embedding $f : K \to \mathbb{R}^d$ then there exists an equivariant map*[5]

$$\widetilde{f}\colon (K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1},$$

where $S^{d(r-1)-1} = \left\{(y_1, \dots, y_r) \in (\mathbb{R}^d)^r \mid \sum_{i=1}^r y_i = 0, \sum_{i=1}^r \|y_i\|_2^2 = 1\right\}$, and the symmetric group $\mathfrak{S}_r$ acts on both spaces by permuting components.

Proof. Given $f\colon K \to \mathbb{R}^d$, define $f^r\colon (K)^r_\Delta \to (\mathbb{R}^d)^r$ by $f^r(x_1, \dots, x_r) := (f(x_1), \dots f(x_r))$. Then f is an almost r-embedding iff its image avoids the *thin diagonal* $\delta_r(\mathbb{R}^d) := \{(y, \dots, y) \mid y \in \mathbb{R}^d\} \subset (\mathbb{R}^d)^r$. Moreover, $S^{d(r-1)-1}$ is the unit sphere in the orthogonal complement $\delta_r(\mathbb{R}^d)^\perp \cong \mathbb{R}^{d(r-1)}$, and there is a straightforward homotopy equivalence $\rho\colon (\mathbb{R}^d)^r \setminus \delta_r(\mathbb{R}^d) \simeq S^{d(r-1)-1}$. Both f^r and ρ are equivariant hence so is their composition

$$\widetilde{f} := \rho \circ f^r\colon (K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}. \qquad \blacktriangleleft$$

---

[3] We emphasize that the definitions of global r-fold points and of almost r-embeddings depend on the actual simplicial complex K (the specific triangulation), not just the underlying topological space.

[4] For more background on deleted products and the broader *configuration space/test map* framework, see, e.g., [21] or [39, 40].

[5] Here and in what follows, if X and Y are spaces on which a finite group G acts (all group actions will be from the right) then we will use the notation $F\colon X \to_G Y$ for maps that are **equivariant**, i.e., that satisfy $F(x \cdot g) = F(x) \cdot g$ for all $x \in X$ and $g \in G$).

Our main result is that the converse of Lemma 1 holds in a wide range of dimensions.

▶ **Theorem 2** (Sufficiency of the Deleted Product Criterion in the r-Metastable Range)**.** *Let $m, d \geq 1$ and $r \geq 2$ be integers satisfying*

$$rd \geq (r+1)m + 3. \tag{1}$$

Suppose that K is a finite m-dimensional simplicial complex and that there exists an equivariant map $F : (K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$. Then there exists an almost r-embedding $f\colon K \to \mathbb{R}^d$.

▶ Remark.

a. When studying almost r-embeddings, it suffices to consider maps $f\colon K \to \mathbb{R}^d$ that are *piecewise-linear*[6] (*PL*) and *in general position*.[7]

b. Theorem 2 is trivial for *codimension* $d - m \leq 2$. Indeed, if r, d, m satisfy (1) and, additionally, $d - m \leq 2$ then a straightforward calculation shows that $(r-1)d > rm$, so that a map $K \to \mathbb{R}^d$ in general position has no r-fold points.

c. The special case $r = 2$ of Theorem 2 corresponds to the classical *Haefliger–Weber Theorem* [16, 37], which guarantees that for $2d \geq 3m + 3$ the existence of an equivariant map $(K)^2_\Delta \to_{\mathfrak{S}_2} S^{d-1}$ guarantees the existence of an almost embedding $f\colon K \to \mathbb{R}^d$. An almost embedding can be then be turned into an embedding by a delicate construction of Skopenkov [32] or Weber [37]. The condition $2d \geq 3m + 3$ is often referred to as the **metastable range**; correspondingly, we call Condition (1) the ***r*-metastable range**.

d. Theorem 2 significantly extends one of the main results of our previous paper [18, Thm. 7] and [17, Thm. 3], which treated the special case $(r-1)d = rm$, $d - m \geq 3$. That special case corresponds to the situation when the set Σ^r of global r-fold points is 0-dimensional, i.e., consists of a finite number of points. In the present paper, we deal with the case where Σ^r is of higher dimension.

e. The r-metastable range is very close to the condition $rd > (r+1)m$ that guarantees that a map $f\colon K \to \mathbb{R}^d$ in general position does not have any $(r+1)$-fold points.

Background and Motivation: Topological Tverberg-Type Problems

Tverberg's classical theorem [35] in convex geometry can be rephrased as follows: if $N = (d+1)(r-1)$ then any *affine* map from the N-dimensional simplex σ^N to $\mathbb{R}^d$ has a global r-fold point, i.e., there does not exist an *affine* almost r-embedding of σ^N in $\mathbb{R}^d$.

Bajmoczy and Bárány [2] and Tverberg [15, Problem 84] raised the question whether the conclusion holds true, more generally, for arbitrary continuous maps:

▶ **Conjecture 3** (Topological Tverberg Conjecture)**.** *Let $r \geq 2$, $d \geq 1$, and $N = (d+1)(r-1)$. Then there is no almost r-embedding $\sigma^N \to \mathbb{R}^d$.*

This was proved by Bajmoczy and Bárány [2] for $r = 2$, by Bárány, Shlosman, and Szűcs [4] for all primes r, and by Özaydin [24] for prime powers r, but the case of arbitrary r remained open and was considered a central unsolved problem of topological combinatorics.

There are numerous close relatives and other variants of (topological) Tverberg-type problems and results. These can be seen as *generalized nonembeddability results* or *problems* and typically state that a particular complex K (or family of complexes) does not admit an almost r-embedding into $\mathbb{R}^d$. Well-known examples are the *Colored Tverberg Problem*

[6] Recall that f is PL if there is some subdivision K' of K such that $f|_\sigma$ is affine for each simplex σ of K'.

[7] Every continuous map $g\colon K \to \mathbb{R}^d$ can be approximated arbitrarily closely by PL maps in general position, and if g is an almost r-embedding, then the same holds for any map sufficiently close to g.

[5, 3, 41, 40, 6] and generalized *Van Kampen–Flores-type results* [29, 36]. Theorem 2 provides a general necessary and sufficient condition for topological Tverberg-type results in the r-metastable range.

The topological Tverberg conjecture and the subsequent developments played an important role in the introduction and use of methods from *equivariant topology* in discrete and computational geometry. The prime and prime power cases of Conjecture 3 were proved via Lemma 1, $(\sigma^N)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$. However, this fails in the remaining cases: Özaydin [24, Thm. 4.2] showed that if r is *not* a prime power then there exists an equivariant map $F\colon (\Delta^N)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$.

In the extended abstract of our previous paper [17], we proposed a new approach to the conjecture, based on the idea of combining Özaydin's result with the sufficiency of the deleted product product ([17, Thm 3]) to construct counterexamples, i.e., almost r-embeddings $\sigma^N \to \mathbb{R}^d$, whenever r is not a prime power. At the time we suggested this in [17], there remained what seemed a very serious obstacle to completing this approach: Our theory required the assumption of *codimension* $d - \dim K \geq 3$, which is not satisfied for $K = \sigma^N$.

In a recent breakthrough, Frick [12] was the first to find a way to overcome this "codimension 3 barrier" and to construct counterexamples to the topological Tverberg conjecture for all parameters (d, r) with $d \geq 3r + 1$ and r not a prime power, by a clever reduction (using a combinatorial trick discovered independently in [14, p. 445-446] and [7, Thm. 6.3]) to a suitable lower-dimensional skeleton for which the codimension condition is satisfied and the required almost r-embedding exists by Özaydin's result and ours.

A different solution to the codimension 3 obstacle (based on the notion of *prismatic maps*) is given in the full version of our paper [18], leading to counterexamples for $d \geq 3r$. In joint work with Avvakumov and Skopenkov [1], we recently improved this further and obtained counterexamples for $d \geq 2r$, using an extension (for $r \geq 3$) of [18, Thm. 7] to *codimension 2*.

In conclusion, methods from equivariant topology and the general framework of *configuration spaces* and *test maps* [39, 40] have been very successfully used in discrete and computational geometry. In particular, *equivariant obstruction theory* and, more generally *equivariant homotopy theory*, provide powerful tools for deciding whether suitable test maps exist. However in cases where the existence of a test map does not settle the problem (as with the topological Tverberg conjecture), further geometric ideas are needed. The general philosophy and underlying idea here and in the two companion papers [18, 1] is to complement equivariant methods by methods from *geometric topology*, in particular *piecewise-linear topology*, and we hope that these will find further applications.

▶ Remark (Further Questions and Future Research).

a. *Beyond the r-Metastable Range.* Is condition (1) in Theorem 2 necessary? In the case $r = 2$, it is known that for $d \geq 3$, the Haefliger–Weber Theorem fails outside the metastable range: for every pair (m, d) with $2d < 3m + 3$ and $d \geq 3$, there are examples [20, 31, 11, 30, 13] of m-dimensional complexes K such that $(K)^2_\Delta \to_{\mathfrak{S}_2} S^{d-1}$ but K does not embed into $\mathbb{R}^d$. Moreover, in the case $r = 2$, $m = 2$ and $d = 4$, the examples do not even admit an almost embedding into $\mathbb{R}^4$, see [1].
On the other hand, as remarked above, in [1] the following extension of [18, Thm. 7] is proved: if $r \geq 3$ $d = 2r$, and $m = 2(r - 1)$, then a finite m-dimensional complex K admits an almost r-embedding if and only if there exists an equivariant map $(K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$.
It would be interesting to know whether there is analogous extension (for $r \geq 3$) of Theorem 2 that is nontrivial in codimension $d - m = 2$.

b. *The Planar Case and Hanani–Tutte.* In the classical setting ($r = 2$) of embeddings, the case $d = 2, m = 1$ of *graph planarity* is somewhat exceptional: the parameters lie outside

the (2-fold) metastable range, but the existence of an equivariant map $F\colon (K)^2_\Delta \to_{\mathfrak{S}_2} S^1$ is sufficient for a graph K to be planar, by the *Hanani–Tutte Theorem*[8] [10, 34]. The classical proofs of that theorem rely on *Kuratowski's Theorem*, but recently [25, 26], more direct proofs have been found that do not use forbidden minors. It would be interesting to know whether there is an analogue of the Hanani–Tutte theorem for almost r-embeddings of 2-dimensional complexes in $\mathbb{R}^2$, as an approach to constructing counterexamples to the topological Tverberg conjecture in dimension $d = 2$. We plan to investigate this in a future paper.

Structure of the Paper

The remainder of the paper is devoted to the proof of Theorem 2. By Lemma 1, we only need to show that the existence of an equivariant map $(K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$ implies the existence of an almost r-embedding $K \to \mathbb{R}^d$. Moreover, by Remarks 1 (b) and (d), we may assume, in addition to the parameters being in the r-fold metastable range, that the *codimension* $d - m$ of the image of K in $\mathbb{R}^d$ is at least 3, and that the intersection multiplicity r is also at least 3. Thus, we will work under the following hypothesis:

$$rd \geq (r+1)m + 3, \quad d - m \geq 3, \quad \text{and} \quad r \geq 3. \tag{2}$$

The proof of Theorem 2 is based on two main lemmas: Lemma 5 (*Reduction Lemma*) reduces the situation to a single r-tuple of pairwise disjoint simplices of K, and Lemma 7 (*generalized Weber–Whitney Trick*) solves that reduced situation. In Section 2, we give the precise (and somewhat technical) statements of these lemmas, along with some background, and prove the Reduction Lemma 5. In Section 3, we show how to prove Theorem 2 using these lemmas.

Due to the page limit, the proof of Lemma 7 is omitted from this *extended abstract*; we refer to the full version of this paper [19] for the details.

2 The Two Main Lemmas

In this section, we formulate the two main lemmas on which the proof of Theorem 2 rests.

We work in the *piecewise-linear* (*PL*) category (standard references are [38, 28]). All manifolds (possibly with boundary) are PL-manifolds (can be triangulated as locally finite simplicial complexes such that the link of every nonempty face is either a PL-sphere or a PL-ball), and all maps between **polyhedra** (geometric realizations of simplicial complexes) are PL-maps (*i.e.*, simplicial on sufficiently fine subdivisions).[9] In particular, all balls are PL-ball and all spheres are PL-spheres (PL-homeomorphic to a simplex and the boundary of a simplex, respectively).

A submanifold P of a manifold Q is **properly embedded** if $\partial P = P \cap \partial Q$. The **singular set** of a PL-map f defined on a polyhedron K is the closure in K of the set of points at which f is not injective.

One basic fact that we will use for the proofs of both Lemmas 5 and 7 is the following version of *engulfing* [38, Ch. VII]:

[8] The existence of an equivariant map implies, via standard equivariant obstruction theory, that there exists a map from the graph K into $\mathbb{R}^2$ such that the images of any two disjoint (*independent*) edges intersect an even number of times, which is the hypothesis of the Hanani–Tutte Theorem.

[9] The PL assumption is no loss of generality: if K is a finite simplicial complex and $f\colon K \to \mathbb{R}^d$ is an almost r-embedding then f can be slightly perturbed to a PL map with the same property.

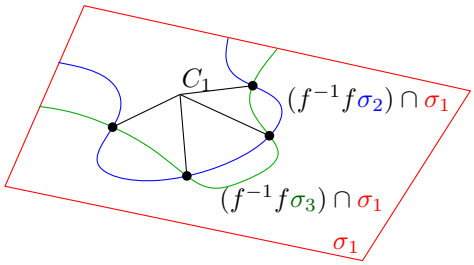


Figure 1 For $r = 3$, the construction of C_1 inside of σ_1. The collapsible polyhedron C_1 is a "cone" over the triple intersection set S_1 (which consists of four isolated points in the picture).

▶ **Theorem 4** (Engulfing, [38, Ch. VII, Thm. 20]). *Let M be an m-dimensional k-connected manifold with $k \le m-3$. Let X a compact x-dimensional subpolyhedron in the interior of M. If $x \le k$, then there exists a collapsible subpolyhedron C in the interior of M with $X \subseteq C$ and $\dim(C) \le x+1$.*

The collapsible polyhedron C can be thought of as an analogue of a "cone" over X.

▶ **Lemma 5** (Reduction Lemma). *Let m, d, r be three positive integers satisfying (2). Suppose $f\colon K \to \mathbb{R}^d$ is a map in general position, and $\sigma_1, \dots, \sigma_r$ be pairwise disjoint simplices of K of dimension $s_1, \dots, s_r \le m$ such that $f^{-1}(f(\sigma_1)\cap\cdots\cap f(\sigma_r))\cap\sigma_i$ is contained in the* interior *of σ_i. Then there exists a ball B^d in $\mathbb{R}^d$ such that*

1. *B^d intersects each $f(\sigma_i)$ in a ball that is properly embedded in B^d, and that ball avoids the image of the singular set of $f|_{\sigma_i}$, as well as $f(\partial\sigma_i)$;*
2. *B^d contains $f(\sigma_1)\cap\cdots\cap f(\sigma_r)$ in its interior; and*
3. *B^d does not intersect any other parts of the image $f(K)$.*

Proof. Let us consider $S_i := f^{-1}(f(\sigma_1)\cap\cdots\cap f(\sigma_r))\cap\sigma_i$. By general position [28, Thm 5.4] this is a polyhedron of dimension at most $s_1+\cdots+s_r-(r-1)d \le rm-(r-1)d$. By Theorem 4, we find $C_i \subseteq \sigma_i$ collapsible, containing S_i, and of dimension at most $rm-(r-1)d+1$. Figure 1 illustrates the case $r = 3$.

The dimension of the singular set of $f|_{\sigma_i}$ is at most $2s_i - d$. Hence, C_i is disjoint from it since $(rm-(r-1)d+1)+(2s_i-d)-s_i \le (r+1)m-rd+1$, which is negative in the metastable range. Thus, f is injective in a neighbourhood of C_i.

Again by Theorem 4, we find in $\mathbb{R}^d$ a collapsible polyhedron $C_{\mathbb{R}^d}$ of dimension at most $rm-(r-1)d+2$ and containing $f(C_1)\cup\cdots\cup f(C_r)$. Figure 2 illustrates the construction for $r = 3$. By general position we have the following properties:

1. $C_{\mathbb{R}^d}$ intersects $f(\sigma_i)$ exactly in $f(C_i)$. Indeed, in the metastable range, $rm-(r-1)d+2+s_i-d \le (r+1)m-rd+2 < 0$.
2. $C_{\mathbb{R}^d}$ does not intersect any other part of $f(K)$ (by a similar computation).

We take a small **regular neighbourhood** [28, Ch. 3] B of $C_{\mathbb{R}^d}$, which still avoids the singular set of each $f|_{\sigma_i}$ as well as other parts of $f(K)$. This regular neighbourhood is a ball, since $C_{\mathbb{R}^d}$ is collapsible. The intersection $B\cap f(\sigma_i)$ is a regular neighbourhood of $f(C_i)$ which is also a collapsible space, hence $B\cap f(\sigma_i)$ is a ball (properly contained in B). ◀

An **ambient isotopy** H of a PL-manifold X is a collection of homeomorphisms $H_t : X \to X$ for $t \in [0,1]$, which vary continuously with t, and with $H_0 = \mathrm{id}$. We say that an ambient isotopy H **throws** a subspace $Y \subseteq X$ onto Z if $H_1(Y) = Z$, see [38, Ch. V].

We say that an ambient isotopy H of X is **proper** if $H_t|_{\partial X} = \mathrm{id}_{\partial X}$ for all t.

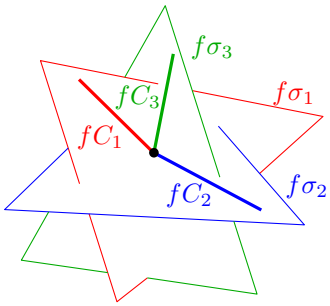


Figure 2 For $r = 3$, the polyhedron $C_{\mathbb{R}^d}$ is a "cone" over $fC_1 \cup fC_2 \cup fC_3$.

▶ **Definition 6.** Let m, d, r be three positive integers satisfying (2). Let $\sigma_1, \dots, \sigma_r$ be balls of dimensions $s_1, \dots, s_r \leq m$. We define

$$s := s_1 + \ldots + s_r.$$

Let f be a continuous map, mapping the disjoint union of the σ_i to a d-dimensional ball B^d, i.e.,

$$f : \sigma_1 \sqcup \cdots \sqcup \sigma_r \to B^d.$$

We define the **test map** $\widetilde{f}$ associated to f

$$\widetilde{f} : \sigma_1 \times \cdots \times \sigma_r \to B^d \times \cdots \times B^d, \quad \text{by} \quad (x_1, ..., x_r) \mapsto (fx_1, ..., fx_r).$$

If, for each $i = 1, ..., r$,

$$f\sigma_1 \cap \cdots \cap f\partial\sigma_i \cap \cdots \cap f\sigma_r = \emptyset,$$

then $\widetilde{f}\partial(\sigma_1 \times \cdots \times \sigma_r) \subset B^d \times \cdots \times B^d$, avoids the **thin diagonal** $\delta_r(B^d) = \{(x, \dots, x) \mid x \in B^d\}$ of B^d. Thus,

$$\partial(\sigma_1 \times \cdots \times \sigma_r) \to (B^d \times \cdots \times B^d) \setminus \delta_r(B^d). \tag{3}$$

Observe that $\partial(\sigma_1 \times \cdots \times \sigma_r) \cong S^{s-1}$, where $s := \sum_i s_i$, and $(B^d \times \cdots \times B^d) \setminus \delta_r(B^d)$ is homotopy equivalent to $S^{d(r-1)-1}$. Therefore, the map (3) defines an element

$$\alpha(f) \in \pi_{s-1}(S^{d(r-1)-1}),$$

which we call **intersection class of f**.

▶ **Lemma 7** (Generalized Weber-Whitney Trick)**.** *Let m, d, r be three positive integers satisfying (2). Let $\sigma_1, \dots, \sigma_r$ be balls of dimensions $s_1, \dots, s_r \leq m$ properly contained in a d-dimensional ball B and with $\sigma_1 \cap \cdots \cap \sigma_r$ in the* interior *of B.*

1. *Let us denote by α the intersection class of the map $\sigma_1 \sqcup \cdots \sqcup \sigma_r \to B^d$.*
   If $\alpha = 0$, then there exists $(r-1)$ proper ambient isotopies of B that we can apply to $\sigma_1, \dots, \sigma_{r-1}$, respectively, to remove the r-intersection set; i.e., there exist $(r-1)$ proper isotopies $H_t^1, \dots, H_t^{r-1}$ of B throwing σ_i onto $\sigma_i' := H_1^i \sigma_i$ and such that

   $$\sigma_1' \cap \cdots \cap \sigma_{r-1}' \cap \sigma_r = \emptyset.$$

2. *Let us assume that $\sigma_1 \cap \cdots \cap \sigma_r = \emptyset$ and $\sigma_2 \cap \cdots \cap \sigma_r \neq \emptyset$, and let $z \in \pi_s(S^{d(r-1)-1})$. There exists J_t a proper ambient isotopy of B such that*

- $J_1\sigma_1 \cap \sigma_2 \cap \cdots \cap \sigma_{r-1} \cap \sigma_r = \emptyset$,
- *The intersection class of f is z, where*

$$f : (\sigma_1 \times I) \sqcup \sigma_2 \sqcup \cdots \sqcup \sigma_r \to B^d$$

is defined as the inclusion on σ_i for $i \geq 2$, and for $(x,t) \in \sigma_1 \times I$, $f(x,t) = J_t(x)$.

▶ Remark.

- For $r = 2$, Lemma 7 already appears in Section 4 of Weber's thesis [37].
- Roughly speaking, Part 1 of Lemma 7 means that if the intersection class vanishes, then one can resolve the r-intersection set.
  Part 2 means that each element of $\pi_s(S^{d(r-1)-1})$ can be obtained by moving from a fixed solution to a new solution.

3 Proof of Theorem 2

Here, we show how to use Lemmas 5 and 7 to prove the main theorem. The inductive argument used in the proof mirrors that of Section 5 in Weber's thesis [37], where Theorem 2 is proven for $r = 2$.

Proof of Theorem 2. We are given $F : (K)^r_\Delta \to_{\mathfrak{S}_r} S^{d(r-1)-1}$, and we want to construct $f : K \to \mathbb{R}^d$ without global r-fold point.

We start with a map $f : K \to \mathbb{R}^d$ in general position. Inductively, we will redefine f on the skeleta of K as to get the desired property. There are two levels in the induction. To describe these, let us fix a total ordering of the simplices of K that extends the partial ordering by dimension, *i.e.*,

$$K = \{\tau_1, \ldots, \tau_N\}, \qquad \dim \tau_i \leq \dim \tau_{i+1} \text{ for } 1 \leq i \leq N-1.$$

First, we give a very informal plan of the "double induction" that we are going to use in the proof: we go over the list of simplices $\tau_1, ..., \tau_N$, and for each simplex τ_i we consider all the global r-fold points of τ_i with all the simplices *before* τ_i in the list. More precisely, we consider the list l_i of all r-tuples of pairwise disjoint simplices containing τ_i and simplices *before* τ_i in the list $\tau_1, ..., \tau_N$. For each r-tuple in l_i, we need to eliminate its global r-fold points.

Therefore, once τ_i is fixed, we have a *new list* l_i. We are going to *order* l_i (by a notion of dimension), and then inductively scan over it and remove the global r-fold points for each r-tuple in l_i.

Let us describe now the first level of the inductive argument. We have to prove the following: Suppose we are given a map $f : K \to \mathbb{R}^d$ in general position with the following two properties:

1. Restricted to the subcomplex $L = \{\tau_1, \ldots, \tau_{N-1}\}$ the map $f|_L$ does not have any r-fold points between disjoint r-tuples of simplices;
2. $\widetilde{f}$ restricted to $(L)^r_\Delta$ is $\mathfrak{S}_r$-equivariantly homotopic to F, where $\widetilde{f}$ is the map defined in Lemma 1.

Then we can redefine f as to have these two properties on the whole of K. This is the first level of induction.

For the second level of the induction, let us define the **dimension** of a finite set of simplices as the sum of their individual dimensions. For the the purposes of this proof, we use the terminology **k-collection** for a set of cardinality k. Consider those $(r-1)$-collections

t of simplices of L that, together with τ_N, form an r-collection of pairwise disjoint simplices. We fix a total ordering of these $(r-1)$-collections that extends the partial ordering given by dimension, *i.e.*, we list them as

$$t_1, \dots, t_M,$$

with $\dim t_i \leq \dim t_{i+1}$ for $1 \leq i < M$. (Thus, each t_i is an $(r-1)$-collection of simplices of L, and t_i joined with τ_N is a r-collection of pairwise disjoint simplices.) Once again, inductively, it suffices to prove the following: Assuming that f has the two properties

1. For each $(r-1)$-collection t_i in the list $t_1, \dots, t_{M-1}$, the map f does not have any r-fold points with preimages in the r-collection formed by adjoining τ_N to t_i.
2. the map $\widetilde{f}$ is $\mathfrak{S}_r$-equivariantly homotopic to F on the complex

$$(L)^r_\Delta \cup \bigcup_{i \leq M-1} [t_i \cup \{\tau_N\}] \subseteq (K)^r_\Delta,$$

where the operator $[-]$ converts an unordered r-collection of pairwise disjoint simplices of K into the set of its corresponding cells[10] in $(K)^r_\Delta$.

Then we can modify f as to have these two properties on the list $t_1, \dots, t_M$.

In order to do so, let us consider the r-collection $t_M \cup \{\tau_N\}$. We rename its elements as

$$t_M \cup \{\tau_N\} = \{\sigma_1, \dots, \sigma_r\}, \quad (\text{with } \tau_N = \sigma_r).$$

By the induction hypothesis (namely the order on the τ_i and the t_i), for each $i = 1, \dots, r$, $f^{-1}(f\sigma_1 \cap \dots \cap f\sigma_r) \cap \sigma_i$ is contained in the *interior* of σ_i (since the induction has already "worked" on the simplices in $\partial\sigma_i$). Furthermore, the map $\widetilde{f} : \partial(\sigma_1 \times \dots \times \sigma_r) \to S^{d(r-1)-1}$ is homotopic to F, this also follows from the ordering on the τ_i and the t_i (the homotopy is already defined on all the cells of $\partial(\sigma_1 \times \dots \times \sigma_r)$).

We are in position to apply Lemma 5: we find a ball B^d in $\mathbb{R}^d$ with the three properties listed in the Lemma. Let us call σ'_i the sub-ball in σ_i properly embedded into B^d, *i.e.*, $\sigma'_i \overset{f}{\hookrightarrow} B^d$, and $f\partial\sigma'_i = \partial B^d \cap f\sigma'_i$.

By the Combinatorial Annulus Theorem [8, 3.10], there exists an isotopy of σ_i in itself that progressively retracts σ_i to σ'_i. *I.e.*, there exists $G^i_t : \sigma_i \to \sigma_i$ with G^i_0 being the identity and G^i_1 being a homeomorphism between σ_i and σ'_i. We define a homotopy by

$$\begin{array}{rccc} G: & \partial(I \times \sigma_1 \times \dots \times \sigma_r) & \xrightarrow{fG^1 \times \dots \times fG^r} & \mathbb{R}^d \times \dots \times \mathbb{R}^d \setminus \delta_r \mathbb{R}^d \\ & (t, x_1, \dots, x_r) & \longmapsto & (fG^1_t x_1, \dots fG^r_t x_r). \end{array} \tag{4}$$

By the induction hypothesis,

$$\partial(\sigma_1 \times \dots \times \sigma_r) \xrightarrow{f \times \dots \times f} \mathbb{R}^d \times \dots \times \mathbb{R}^d \setminus \delta_r \mathbb{R}^d \tag{5}$$

is homotopic to F, and F is defined over $\sigma_1 \times \dots \times \sigma_r$. Therefore, the homotopy class of

$$\partial(\sigma'_1 \times \dots \times \sigma'_r) \xrightarrow{f \times \dots \times f} B^d \times \dots \times B^d \setminus \delta_r B^d$$

is trivial. Hence, we can use the first part of the Lemma 7 to find $(r-1)$ proper ambient isotopies of B, say $H^1_t, \dots, H^{r-1}_t$, such that $H^1_1(f\sigma'_1) \cap \dots \cap H^{r-1}_1(f\sigma'_{r-1}) \cap f\sigma'_r = \emptyset$. This removes the r-fold points.

[10] E.g., $[\{\alpha, \beta, \gamma\}] = \{\alpha \times \beta \times \gamma, \alpha \times \gamma \times \beta, \beta \times \alpha \times \gamma, \beta \times \gamma \times \alpha, \gamma \times \alpha \times \beta, \gamma \times \beta \times \alpha\}$.

To finish the induction, we also need to extend the equivariant homotopy between $\widetilde{f}$ and F on the cell $\sigma_1 \times \cdots \times \sigma_r$, as the homotopy is already defined on $\partial(\sigma_1 \times \cdots \times \sigma_r)$. This is when the second part of Lemma 7 becomes useful.

We define a map on $\partial(I \times \sigma_1 \times \cdots \times \sigma_r) \to \mathbb{R}^d \times \cdots \times \mathbb{R}^d \setminus \delta_r \mathbb{R}^d$ in the following way:

1. on $\{0\} \times \sigma_1 \times \cdots \times \sigma_r$, we use F,
2. on $[0, \frac{1}{3}] \times \partial(\sigma_1 \times \cdots \times \sigma_r)$, we use the homotopy from F to (5),
3. on $[\frac{1}{3}, \frac{2}{3}] \times \partial(\sigma_1 \times \cdots \times \sigma_r)$, we use G,
4. on $[\frac{2}{3}, 1] \times \partial(\sigma_1 \times \cdots \times \sigma_r)$, we use $(H_t^1 \times \cdots \times H_t^{r-1} \times \mathrm{id}) \circ (fG_1^1 \times \cdots \times fG_1^r)$,
5. $\{1\} \times \sigma_1 \times \cdots \times \sigma_r$, we use $(H_1^1 \times \cdots \times H_1^{r-1} \times \mathrm{id}) \circ (fG_1^1 \times \cdots \times fG_1^r)$.

This defines a class $\theta \in \pi_{\sum \dim \sigma_i}(S^{d(r-1)-1})$. To conclude, we need to have $\theta = 0$ (this is the condition to be able to extend to homotopy between $\widetilde{f}$ and F).

By the second part of Lemma 7, we can[11] perform a "second move" on σ_1 with an ambient isotopy J_t of B such that

$$\partial(I \times \sigma_1 \times \cdots \times \sigma_r) \xrightarrow{(J_t \times \mathrm{id} \times \cdots \times \mathrm{id}) \circ (H_1^1 \times \cdots \times H_1^{r-1} \times \mathrm{id}) \circ (fG_1^1 \times \cdots \times fG_1^r)} \mathbb{R}^d \times \cdots \times \mathbb{R}^d \setminus \delta_r \mathbb{R}^d$$

represents exactly $-\theta$. Therefore, by using this last move, we can assume that $\theta = 0$, *i.e.*, we can extend the equivariant homotopy between $\widetilde{f}$ and F, as needed for the induction. ◀

References

1 S. Avvakumov, I. Mabillard, A. Skopenkov, and U. Wagner. Eliminating higher-multiplicity intersections, III. Codimension 2. Preprint, `http://arxiv.org/abs/1511.03501`, 2015.

2 E. G. Bajmóczy and I. Bárány. On a common generalization of Borsuk's and Radon's theorem. *Acta Math. Acad. Sci. Hungar.*, 34(3-4):347–350 (1980), 1979. `doi:10.1007/BF01896131`.

3 I. Bárány and D. G. Larman. A colored version of Tverberg's theorem. *J. London Math. Soc. (2)*, 45(2):314–320, 1992. `doi:10.1112/jlms/s2-45.2.314`.

4 I. Bárány, S. B. Shlosman, and A. Szűcs. On a topological generalization of a theorem of Tverberg. *J. London Math. Soc., II. Ser.*, 23:158–164, 1981.

5 Imre Bárány, Zoltán Füredi, and Lászlo Lovász. On the number of halving planes. *Combinatorica*, 10(2):175–183, 1990.

6 P. V. M. Blagojević, B. Matschke, and G. M. Ziegler. Optimal bounds for the colored Tverberg problem. *J. Eur. Math. Soc.*, 17(4):739–754, 2015.

7 Pavle V. M. Blagojević, Florian Frick, and Günter M. Ziegler. Tverberg plus constraints. *Bull. Lond. Math. Soc.*, 46(5):953–967, 2014. `doi:10.1112/blms/bdu049`.

8 John L. Bryant. Piecewise linear topology. In *Handbook of geometric topology*, pages 219–259. North-Holland, Amsterdam, 2002.

9 Martin Čadek, Marek Krčál, and Lukáš Vokřínek. Algorithmic solvability of the lifting-extension problem. Preprint, `arXiv:1307.6444`, 2013.

[11] We can always obtain the assumption $\sigma_2 \cap \cdots \cap \sigma_r \neq \emptyset$ by modifying the map f as follows [18, "Finger moves" in the proof of Lemma 43]: we pick $r-1$ spheres $S^{s_2}, ..., S^{s_r}$ in the interior of B^d of dimension $s_2, ..., s_r$ in general position and such that $S^{s_2} \cap \cdots \cap S^{s_r}$ is a sphere S. Then, for $i = 2, ..., r$, we pipe σ_i' to S^{s_i}. The resulting map has the desired property.
This "piping" change can be absorbed by a slight modification (and renumbering) of the H_t^i. The support of these modifications is a collection of regular neighborhoods of 1-polyhedra (= paths used for piping).
Also, note that the cases when, by general position, $\dim S < 0$ corresponds the trivial cases $\theta = 0$. Indeed, $\dim S < 0$ corresponds to $(d - s_2) + \cdots + (d - s_r) > d$, i.e., $(r-1)d + s_1 - d > \sum s_i$, and since $s_1 - d \leq -3$, we have $(r-1)d - 1 > \sum s_i$, and so $\pi_{\sum \dim \sigma_i}(S^{d(r-1)-1}) = 0$.

10 Haim Chojnacki (Hanani). Über wesentlich unplättbare Kurven im dreidimensionalen Raume. *Fund. Math.*, 23:135–142, 1934.

11 Michael H. Freedman, Vyacheslav S. Krushkal, and Peter Teichner. van Kampen's embedding obstruction is incomplete for 2-complexes in $\mathbf{R}^4$. *Math. Res. Lett.*, 1(2):167–176, 1994. doi:10.4310/MRL.1994.v1.n2.a4.

12 Florian Frick. Counterexamples to the topological Tverberg conjecture. Preprint, arXiv:1502.00947, 2015.

13 Daciberg Gonçalves and Arkadiy Skopenkov. Embeddings of homology equivalent manifolds with boundary. *Topology Appl.*, 153(12):2026–2034, 2006.

14 Mikhail Gromov. Singularities, expanders and topology of maps. part 2: From combinatorics to topology via algebraic isoperimetry. *Geometric and Functional Analysis*, 20(2):416–526, 2010.

15 P. M. Gruber and R. Schneider. Problems in geometric convexity. In *Contributions to geometry (Proc. Geom. Sympos., Siegen, 1978)*, pages 255–278. Birkhäuser, Basel-Boston, Mass., 1979.

16 André Haefliger. Plongements différentiables dans le domaine stable. *Comment. Math. Helv.*, 37:155–176, 1962/1963.

17 Isaac Mabillard and Uli Wagner. Eliminating Tverberg points, I. An analogue of the Whitney trick. In *Proc. 30th Ann. Symp. on Computational Geometry*, pages 171–180, 2014.

18 Isaac Mabillard and Uli Wagner. Eliminating higher-multiplicity intersections, I. a Whitney trick for Tverberg-type problems. Preprint, arXiv:1508.02349, 2015.

19 Isaac Mabillard and Uli Wagner. Eliminating Higher-Multiplicity Intersections, II. The Deleted Product Criterion in the r-Metastable Range. *Preprint* arXiv:1601.00876, 2016.

20 S. Mardešić and J. Segal. ε-Mappings and generalized manifolds. *Mich. Math. J.*, 14:171–182, 1967.

21 Jiří Matoušek. *Using the Borsuk-Ulam theorem*. Springer-Verlag, Berlin, 2003.

22 Jiří Matoušek, Eric Sedgwick, Martin Tancer, and Uli Wagner. Embeddability in the 3-sphere is decidable. Preprint, arXiv:1402.0815, 2014.

23 Jiří Matoušek, Martin Tancer, and Uli Wagner. Hardness of embedding simplicial complexes in $\mathbb{R}^d$. *J. Eur. Math. Soc.*, 13(2):259–295, 2011.

24 Murat Özaydin. Equivariant maps for the symmetric group. Unpublished manuscript, available online at http://minds.wisconsin.edu/handle/1793/63829, 1987.

25 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing even crossings. *J. Combin. Theory Ser. B*, 97(4):489–500, 2007. doi:10.1016/j.jctb.2006.08.001.

26 Michael J. Pelsmajer, Marcus Schaefer, and Daniel Štefankovič. Removing independently even crossings. *SIAM J. Discrete Math.*, 24(2):379–393, 2010. doi:10.1137/090765729.

27 Dušan Repovš and Arkadiy B. Skopenkov. New results on embeddings of polyhedra and manifolds into Euclidean spaces. *Uspekhi Mat. Nauk*, 54(6(330)):61–108, 1999.

28 Colin Patrick Rourke and Brian Joseph Sanderson. *Introduction to piecewise-linear topology*. Springer Study Edition. Springer-Verlag, Berlin, 1982. Reprint.

29 K. S. Sarkaria. A generalized van Kampen-Flores theorem. *Proc. Amer. Math. Soc.*, 111(2):559–565, 1991. doi:10.2307/2048349.

30 J. Segal, A. Skopenkov, and S. Spież. Embeddings of polyhedra in $\mathbb{R}^m$ and the deleted product obstruction. *Topology Appl.*, 85(1-3):335–344, 1998.

31 J. Segal and S. Spież. Quasi embeddings and embeddings of polyhedra in $\mathbb{R}^m$. *Topology Appl.*, 45(3):275–282, 1992.

32 A. Skopenkov. On the deleted product criterion for embeddability in $\mathbf{R}^m$. *Proc. Amer. Math. Soc.*, 126(8):2467–2476, 1998. doi:10.1090/S0002-9939-98-04142-2.

33 Arkadiy B. Skopenkov. Embedding and knotting of manifolds in Euclidean spaces. In *Surveys in contemporary mathematics*, volume 347 of *London Math. Soc. Lecture Note Ser.*, pages 248–342. Cambridge Univ. Press, Cambridge, 2008.

34 William T. Tutte. Toward a theory of crossing numbers. *J. Combin. Theory*, 8:45–53, 1970.

35 H. Tverberg. A generalization of Radon's theorem. *J. London Math. Soc.*, 41:123–128, 1966.

36 A. Yu. Volovikov. On the van Kampen-Flores theorem. *Mat. Zametki*, 59(5):663–670, 797, 1996. doi:10.1007/BF02308813.

37 Claude Weber. Plongements de polyhèdres dans le domaine métastable. *Comment. Math. Helv.*, 42:1–27, 1967.

38 Erik Christopher Zeeman. *Seminar on combinatorial topology.* Institut des Hautes Études Scientifiques, 1966.

39 Rade T. Živaljević. User's guide to equivariant methods in combinatorics. *Publ. Inst. Math. Beograd*, 59(73):114–130, 1996.

40 Rade T. Živaljević. User's guide to equivariant methods in combinatorics. II. *Publ. Inst. Math. (Beograd) (N.S.)*, 64(78):107–132, 1998.

41 Rade T. Živaljević and Sinisa T. Vrećica. The colored Tverberg's problem and complexes of injective functions. *J. Combin. Theory Ser. A*, 61(2):309–318, 1992.

Peeling and Nibbling the Cactus: Subexponential-Time Algorithms for Counting Triangulations and Related Problems*

Dániel Marx[†1] **and Tillmann Miltzow**[2]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
dmarx@cs.bme.hu

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
t.miltzow@gmail.com

Abstract

Given a set of n points S in the plane, a triangulation T of S is a maximal set of non-crossing segments with endpoints in S. We present an algorithm that computes the number of triangulations on a given set of n points in time $n^{(11+o(1))\sqrt{n}}$, significantly improving the previous best running time of $O(2^n n^2)$ by Alvarez and Seidel [SoCG 2013]. Our main tool is identifying separators of size $O(\sqrt{n})$ of a triangulation in a canonical way. The definition of the separators are based on the decomposition of the triangulation into nested layers ("cactus graphs"). Based on the above algorithm, we develop a simple and formal framework to count other non-crossing straight-line graphs in $n^{O(\sqrt{n})}$ time. We demonstrate the usefulness of the framework by applying it to counting non-crossing Hamilton cycles, spanning trees, perfect matchings, 3-colorable triangulations, connected graphs, cycle decompositions, quadrangulations, 3-regular graphs, and more.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases computational geometry, triangulations, exponential-time algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.52

1 Introduction

Given a set S of n points in the plane, a triangulation T of S is defined to be a maximal set of non-crossing line segments with both endpoints in S. This set of segments together with the set S defines a plane graph. It is easy to see that every bounded face of a triangulation T is indeed a triangle. We assume that S is in general position: no three points of S are on a line. Triangulations are one of the most studied concepts in discrete and computational geometry, studied both from combinatorial and algorithmic perspectives [8, 9, 11, 22, 23, 27, 28, 43]. It is well known that the number of possible triangulations of n points in convex position is exactly the $(n-2)$-th Catalan number, but counting the number of triangulations of arbitrary point sets seems to be a much harder problem. There is a long line of research devoted to finding better and better exponential-time algorithms for counting triangulations [1, 2, 3, 4, 5, 6, 7, 21, 26, 30, 33, 42]. The sequence of improvements culminated in the

* Supported by the ERC grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results", no. 280152.
† The first author is supported by OTKA grant NK105645.


© Dániel Marx and Tillmann Miltzow;
licensed under Creative Commons License CC-BY

$O(2^n n^2)$ time algorithm of Alvarez and Seidel [5], winning the best paper award at SoCG 2013. Our main result significantly improves the running time of counting triangulations by making it subexponential:

▶ **Theorem 1** (General Plane Algorithm). *There exists an algorithm that, given a set S of n points in the plane, computes the number of all triangulations of S in $n^{(11+o(1))\sqrt{n}}$ time.*

It is very often the case that restricting an algorithmic problem to planar graphs allows us to solve it with much better worst-case running time than what is possible for the unrestricted problem. One can observe a certain "square root phenomenon": in many cases, the best known running time for a planar problem contains a square root in the exponent. For example, the 3-Coloring problem on an n-vertex graph can be solved in subexponential time $2^{O(\sqrt{n})}$ on planar graphs (e.g., by observing that a planar graph on n vertices has treewidth $O(\sqrt{n})$), but only $2^{O(n)}$ time algorithms are known for general graphs. Moreover, it is known that if we assume the Exponential-Time Hypothesis (ETH), which states that there is no $2^{o(n)}$ time algorithm for n-variable 3SAT, then there is no $2^{o(\sqrt{n})}$ time algorithm for 3-Coloring on planar graphs and no $2^{o(n)}$ time algorithm on general graphs [37]. The situation is similar for the planar restrictions of many other NP-hard problems, thus it seems that the appearance of the square root of the running time is an essential feature of planar problems. A similar phenomenon occurs in the framework of parameterized problems, where running times of the form $2^{O(\sqrt{k})} \cdot n^{O(1)}$ or $n^{O(\sqrt{k})}$ appear for many planar problems and are known to be essentially best possible (assuming ETH) [12, 14, 13, 25, 19, 20, 15, 17, 16, 44, 24, 18, 31, 32, 10, 41, 40, 39].

A triangulation of n points can be considered as a planar graph on n vertices, hence it is a natural question whether the square root phenomenon holds for the problem of counting triangulations. Indeed, for the related problem of finding a minimum weight triangulation, subexponential algorithms with running time $n^{O(\sqrt{n})}$ are known [34, 35]. These algorithms are based on the use of small balanced separators. Given a plane triangulation on n points in the plane, it is well known that there exists a balanced $O(\sqrt{n})$-sized separator that divides the triangulation into at least two independent graphs [36]. The basic idea is to guess a correct $O(\sqrt{n})$-sized separator of a minimum weight triangulation and recurse on every occurring subproblem. As there are only $n^{O(\sqrt{n})}$ potential graphs on $O(\sqrt{n})$ vertices, one can show that the whole algorithm takes $n^{O(\sqrt{n})}$ time [34, 35].

Unfortunately, this approach has serious problems when we try to apply it to counting triangulations. The fundamental issue with this approach is that a triangulation of course may have more than one $O(\sqrt{n})$-sized balanced separators and hence we may overcount the number of triangulations, as a triangulation would be taken into account in more than one of the guesses. To get around this problem, an obvious simple idea would be to say that we always try to guess a "canonical" separator, for example, the lexicographically first separator. However, it is a complete mystery how to guarantee in subsequent recursion steps that the separator we have chosen is indeed the lexicographic first for all the triangulations we want to count. Perhaps the most important technical idea of the paper is finding a suitable way of making the separators canonical.

This extended abstract is committed to give a comprehensive self-contained explanation of the main concepts of the algorithm. For a version with all details and proofs see [38].

1.1 Preliminaries

We interpret a collection of points V and non-crossing segments E in $\mathbb{R}^2$ as a plane graph, if every segment $e \in E$ shares exactly its endpoints with V. We usually identify points with

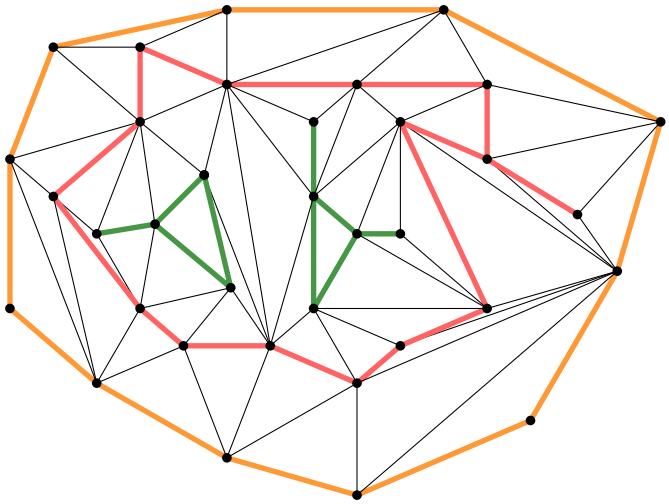


Figure 1 A triangulation with three cactus layers. Layer 1 is colored orange, Layer two is colored red and Layer 3 is colored green. Note that the layers do not need to be connected.

vertices and edges with segments. We always denote by n the number of vertices. A plane graph induces naturally a partition of the plane into open faces, open segments and a finite collection of points. The unique unbounded face is called the *outer face*. We call a plane graph G a *cactus graph* (or just *cactus*) if all its vertices and edges are incident to the *outer face*. Such a graph is outerplane, but some outerplane graphs are not cacti. For convenience, we do *not* require cactus graphs to be connected. A *triangulation* of a set of points is a maximal plane graph on those points.

We define a decomposition of a triangulation into nested *(cactus) layers* of cacti. The first (cactus) layer is defined by the set of vertices and edges incident to the outer face. Inductively, the i-th layer is defined by the vertices and edges incident to the outer faces after the first $i-1$ layers are removed and has *index* i. We say layer i is further *outside* then layer j if $i < j$ and in this case layer j is more *inside* than layer i. The *outerplanar index* of a graph is defined by the number of non-empty (cactus) layers. Further, we can give each vertex uniquely the *index* of the layer it is contained in. It is not difficult to see that the index equals the distance to the boundary of the convex hull.

We define $\partial CH(S)$ as the boundary of the convex hull of the point set S. The *onion layers* of a set of points S are defined inductively in a similar fashion. The first layer is $\partial CH(S)$. The i-th layer is the boundary of the convex hull after the first $i-1$ layers are removed.

The definition of cactus layers and onion layers should not be confused: the onion layers are completely defined by the point set only, whereas cactus layers are defined by the point set and the triangulation. In particular, it is easy to construct a point set with an arbitrary large number of onion layers, but having a triangulation with only two cactus layers.

1.2 Results

Given a triangulation T, we define small *canonical* separators by distinguishing two cases. If T has more than $\sqrt{n}$ cactus layers, then one of the first $\sqrt{n}$ layers has size at most $\sqrt{n}$ and we can define the one with smallest index to be the canonical separator. Using such a separator, we *peel off* some cacti to reduce the problem size. In the case when we have

only a few cactus layers, we can define short canonical separator paths from any vertex to the outer face of the triangulation. We formalize both ideas into a dynamic programming algorithm. The main difficulty is to define the subproblems appropriately. We define ring subproblems to be suitable for layer separators and nibbled ring subproblems to be suitable for the separator paths.

As a byproduct of this algorithmic scheme, we can efficiently count triangulations with a small number of layers. This is similar to previous work on finding a minimum weight triangulation [6] and counting triangulations [2] for point sets with a small number of onion layers.

▶ **Theorem 2** (Thin Plane Algorithm). *There exists an algorithm that given a set S of n points in the plane computes the number of all triangulations of S with outerplanar index k in $n^{O(k)}$ time.*

One may want to count triangulations subject to certain constraints (e.g., degree bounds, or bounds on the angles of the triangles, etc.) or generalize the problem to counting colored triangulations with colors on the vertices or edges. We introduce an annotated version of the problem to express such generalizations in a clean and formal way. An *annotated* triangle is a 9-tuple consisting of 3 points of S, which form an empty triangle and 6 strings, one for each vertex and edge of the triangle. An *annotation system* is a list L of annotated triangles. An *annotated triangulation* T is a triangulation with a string defined for each vertex and each edge (so these strings define an annotated triangle for each triangle of T). Given an annotation system L, we call an annotated triangulation T *valid* if every annotated triangle Δ of T belongs to L. With little extra effort, we can generalize our algorithms to count also valid annotated triangulations. We denote by $|L|$ the number of annotated triangles and assume that each string can be described with $n^{O(1)}$ bits.

▶ **Theorem 3** (Counting Annotated Triangulations). *Given an annotation system L and a set S of n points in the plane, we can count all valid annotated triangulations in time $n^{(11+o(1))\sqrt{n}} \cdot |L|^{(12+o(1))\sqrt{n}}$.*

As examples of this generalization, we can count triangulations that are 3-colorable or where each point has a specified degree in the triangulation: all we need is to carefully design a suitable annotation system.

▶ **Theorem 4.** *Given a set S of n points in the plane, we can count all 3-colorable triangulations of S in time $n^{O(\sqrt{n})}$.*

▶ **Theorem 5.** *Given a set S of n points in the plane with prescribed degrees on each vertex, we can count all triangulations T satisfying the degree constraints in time $n^{O(\sqrt{n})}$.*

More generally, instead of triangulations, we could be interested in counting other geometric graph classes, such as non-crossing perfect matchings, non-crossing Hamilton cycles, etc. Surprisingly, many such problems can be expressed in a completely formal way in our framework of counting annotated triangulations. The idea here is to extend the geometric graph into a 2-edge-colored triangulation, with one color forming the original geometric graph itself and the other color representing the edges of the triangulation that were not present in the original graph. To make this idea work, we have to ensure that for each member of our graph class, we count only one 2-edge-colored triangulation. This is nontrivial, as a given geometric graph can be extended into a 2-edge-colored triangulation in many different ways. Similarly to previous work [2, 4], we use the notion of constrained Delaunay triangulation (see [29]) to enforce that each graph has a unique extension into a valid 2-edge-colored

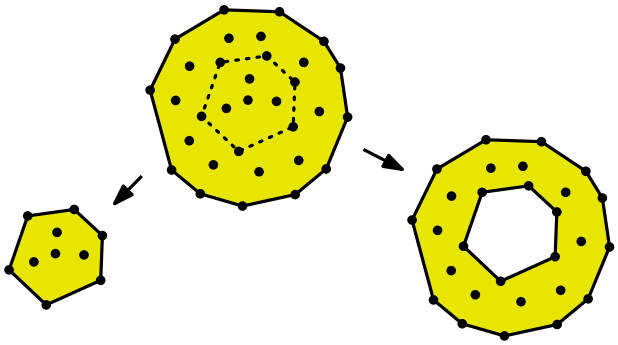


Figure 2 Cactus layes are ideal separators, as they separate in a simple way the inside from the outside and we can easily *peel off* large layers from the outside.

triangulation. By formalizing this idea and carefully designing annotation systems, it is possible to get $n^{O(\sqrt{n})}$ time algorithms for a large number of graph classes. The following theorem states some important examples to demonstrate the applicability of our approach.

▶ **Theorem 6** (Counting Geometric Structures). *The following non-crossing structures can be counted in time $n^{O(\sqrt{n})}$ on a set of n points in the plane: the set of all graphs, perfect matchings, cycle decompositions, Hamilton cycles, Hamiltonian paths, Euler tours, spanning trees, d-regular graphs, and quadrangulations.*

We would like to emphasize that the proof of Theorem 6 uses the algorithm of Theorem 3 as a black box. Thus these results can be proved in a completely formal way without the need for revisiting the details of the proof of Theorem 3. In addition to the actual algorithms presented in the paper, we consider our second main contribution to be the development of the framework of annotated triangulations and demonstrating its flexibility in modeling other problems.

1.3 The Essence of the Key Ideas of the Algorithm

The complete proof can be found in the full online version. We want to use this extended abstract to present the key ideas of the algorithm in a level of detail that gives the reader a good understanding of the algorithm without indulging into the nuances of the technical details. Before we do this, we want to give the reader the essence in an even more condensed high level form.

Similar to almost all previous algorithms, our algorithm is based on separators. There are two major differences: the separators are defined not in terms of the input (point set), but in terms of the output (triangulation). We use *two* kind of separators. Depending on the type of triangulation we want to count, we choose which kind of separator we use.

Given a triangulation with outerplanar index $k > \sqrt{n}$, there exists a layer of size at most $\sqrt{n}$ by the pigeonhole principle, see Figure 2. This layer is an ideal candidate for a separator. As layers are nested and separate the inner from the outer part completely. In order to make them canonical, we choose the *outermost* small cactus layer and *peel it off*. In subsequent recursions, we have to ensure that we count only triangulations where this was indeed the outermost small cactus layer. This can be done by guessing all possible sizes of all layers that have smaller index.

In case that the outerplanar index k is below $\sqrt{n}$, there exists a path of length at most $k-1$ from any vertex to the outermost layer, see Figure 3. (The idea is that every vertex

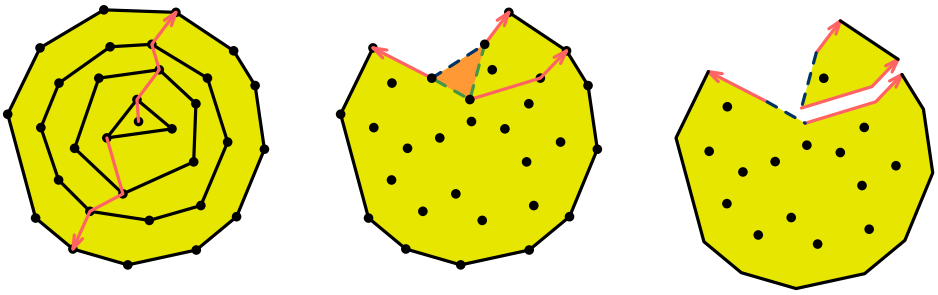


Figure 3 Paths from the interior to the outside are short separators, if the triangulation does not have too many layers. The resulting dynamic programming scheme is a little unintuitive, as it *nibbles off* the algorithmic problem from the outside.

either is adjacent to the outer face or has a neighbor with smaller index.) If done correctly, these paths can be used as a separator within a well designed dynamic programming scheme. Anagnostou and Corneil [6] demonstrated this for finding a minimum weight triangulation, using layers defined in terms of the input instead of layers defined in terms of the output. The algorithmic idea is essentially the same. Alvarez, Bringmann, Curticapean and Ray showed how to make these separators canonical and thus suitable for counting problems [2], by giving each vertex a fixed distinguished rank and always choose the next vertex on the separator path with the smallest available rank.

We use these separator paths in an, at first, unintuitive way. Consider the case, where we have already made a few recursion steps, see Figure 3. (The very first step is degenerate and not suitable for an illustrative example.) In this situation, we have to triangulate a simple polygon, where one edge $e = (u, v)$ is singled out. Some part of the polygon comes from separator paths of previous recursion steps. We guess all triangles $\Delta = \Delta(u, v, w)$ incident to $e = (u, v)$ and all short paths p from w to $\partial CH(S)$. In this way, we attain a large and a small subproblem. The triangle Δ defines special edges for subsequent subproblems. We repeat the procedure for all appearing subproblems till the subproblems are of constant size. We *nibble off* small bites in each recursion step. Only at later stages larger bites are taken.

The technical and conceptual difficulties come from the need to combine the two different separators into *one* dynamic programming scheme. Note that we might first guess a layer then the sizes of layers with smaller index and thereafter use the path separators as described above. Thus, we have to define very carefully subproblems for our dynamic programing routine that are specifically designed to work for both separators.

The runtime bound follows from the size of the separators. Whenever we guess a separator of size $O(\sqrt{n})$, we have at most $\binom{n}{O(\sqrt{n})} = n^{O(\sqrt{n})}$ possibilities.

2 Ring Subproblems

Our algorithm is based on dynamic programming: we define a large number of subproblems that are *more general* than the problem we are trying to solve. We generalize the problem by considering *rings:* we need to triangulate a point set in a region between a polygon and a cactus. Additionally, we may have layer-constraints prescribing that a certain number of vertices should appear on certain layers.

We proceed to give the crucial ingredients of ring subproblems, which serve as the basis of the dynamic programming, see Figure 4 for an illustration. We omit any technical conditions, which are important to show correctness, but are not essential for the main ideas.

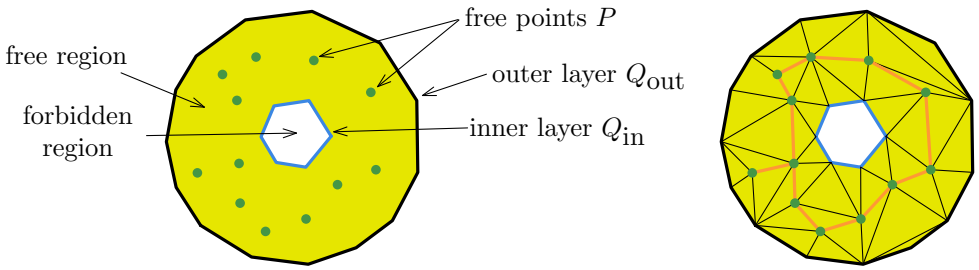


Figure 4 At the left a layer-unconstrained ring subproblem is depicted. At the right a valid triangulation of the ring subproblem is drawn. It consists of three layers: The outer layer drawn in black, the "middle" layer drawn in orange and the inner layer drawn in blue.

▶ **Definition 7** (Ring Subproblems). A *ring subproblem* $\mathcal{S}$ consists of an *outer layer* Q_{out} an *inner layer* Q_{in}, some width information, and an optional *layer-constraint vector* c.

outer layer: The *outer layer* Q_{out} is a simple poylgon.
inner layer: The *inner layer* Q_{in} is a cactus contained in the outer layer. An important special case is that the inner layer might be empty.
layer-constraint vector: This vector c prescribes the size of each layer of the triangulation we are looking for. We do not always specify a layer constraint vector: We distinuish between layer-constrained ring subproblems and layer-*un*constrained ring subproblems.
width: The *width* is a natural number w that specifies how many non-empty cactus layers any valid triangulation may have.
free region: The *free region* is the region "between" the inner and outer layer.
free points: The *free points* P are the points of the original point set inside the free region.

Given a layer-unconstrained ring subproblem $\mathcal{S}$ and a layer-constraint vector c, we denote by $\mathcal{S}(c)$ the layer-constrained ring subproblem defined by them.

▶ **Definition 8** (Valid Triangulation). Given a ring subproblem $\mathcal{S}$, consider a graph T extending the graph formed by $Q_{\text{in}} \cup Q_{\text{out}}$ and the free points. We assume that all faces in the free region are triangles and there are no edges outside the free region. The graph T can be decomposed into cactus layers L_i as explained in Section 1.1. We call such a graph T of $\mathcal{S}$ a *valid triangulation of* $\mathcal{S}$ if the following conditions are satisfied:

1. The inner layer Q_{in} corresponds indeed with the innermost cactus layer of T.
2. The number of layers equals the width w.
3. In case that a layer constraint vector c is given, we require that each layer has the size given by c.

Note that Condition 3 subsumes Condition 2, in case a layer-constraint vector is given.

▶ **Theorem 9.** *There exists an algorithm that given an annotated layer-unconstrained ring subproblem $\mathcal{S}$ with n free points computes the number of all triangulations of $\mathcal{S}$ in time $n^{(11+o(1))\sqrt{n}}$.*

Given a set of points S, we can define a set of ring subproblems, such that each triangulation of S is a valid triangulation of exactly one of the ring subproblems. We use the convex hull as outer layer and the empty graph as inner layer. We do not need layer constraints. It is technical, but straightforward to show that this works indeed.

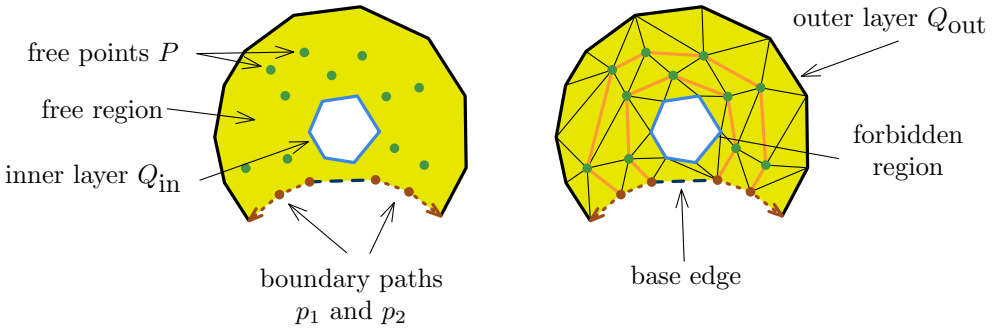


Figure 5 Left: A nibbled ring subproblem $\mathcal{S}$ consisting of a base edge depicted in dark blue dashed; two boundary paths, displayed in dotted brown from the base edge to the outer layer Q_{out}, displayed in black; the free region depicted in yellow; containing free points, depicted in green; The forbidden region is depicted in white. Right: A nibbled ring subproblem together with a valid triangulation. It consists of four layers. The middle two layers are colored orange. The base edge belongs to the third layer.

3 Thin Rings

This section sketches the proof of the following theorem, which gives an algorithm for solving ring subproblems with a certain width w. This algorithm will be invoked by the main algorithm for values $w \leq \sqrt{n}$. We will sketch the main ideas of the algorithm and its runtime analysis.

▶ **Theorem 10.** *There exists an algorithm that given a (layer-constrained or layer-unconstrained) ring subproblem $\mathcal{S}$ with width w and at most n free points, computes the number of all valid triangulations of $\mathcal{S}$ in time $n^{(5+o(1))w}$.*

Theorem 10 implies easily Theorem 2 in a similar fashion as Theorem 9 implies Theorem 3. We use path-separators for this algorithm. This requires a yet more specialized definition of subproblems for our dynamic programming scheme: *nibbled ring subproblems*.

We start with the key ingredients of *nibbled ring problems* omitting the parts that are equivalent to ring subproblems, see Figure 5 for an illustration of the nibbled ring subproblems. We omit the inner layer, the width, the free region, the free points and the layer constraints.

▶ **Definition 11** (Nibbled Ring Subproblem)**.** The outer layer is always a polygonal chain (i.e., a plane graph that is a path as a graph.). We have additionally two disjoint *boundary paths* p_1 and p_2 that end on the outer layer and start at the base edge. Note that the boundary paths are allowed to have length zero. The base edge, the boundary paths and the polygonal chain form a cycle that bounds the free region, see Figure 5.

Next we define valid triangulations for a nibbled ring subproblems. As for ring problems, we also want to define a cactus layer structure for nibbled ring problems that is consistent with the previous definition. The definition of cactus-layers does not generalize in a straightforward way to nibbled ring subproblems. Recall that the length of the shortest path from any vertex of the i-th layer to the boundary of the convex hull (outer layer) has length $i-1$. Therefore, we will define the layers in terms of distance to the outer layer. This way of defining layers is equivalent for triangulations of point sets and carries over to nibbled ring subproblems.

Another important ingredient is the concept of *order labels* of the vertices. Each vertex has a distinct order label from $1, \ldots, n$. The importance here is that the order label is defined as a preprocessing step. It is *never* altered during the algorithm.

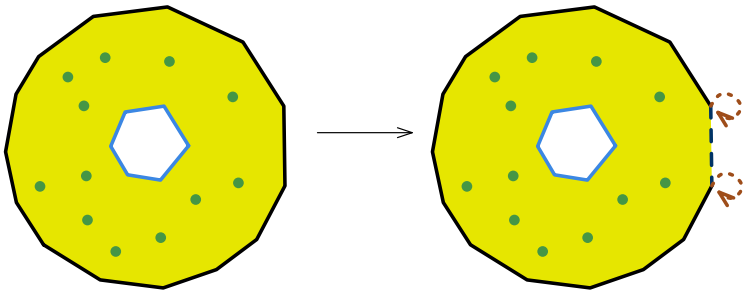


Figure 6 Left: a ring subproblem Right: the transformed nibbled ring subproblem.

▶ **Definition 12** (Valid Triangulation)**.** Given a nibbled ring subproblem $\mathcal{S}$ consider a triangulation T of $\mathcal{S}$. For each vertex $v \in V(T)$, we define $d(v)$ as the length of the shortest path to the outer-layer Q_{out}. We call T a *valid triangulation of* $\mathcal{S}$ if the conditions for ring subproblems and the following additional condition are satisfied:

4. For any vertex v_i of any boundary path $p = (v_1, \ldots, v_k)$, we have that $d(v_i) = d(v_{i+1}) + 1$ holds. Furthermore, we require that the neighbor v_{i+1} of v_i be the neighbor of v_i in T with the smallest order label among the neighbors with smaller distance to the outer layer.

▶ **Theorem 13.** *There exists an algorithm that given a nibbled ring subproblem $\mathcal{S}$ with width w computes the number of all valid triangulations of $\mathcal{S}$ in time $n^{(5+o(1))w}$.*

Theorem 13 easily implies Theorem 10, using a simple transformation from ring subproblems to nibbled ring subproblems, see Figure 6.

We start with a description of the separators, see Figure 7. Given a valid triangulation T of a nibbled ring subproblem $\mathcal{S}$, recall that every vertex on layer i has a neighbor w.r.t. T in layer $i-1$. Furthermore, there is a unique triangle Δ incident to the base edge. From vertex v of Δ, which is not incident to the base edge, there exists a path to the outer layer of $\mathcal{S}$ by always choosing an adjacent vertex closer to the outer layer. There is *exactly one* such path p, if we further require that the vertex with lowest order label is taken. (Recall that the order label is fixed in advance.) Such paths are called *canonical outgoing paths*.

Let us now move to the algorithmic scheme using the paths as separators. We recurse on a nibbled ring subproblem by guessing all potential such triangles Δ incident to the base edge and all potential canonical paths p as described above. For each such path, we can define two subproblems $\mathcal{S}_{\text{right}}(p)$ and $\mathcal{S}_{\text{left}}(p)$, see Figure 7. We can restrict our triangulation T to these subproblems and receive two new triangulations T_{left} and T_{right}, and conversely, given two triangulations T_{left} and T_{right}, we can combine it to a triangulation T. This is the point where property 4 of Definition 12 becomes relevant: if T_{left} and T_{right} satisfy this property, then it will be true for T that the path p is the canonical outgoing path starting at vertex v of Δ.

Thus we count recursively the number of valid triangulations of subproblems $\mathcal{S}_{\text{right}}$ and $\mathcal{S}_{\text{left}}$ and get *exactly* the number of valid triangulations of $\mathcal{S}$ where Δ is the triangle incident to the base edge and p is the canonical outgoing path starting at vertex v of Δ. More precisely, we sum over all potential canonical outgoing paths p and multiply the number of valid triangulations for $\mathcal{S}_{\text{left}}(p)$ with the number of valid triangulations for $\mathcal{S}_{\text{right}}(p)$.

If there are layer constraints in $\mathcal{S}$, then we have to do some more work. Let d, d_{left}, and d_{right} be the vectors that indicate the size of the layers for T, T_{left}, and T_{right} respectively.

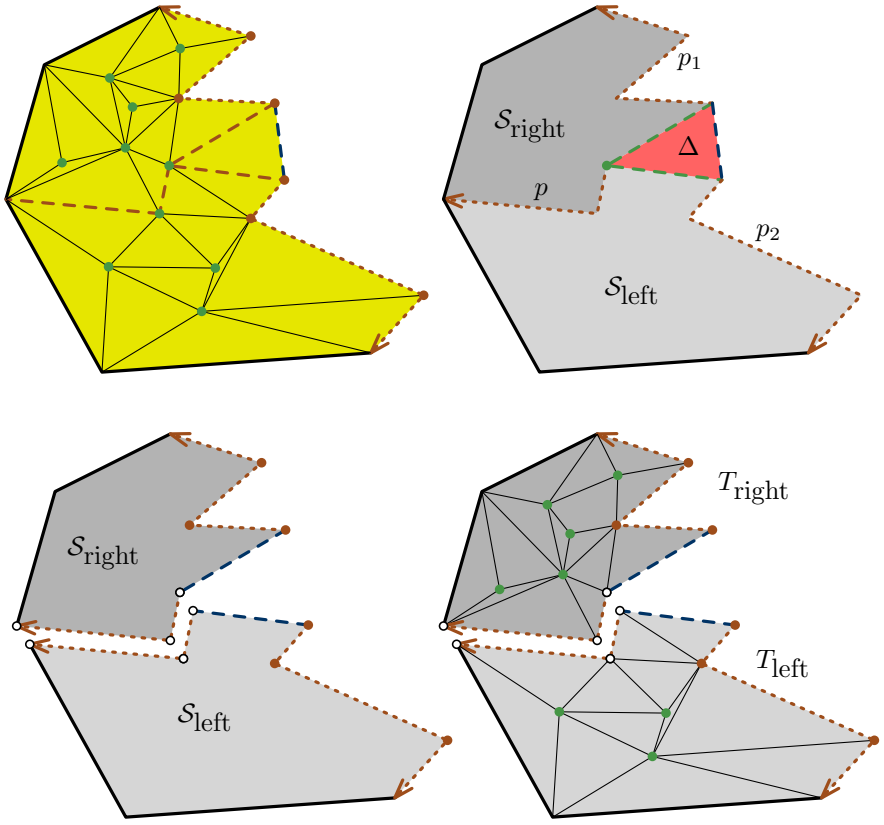


Figure 7 On the top left is a nibbled ring subproblem $\mathcal{S}$ together with a valid triangulation T. There is a unique triangle Δ adjacent to the base edges. From the vertex v of Δ that is not incident to the base edge exists a path p to the outer layer. The triangle Δ and the path p are drawn dashed brown. The path p is uniquely determined if we always use the vertex with the smallest order label among all available choices. On the top right the nibbled ring problem $\mathcal{S}$ is depicted together with the separator path p that splits it into two subproblem $\mathcal{S}_{\text{right}}$ and $\mathcal{S}_{\text{left}}$. At the bottom left, both subproblems are displayed. The three white vertices are shared. At the bottom right the restricted triangulations T_{left} and T_{right} are displayed.

Except for the vertices shared by T_{left} and T_{right}, it holds that d equals $d_{\text{left}} + d_{\text{right}}$. Now, let us go back to our subproblems $\mathcal{S}_{\text{left}}$ and $\mathcal{S}_{\text{right}}$. Let c be the layer-constraint vector of $\mathcal{S}$. Then, we define all pairs of *compatible* layer-constraints $(c_{\text{left}}, c_{\text{right}})$ such that two valid triangulations for $\mathcal{S}_{\text{right}}(c_{\text{right}})$ and $\mathcal{S}_{\text{left}}(c_{\text{left}})$ respectively give a triangulation for $\mathcal{S}$ with the correct number of vertices on each layer. Thus for each pair $\mathcal{S}_{\text{left}}$ and $\mathcal{S}_{\text{right}}$ of nibbled ring subproblems we sum over all pairs of compatible layer constraint vectors c_{left} and c_{right} and multiply the number of valid triangulations of $\mathcal{S}_{\text{left}}(c_{\text{left}})$ with the number of valid triangulations of $\mathcal{S}_{\text{right}}(c_{\text{right}})$.

Here, the technical difficulty is to take into consideration the vertices shared by both subproblems. Further we need to ensure that vertices in the i-th layer of $\mathcal{S}_{\text{right}}$ will also be in the i-th layer of $\mathcal{S}$. We recurse on all subproblems occurring in this way.

Proof sketch (of Theorem 13). The correctness of the algorithm follows from the correctness of the recursion. The bound on the running time follows from bounding the time required to solve a subproblem times the number of subproblems. We save our intermediate results in a search tree in order to prevent to handle any subproblem more than once. The bound on

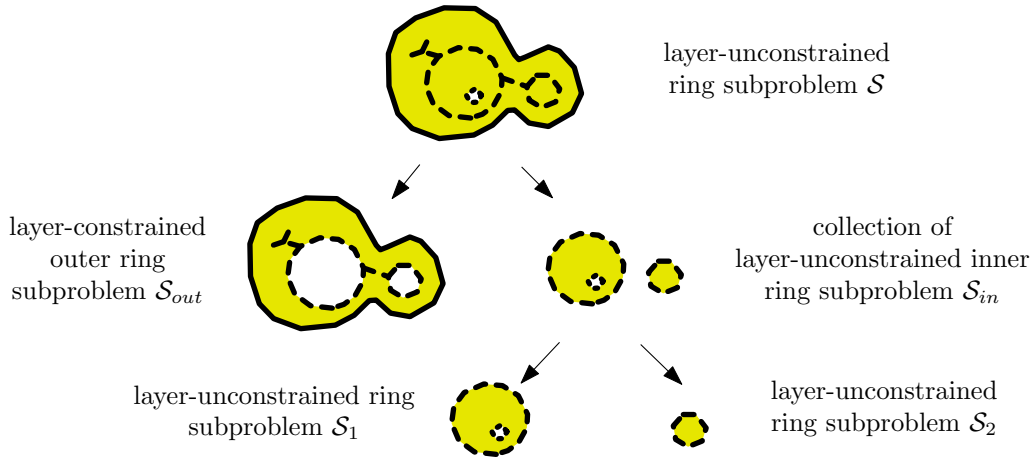


Figure 8 On the top is a ring subproblem $\mathcal{S}$ depicted with the outer layer drawn with solid lines, the inner layer in dotted and a separator layer is dashed. In the middle left the outer ring subproblem is depicted and on the middle right is the inner ring subproblem depicted. At the bottom two layer-unconstrained ring subproblems are depicted. The free region is marked yellow in all cases.

the number of subproblems follows from the fact that all of their items are fixed by at most two path separators, and from the fact that a separator has at most length w, thus there are at most $n^{O(w)}$ of them. The number of layer constraints is bounded by the assumption that at most w layers are constrained. The time for the recursive steps for one subproblem can be asymptotically bounded by the number of recursions, which in turn depends only on the number of potential canonical paths and ways to split the layer constraints in a compatible way. ◀

4 Layer-Unconstrained Ring Subproblems

We sketch the main algorithm in this section and estimate its running time.

The way we solve general ring subproblems is to distinguish two cases. In the case that the ring subproblem is thin, that is, has only few layers ($\leq \sqrt{n}$), we will use the algorithm of Theorem 10 as explained in Section 3. In case that we have many layers ($w > \sqrt{n}$), we know that one of the outermost $\sqrt{n}$ layers must be of size $\leq \sqrt{n}$ by the pigeon hole principle. We use this layer as a separator that splits the problem into a thin outer part and an inner part.

In the course of the algorithm, we will *never* add layer-constraint vectors to ring subproblems with more than $\sqrt{n}$ layers. Thus we do not have to deal in this part of the algorithm with layer-constraint vectors.

We start with a description of the *layer-separators*: To be more explicit, let $\mathcal{S}$ be a ring subproblem and let T be a valid triangulation of $\mathcal{S}$. The width w tells us already the exact number of layers that T has. Consider the case that T has more than $\sqrt{n}$ layers. Then among the $\sqrt{n}$ layers closest to the outer layer, one must have size less than or equal to $\sqrt{n}$. Note that the layer L that is actually closest to the outer layer of $\mathcal{S}$ is *uniquely* determined. This layer is our separator layer.

The algorithm works as follows: We guess all potential separator layers L, which requires guessing at most $\sqrt{n}$ points and $O(\sqrt{n})$ edges. Each guess defines an inner ring subproblem $\mathcal{S}_{\text{in}}$ and an outer ring subproblem $\mathcal{S}_{\text{out}}$, as depicted in Figure 8. We recurse on all $\mathcal{S}_{\text{in}}$ and $\mathcal{S}_{\text{out}}$ created in this way. To compute the number of valid triangulations of $\mathcal{S}$, we sum over

all separator layers L and multiply the number of valid triangulations of $\mathcal{S}_{\text{in}}$ with the number of valid triangulations of $\mathcal{S}_{\text{out}}$, for each appearing L.

In case the cactus layer L is disconnected or has disconnected bounded faces, we deal with a collection of inner ring subproblems. In this case, we recurse on each inner ring subproblem separetly, and multiply the numbers of valid triangulations of each ring subproblem.

We can restrict T to $\mathcal{S}_{\text{out}}$ to attain a triangulation T_{out}. It is clear that all layers different from L and Q_{out} in T_{out} have size larger than $\sqrt{n}$. Therefore, we want to count only those triangulations of $\mathcal{S}_{\text{out}}$ that have all layers (except L and Q_{out}) of size larger than $\sqrt{n}$. We use layer constraints for this purpose: we solve $\mathcal{S}_{\text{out}}$ with every possible layer constraint where every layer is require to have size greater than $\sqrt{n}$. Thus, we sum over all such layer-constraint vector c the number of valid triangulations of $\mathcal{S}_{\text{out}}(c)$. Also note that $\mathcal{S}_{\text{out}}$ has at most $\sqrt{n}$ layers and thus the number of constraint layers are at most $\sqrt{n}$ holds by induction. Note that we do not recurse on the outer ring subproblems by the main algorithm, but rather solve them with the algorithm of Section 3. The runtime is given in Theorem 10.

The running time can be estimated by bounding the total number of ring subproblems times the time spent per ring subproblem. Each layer-unconstrained ring subproblem is defined by an inner and an outer layer. In the course of the algorithm only inner and outer layers of size less than or equal to $\sqrt{n}$ are guessed, and there are at most $n^{O(\sqrt{n})}$ of them. In the recursion step, we either guess all potential layer separators or we deal with rings of width smaller than $\sqrt{n}$. In the first case there are at most $n^{O(\sqrt{n})}$ many guesses. In the second case, the runtime is given by Theorem 10. Thus the total running time is bounded by $n^{O(\sqrt{n})} \cdot n^{O(\sqrt{n})} = n^{O(\sqrt{n})}$.

5 Counting Annotated Triangulations

Here we describe how we adapt the algorithm to also count annotated triangulations with given annotation system L. The adaptation consists of a few straightforward steps. Most prominently, whenever we guess a separator, we guess an annotation for all k_1 vertices and k_2 edges that we guessed. It is easy to see that there are at most $|L|^{k_1+k_2}$ possible guesses. The next step is to check each time we guess a triangle whether the triangle belongs to L. The last step is more subtle. We need to guess also annotations on the outer layer of our initial ring subproblems. However, the initial outer layer is the convex hull, which might contain a linear number of vertices. To circumvent this, we use a standard trick to add a triangle Δ containing the whole point set. This adds some additional constraints on the triangulations we want to count, but circumvents the problem of guessing the annotations of the boundary of the convex hull.

6 Applications for Counting Other Structures

In this section, we develop a framework for counting non-crossing straight line graphs (e.g., non-crossing matchings or non-crossing Hamiltonian cycles), based on our algorithms for counting annotated triangulations. The key idea here is to introduce a new color and extend the graph into a triangulation by filling it with edges of this new color in a *canonical* way. For these filler-edges, we use the constrained Delaunay triangulation, in a similar fashion as previous authors have done already [2, 4]. We give only a sketch of the argument here.

As a running example of this section, let us consider the problem of counting non-crossing perfect matchings. We show how this problem can be reduced to counting annotated triangulations and then Theorem 3 can be invoked to obtain a subexponential-time algorithm.

▶ **Theorem 14** (Counting Perfect Matchings). *There exists an algorithm that, given a set S of n points in the plane, counts the total number of non-crossing perfect straight line matchings in $n^{O(\sqrt{n})}$ time.*

The obvious idea of reducing counting perfect matchings to counting annotated triangulations is the following. Let us say that the edges appearing in a perfect matchings have red color and let us extend a perfect matching to a triangulation with edges of color blue. It is not very difficult to construct an annotation system enforcing that each edge of the triangulation is either red or blue, and each vertex has exactly one red edge incident to it. One way of doing this would be to annotate each vertex v with the index of the other endpoint v' of its red edge and then to enforce this interpretation by forbidding any triangle incident to v that contains an edge separating v from v'. Now counting triangulations with this annotation system *overcounts* the number of perfect matchings: each perfect matching can be extended into one or more triangulations satisfying the annotation system. Thus we need to further restrict the annotation system in a way that ensures that each perfect matching has exactly one, canonical extension.

Let G be a straight line graph on a set of n points and T be a triangulation extending G on the same set of points. We say that T is the *constrained Delaunay triangulation* of G if every edge $e \in E(T) \setminus E(G)$ satisfies the *Delaunay Condition*, that is, the circumference of neither adjacent triangle contains the other adjacent triangle. (In case that e is on the boundary of the convex hull of S, then we define that the Delaunay Condition is automatically satisfied for e.) It is known that each G has a unique constrained Delaunay triangulation.

▶ **Theorem 15** ([29]). *Given a straight line graph G on a set of n points, there exists exactly one constrained Delaunay triangulation T extending G.*

It is not very difficult to construct an annotation system that enforces that the Delaunay Condition holds for every blue edge. First, we annotate each blue edge with a pair of vertices: a pair containing the third vertex of each of the two triangles incident to the edge. This interpretation is easy to enforce. Then we allow only those annotations where the two triangles of the edge satisfy the Delaunay Condition: neither of them is contained in the circumscribed circle of the other. It is clear that now any valid annotated triangulation has the property that it is the Delaunay triangulation of the red edges and in particular any set of red edges has a unique extension into a valid annotated triangulation.

Combining the two annotation systems, we arrive to an annotation system where the valid annotated triangulations are exactly the Delaunay triangulations of non-crossing perfect matchings. Thus counting the number of annotated triangulations using the algorithm of Theorem 3 counts the number of non-crossing perfect matchings. This proves Theorem 14.

In a similar way, we can construct annotation systems requiring the red edges form other non-crossing structures such as Hamiltonian cycles, spanning trees, quadrangulations, and many more. Combining these annotation systems with the annotation system enforcing the Delaunay Condition on the blue edges, we can construct annotation systems counting these structures as well. This allows us to prove Theorem 6 as stated in the introduction.

References

1 Oswin Aichholzer. The path of a triangulation. In *SoCG'99*, pages 14–23. ACM, 1999.

2 Victor Alvarez, Karl Bringmann, Radu Curticapean, and Saurabh Ray. Counting triangulations and other crossing-free structures via onion layers. *D&C*, 53(4):675–690, 2015. doi:10.1007/s00454-015-9672-3.

3 Victor Alvarez, Karl Bringmann, and Saurabh Ray. A simple sweep line algorithm for counting triangulations and pseudo-triangulations. *CoRR*, abs/1312.3188, 2013. URL: http://arxiv.org/abs/1312.3188.

4 Victor Alvarez, Karl Bringmann, Saurabh Ray, and Raimund Seidel. Counting triangulations and other crossing-free structures approximately. *Comput. Geom.*, 48(5):386–397, 2015. doi:10.1016/j.comgeo.2014.12.006.

5 Victor Alvarez and Raimund Seidel. A simple aggregative algorithm for counting triangulations of planar point sets and related problems. In *SoCG'13*, pages 1–8, 2013. doi:10.1145/2462356.2462392.

6 Efthymios Anagnostou and Derek Corneil. Polynomial-time instances of the minimum weight triangulation problem. *Computational Geometry*, 3(5):247–259, 1993.

7 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996.

8 Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. *Computing in Euclidean geometry*, 1:23–90, 1992.

9 B. Chazelle. Triangulating a simple polygon in linear time. *D&C*, 6:485–524, 1991. doi:10.1007/BF02574703.

10 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions). In *SODA*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.

11 Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Cheong Schwarzkopf. *Computational geometry*. Springer, 2000.

12 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Bidimensional parameters and local treewidth. *SIAM J. Discrete Math.*, 18(3):501–511, 2004. doi:10.1137/S0895480103433410.

13 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Fixed-parameter algorithms for (k, r)-Center in planar graphs and map graphs. *ACM Transactions on Algorithms*, 1(1):33–47, 2005. doi:10.1145/1077464.1077468.

14 Erik D. Demaine, Fedor V. Fomin, Mohammad Taghi Hajiaghayi, and Dimitrios M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005. doi:10.1145/1101821.1101823.

15 Erik D. Demaine and Mohammad Taghi Hajiaghayi. Fast algorithms for hard graph problems: Bidimensionality, minors, and local treewidth. In *Graph Drawing*, pages 517–533, 2004. doi:10.1007/978-3-540-31843-9_57.

16 Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *Comput. J.*, 51(3):292–302, 2008. doi:10.1093/comjnl/bxm033.

17 Erik D. Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008. doi:10.1007/s00493-008-2140-4.

18 Frederic Dorn, Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Beyond bidimensionality: Parameterized subexponential algorithms on directed graphs. In *STACS*, pages 251–262, 2010. doi:10.4230/LIPIcs.STACS.2010.2459.

19 Frederic Dorn, Fedor V. Fomin, and Dimitrios M. Thilikos. Subexponential parameterized algorithms. *Computer Science Review*, 2(1):29–39, 2008. doi:10.1016/j.cosrev.2008.02.004.

20 Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010. doi:10.1007/s00453-009-9296-1.

21 Adrian Dumitrescu, Bernd Gärtner, Samuele Pedroni, and Emo Welzl. Enumerating triangulation paths. *Computational Geometry*, 20(1):3–12, 2001.

22 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

23 Steve Fisk. A short proof of chvátal's watchman theorem. *J. Comb. Theory*, 24(3):374, 1978. doi:10.1016/0095-8956(78)90059-X.

24 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, and Saket Saurabh. Subexponential algorithms for partial cover problems. *Inf. Process. Lett.*, 111(16):814–818, 2011. doi:10.1016/j.ipl.2011.05.016.

25 Fedor V. Fomin and Dimitrios M. Thilikos. Dominating sets in planar graphs: Branch-width and exponential speed-up. *SIAM J. Comput.*, 36(2):281–309, 2006. doi:10.1137/S0097539702419649.

26 Peter D Gilbert. New results on planar triangulations. Technical report, 1979.

27 Leonidas J. Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert Endre Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987. doi:10.1007/BF01840360.

28 John Hershberger and Subhash Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, 1995.

29 Øyvind Hjelle and Morten Dæhlen. *Triangulations and Applications*. Springer, 2006.

30 Marek Karpinski, Andrzej Lingas, and Dzmitry Sledneu. A QPTAS for the base of the number of crossing-free structures on a planar point set. In *ICALP'15*, pages 785–796, 2015. doi:10.1007/978-3-662-47672-7_64.

31 Philip N. Klein and Dániel Marx. Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time. In *ICALP (1)*, pages 569–580, 2012. doi:10.1007/978-3-642-31594-7_48.

32 Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for Subset TSP on planar graphs. In *SODA*, pages 1812–1830, 2014. doi:10.1137/1.9781611973402.131.

33 GT Klincsek. Minimal triangulations of polygonal domains. *Ann. Discrete Math*, 9:121–123, 1980.

34 Christian Knauer and Andreas Spillner. A fixed-parameter algorithm for the minimum weight triangulation problem based on small graph separators. In Fedor V. Fomin, editor, *Graph-Theoretic Concepts in Computer Science, 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006, Revised Papers*, volume 4271 of *Lecture Notes in Computer Science*, pages 49–57. Springer, 2006. doi:10.1007/11917496_5.

35 Andrzej Lingas. Subexponential-time algorithms for minimum weight triangulations and related problems. In *CCCG'98*, 1998. URL: http://cgm.cs.mcgill.ca/cccg98/proceedings/cccg98-lingas-subexponential.ps.gz.

36 Richard J Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

37 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

38 Dániel Marx and Tillmann Miltzow. Peeling and nibbling the cactus: Subexponential-time algorithms for counting triangulations and related problems. *CoRR*, to be announced, 2016.

39 Dániel Marx and Michal Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In *ESA'15*, pages 865–877. Springer, 2015. doi:10.1007/978-3-662-48350-3_72.

40 Marcin Pilipczuk, Michal Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for steiner problems on planar and bounded-genus graphs. In *FOCS'14*, pages 276–285. doi:10.1109/FOCS.2014.37.

41 Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Subexponential-time parameterized algorithm for Steiner Tree on planar graphs. In *STACS*, pages 353–364, 2013. doi:10.4230/LIPIcs.STACS.2013.353.

42 S Ray and R Seidel. A simple and less slow method for counting triangulations and for related problems. In *EuroCG'04*, pages 77–80, 2004.

43 Micha Sharir and Adam Sheffer. Counting triangulations of planar point sets. *Electr. J. Comb.*, 18(1), 2011. URL: http://www.combinatorics.org/Volume_18/Abstracts/v18i1p70.html.

44 Dimitrios M. Thilikos. Fast sub-exponential algorithms and compactness in planar graphs. In *ESA*, pages 358–369, 2011. doi:10.1007/978-3-642-23719-5_31.

Convergence between Categorical Representations of Reeb Space and Mapper*

Elizabeth Munch[1] **and Bei Wang**[2]

1 **Department of Mathematics & Statistics, University at Albany – SUNY, Albany, USA**
emunch@albany.edu

2 **Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, USA**
beiwang@sci.utah.edu

Abstract

The Reeb space, which generalizes the notion of a Reeb graph, is one of the few tools in topological data analysis and visualization suitable for the study of multivariate scientific datasets. First introduced by Edelsbrunner et al., it compresses the components of the level sets of a multivariate mapping and obtains a summary representation of their relationships. A related construction called mapper, and a special case of the mapper construction called the Joint Contour Net have been shown to be effective in visual analytics. Mapper and JCN are intuitively regarded as discrete approximations of the Reeb space, however without formal proofs or approximation guarantees. An open question has been proposed by Dey et al. as to whether the mapper construction converges to the Reeb space in the limit.

In this paper, we are interested in developing the theoretical understanding of the relationship between the Reeb space and its discrete approximations to support its use in practical data analysis. Using tools from category theory, we formally prove the convergence between the Reeb space and mapper in terms of an interleaving distance between their categorical representations. Given a sequence of refined discretizations, we prove that these approximations converge to the Reeb space in the interleaving distance; this also helps to quantify the approximation quality of the discretization at a fixed resolution.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Topological data analysis, Reeb space, mapper, category theory

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.53

1 Introduction

Motivation and prior work. Multivariate datasets arise in many scientific applications, ranging from oceanography to astrophysics, from chemistry to meteorology, from nuclear engineering to molecular dynamics. Consider, for example, combustion or climate simulations where multiple physical measurements (e.g. temperature and pressure) or concentrations of chemical species are computed simultaneously. We model these variables mathematically as multiple continuous, real-valued functions defined on a shared domain, which constitute a multivariate mapping $f : \mathbb{X} \to \mathbb{R}^d$, also known as a multi-field. We are interested in understanding the relationships between these real-valued functions, and more generally, in developing efficient and effective tools for their analysis and visualization.

* This work was partially support by NSF IIS-1513616.

© Elizabeth Munch and Bei Wang;
licensed under Creative Commons License CC-BY

Recently, topological methods have been developed to support the analysis and visualization of scalar field data with widespread applicability. In particular, a great deal of work for scalar topological analysis has been focused on computing the Reeb graph [21]. The Reeb graph contracts each contour (i.e. component of a level set) of a real-valued function to a single point and uses a graph representation to summarize the connections between these contours. When the domain is simply connected, this construction forms a contour tree, which has been shown to be effective in many applications including data simplification and exploratory visualization [3]. From a computational perspective, both randomized [11] and deterministic [18] algorithms exist that compute the Reeb graph for a function defined on a simplicial complex K in time $O(m \log m)$, where m is the total number of vertices, edges and triangles in K. Recent work by de Silva et al. [7] has shown that the data of a Reeb graph can be stored in a category-theoretic object called a cosheaf, which opens the way for defining a metric for Reeb graphs known as the interleaving distance. The idea of utilizing a cosheaf over a simplicial complex has also been previously investigated, in particular in the work of Curry [6].

Unlike for real-valued functions, very few tools exist for studying multivariate data topologically as the situation becomes much more complicated. The most notable examples of these tools are the Jacobi set [9] and the Reeb space [10]. The Jacobi set analyzes the critical points of a real-valued function restricted to the intersection of the level sets of other functions. On the other hand, the Reeb space, a generalization of the Reeb graph, compresses the components of the level sets of the multivariate mapping (i.e. $f^{-1}(c)$, for $c \in \mathbb{R}^d$) and obtains a summary representation of their relationships. These two concepts are shown to be related as the image of the Jacobi sets under the mapping corresponds to certain singularities in the Reeb space. An algorithm has been described by Edelsbrunner et al. [10] to construct the Reeb space of a generic piecewise-linear (PL), $\mathbb{R}^d$-valued mapping defined on a combinatorial manifold up to dimension 4. Let n be the number of $(d-1)$-simplices in the combinatorial manifold. Assuming d is a constant, the running time of the algorithm is $O(n^d)$, polynomial in n [19].

A related construction called mapper [22] takes as input a multivariate mapping and produces a summary of the data by using a cover of the range space of the mapping. Such a summary converts the mapping with a fixed cover into a simplicial complex for efficient computation, manipulation, and exploration [14, 17]. When the mapping is a real-valued function (i.e. $d = 1$) and the cover consists of a collection of open intervals, it is stated without proof that the mapper constrcution recovers the Reeb graph precisely as the scale of the cover goes to zero [22]. A similar combinatorial idea has also been explored with the α-Reeb graph [5], which is another relaxed notion of a Reeb graph produced by a cover of the range space consisting of open intervals of length at most α. Recently, Dey et al. [8] extended mapper to its multiscale version by considering a hierarchical family of covers and the maps between them. At the end of their exposition, the authors raised an open question in understanding the continuous object that the mapper construction converges to as the scale of the cover goes to zero, in particular, whether the mapper construction converges to the Reeb space. In addition, Carr and Duke [2] introduced a special case of mapper called the Joint Contour Net (JCN) together with its efficient computation, for a PL mapping defined over a simplicial mesh involving an arbitrary number of real-valued functions. Based on a cover of the range space using d-dimensional intervals, the JCN quantizes the variation of multiple variables simultaneously by considering connected components of interval regions (i.e. $f^{-1}(a, b)$) instead of the connected components of level sets (i.e. $f^{-1}(c)$). It can be computed in time $O(km\alpha(km))$, where m is the size of the input mesh, k is the total number

of quantized interval regions, and α is the slow-growing inverse Ackermann function [2]. The authors stated that the JCN can be considered as a discrete approximation that converges in the limit to the Reeb space [2], although this statement was supported only by intuition and lacked approximation guarantees.

Contributions. In this paper, we are interested in developing theoretical understandings between the Reeb space and its discrete approximations to support its use in practical data analysis. Using tools from category theory, we formally prove the convergence between the Reeb space and mapper in terms of an interleaving distance between their categorical representations (Theorem 1). Given a sequence of refined discretizations, we prove that these approximations converge to the Reeb space in the interleaving distance; this also helps to quantify the approximation quality of the discretization at a fixed resolution. Such a result easily generalizes to special cases of mapper such as the JCN. Our work extends and generalizes the tools from the categorical representation of Reeb graphs [7] to a new categorical framework for Reeb spaces. In particular, we provide for the first time the definition of the interleaving distance for Reeb spaces (Definition 2). We demonstrate that such a distance is an extended pseudometric (Theorem 3) and it provides a simple and formal language for structural comparisons. Finally in the setting of Reeb graphs (when $d = 1$), we demonstrate that mapper converges to the Reeb graph geometrically on the space level (Corollary 6). We further provide an algorithm for constructing a continuous representation of mapper geometrically from its categorical representation.

2 Topological Notions

We now review the relevant background on the Reeb space [10, 19] and mapper[1] [8, 22]. In theory, we assume the data given is a compact topological space $\mathbb{X}$ with an $\mathbb{R}^d$-valued function, $f : \mathbb{X} \to \mathbb{R}^d$, often denoted $(\mathbb{X}, f)$. In practice, we assume the data we work with is a multivariate PL mapping f defined over a simplicial mesh; more restrictively (for easier exposition of our algorithms and proofs), we consider a generic, PL mapping f from a combinatorial manifold [20] to $\mathbb{R}^d$.

Reeb Space. Let $f : \mathbb{X} \to \mathbb{R}^d$ be a generic, continuous mapping[2]. Intuitively, the Reeb space of f parametrizes the set of components of preimages of points in $\mathbb{R}^d$ [10]. Two points $x, y \in \mathbb{X}$ are equivalent, denoted by $x \sim_f y$, if $f(x) = f(y)$ and x and y belong to the same path connected component of the preimage, $f^{-1}(f(x)) = f^{-1}(f(y))$. The *Reeb space* is the quotient space obtained by identifying equivalent points, that is, $\mathcal{R}(\mathbb{X}, f) = \mathbb{X}/\sim_f$, together with the quotient topology inherited from $\mathbb{X}$. A powerful analysis tool, the *Reeb graph*, can be considered a special case in this context when $d = 1$. Reeb spaces have been shown to have triangulations and canonical stratifications into manifolds for nice enough starting data [10].

Mapper. An open cover of a topological space $\mathbb{X}$ is a collection $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of open sets for some indexing set A such that $\bigcup_{\alpha \in A} U_\alpha = \mathbb{X}$. In this paper, we will always assume that each U_α is path-connected and a cover means a finite open cover. We define a finite open cover $\mathcal{U}$ to be a *good* cover if every finite nonempty intersection of sets in $\mathcal{U}$ is contractible. Given

[1] Mapper was originally referred to as a method [22], however we refer to it as a topological construction/object in this paper.
[2] For simplicity, assume f is a PL mapping defined on a combinatorial manifold.

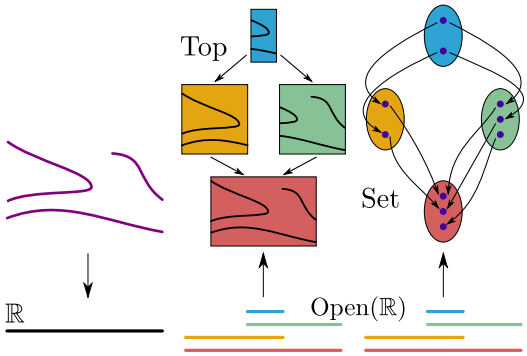


Figure 1 The data of a Reeb graph (on the left) can be stored as a functor. First, we give the middle functor f^{-1} : **Open**($\mathbb{R}$) $\to$ **Top** which sends each open set I to the topological space $f^{-1}(I)$; and sends each inclusion map between open sets $I \subseteq J$ to an inclusion map $f^{-1}(I) \to f^{-1}(J)$. Then the Reeb graph information is represented by composing this functor with the functor π_0 : **Top** $\to$ **Set**, producing a functor on the right $\pi_0 f^{-1}$: **Open**($\mathbb{R}$) $\to$ **Set**. Via π_0, the inclusion maps on the topological spaces become set maps.

a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ of $\mathbb{X}$, let $\mathrm{Nrv}(\mathcal{U})$ denote the simplicial complex that corresponds to the *nerve* of the cover $\mathcal{U}$, $\mathrm{Nrv}(\mathcal{U}) = \left\{\sigma \subseteq A \mid \bigcap_{\alpha \in \sigma} U_\alpha \neq \emptyset\right\}$. Given a (potentially multivariate) continuous map $f : \mathbb{X} \to \mathbb{Y}$ where $\mathbb{Y}$ is equipped with a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, we write $f^*(\mathcal{U})$ as the cover of $\mathbb{X}$ obtained by considering the path connected components of $f^{-1}(U_\alpha)$ for each α. Given such a function f, its *mapper construction* (or *mapper* for short) M is defined to be the nerve of $f^*(\mathcal{U})$, $M(\mathcal{U}, f) := \mathrm{Nrv}(f^*(\mathcal{U}))$ [22]. Intuitively, considering a real-valued function $f : \mathbb{X} \to \mathbb{R}$ and a cover $\mathcal{U}_\varepsilon$ of image$(f) \subseteq \mathbb{R}$ consisting of intervals of length at most ε, the corresponding mapper $M(\mathcal{U}_\varepsilon, f)$ can be thought of as a relaxed Reeb graph that has been conjectured to converge to the Reeb graph of f as ε tends to zero [8, 22], although no formal proofs have been previously provided.

3 Categorical Notions

Category and opposite category. Category theory [15] can be thought of as a generalization of set theory in the sense that the item of study is still a set (technically a proper class), but now we are additionally interested in studying the relationships between the elements of the set. Mathematically, a *category* is an algebraic structure that consists of mathematical *objects* with a notion of *morphisms* (colloquially referred to as *arrows*) between the objects. A category has the ability to compose the arrows associatively, and there is an identity arrow for each object. Examples are abundant and those important to our exposition are: the category of topological spaces (as the objects) with continuous functions between them (as the arrows), denoted as **Top**; the category of sets with set maps, denoted as **Set**; the category of open sets in $\mathbb{R}^d$ with inclusion maps, denoted as **Open**($\mathbb{R}^d$); the category of vector spaces with linear maps, denoted as **Vect**; and the category of real numbers with inequalities connecting them, denoted as **R**. In addition, any simplicial complex K induces a category **Cell**(K) where the objects are the simplices of K, and there is a morphism $\sigma \to \tau$ if σ is a face of τ. Intuitively, we could think of a category as a big (probably infinite) directed multi-graph with extra underlying structures (due to the associativity and identity axioms obeyed by the arrows): the objects are the nodes, and each possible arrow between the nodes is represented as a directed edge. One common example used extensively throughout this paper is the idea of a *poset category*, which is a category **P** in which any pair of elements $x, y \in \mathbf{P}$ has at most one arrow $x \to y$. Categories such as **Open**($\mathbb{R}^d$) and **R** are poset

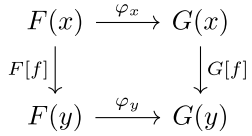


Figure 2 The diagram for a natural transformation.

categories since there is exactly one arrow $I \to J$ between open sets if $I \subseteq J$ and exactly one arrow $a \to b$ between real numbers if $a \leq b$. We often abuse notation and denote arrows in this category by the relation providing the poset structure, e.g. $I \subseteq J$ instead of $I \to J$ and $a \leq b$ instead of $a \to b$. In the graph description, a poset category can be thought of as a directed graph which is not a multigraph. The *opposite category* (or dual category) $\mathcal{C}^{op}$ of a given category $\mathcal{C}$ is formed by reversing the arrows (morphisms), i.e. interchanging the source and target of each arrow.

Functor. A *functor* is a map between categories that maps objects to objects and arrows to arrows. A functor $F : \mathcal{C} \to \mathcal{D}$ for categories $\mathcal{C}$ and $\mathcal{D}$ maps an object x in $\mathcal{C}$ to an object $F(x)$ in $\mathcal{D}$, and maps an arrow $f : x \to y$ of $\mathcal{C}$ to an arrow $F[f] : F(x) \to F(y)$ of $\mathcal{D}$ in a way that respects the identity and composition laws. In the above graph allegory, a functor is a map between graphs which sends nodes (objects) to nodes and edges (arrows) to edges in a way that is compatible with the structure of the graphs. An example of a functor is the homology functor $H_p : \mathbf{Top} \to \mathbf{Vect}$ which sends a topological space $\mathbb{X}$ to its p-th singular homology group $H_p(\mathbb{X})$ (a vector space assuming field coefficients), and sends any continuous map $f : \mathbb{X} \to \mathbb{Y}$ to the linear map between homology groups, $H_p[f] := f_* : H_p(\mathbb{X}) \to H_p(\mathbb{Y})$. Another functor used extensively in this paper is $\pi_0 : \mathbf{Top} \to \mathbf{Set}$ which sends a topological space $\mathbb{X}$ to a set $\pi_0(\mathbb{X})$ where each element represents a path connected component of $\mathbb{X}$, and sends a map $f : \mathbb{X} \to \mathbb{Y}$ to a set map $\pi_0[f] := f_* : \pi_0(\mathbb{X}) \to \pi_0(\mathbb{Y})$.

Natural transformation. We can make any collection of functors of the form $F : \mathcal{C} \to \mathcal{D}$ into a category by defining arrows between the functors. A *natural transformation* $\varphi : F \Rightarrow G$ between functors $F, G : \mathcal{C} \to \mathcal{D}$ is a family of arrows φ in $\mathcal{D}$ such that (a) for each object x of $\mathcal{C}$, we have $\varphi_x : F(x) \to G(x)$, an arrow of $\mathcal{D}$; and (b) for any arrow $f : x \to y$ in $\mathcal{C}$, $G[f] \circ \varphi_x = \varphi_y \circ F[f]$, that is, the diagram of Figure 2 commutes. Any collection of functors $F : \mathcal{C} \to \mathcal{D}$ can thus be turned into a category, with the functors themselves as objects and the natural transformations as arrows, notated as $\mathcal{D}^{\mathcal{C}}$. This notation is used heavily throughout this paper where always $\mathcal{D} = \mathbf{Set}$. If for every object x of $\mathcal{C}$, the arrow φ_x is an isomorphism in $\mathcal{D}$, then φ is a *natural isomorphism* (equivalence) of functors. Two functors F and G are *(naturally) isomorphic* if there exists a natural isomorphism from F to G.

Categorical Reeb graph. For a real-valued function $f : \mathbb{X} \to \mathbb{R}$, the data of its corresponding Reeb graph can be stored as a functor $F := \pi_0 f^{-1} : \mathbf{Open}(\mathbb{R}) \to \mathbf{Set}$, defined by sending each open set I to a set $F(I) := \pi_0 f^{-1}(I)$ that contains all the path connected components of $f^{-1}(I)$; and by sending an inclusion $I \subseteq J$ to a set map $F[I \subseteq J] : F(I) \to F(J)$ induced by the inclusion $f^{-1}(I) \subseteq f^{-1}(J)$. This is illustrated in Figure 1. The objects $F(I)$ store the connected components sitting over any open set; the information from the arrows $F(I) \to F(J)$ gives the information needed to glue together all of this data. This construction produces a categorical representation of the Reeb graph, referred to as the *categorical Reeb graph*. It was used in [7] to define the interleaving distance for Reeb graphs which we generalize to Reeb spaces in Section 5.

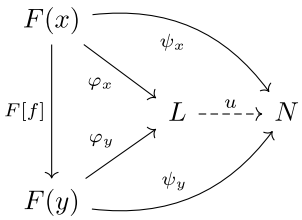


Figure 3 Defining a colimit.

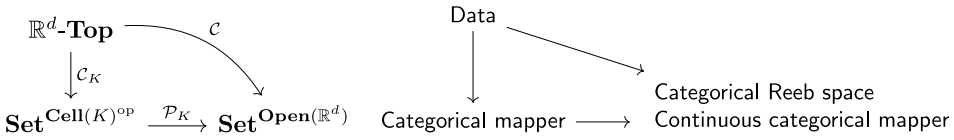


Figure 4 The diagram for connecting categorical representations of the Reeb space and the mapper. Note that the diagram is *not* commutative. Theorem 1 measures the amount that this diagram deviates from being commutative.

Colimit. The final category theoretic notion necessary for our results are colimits. The *cocone* (N, ψ) of a functor $F : \mathcal{C} \to \mathcal{D}$ is an object N of $\mathcal{D}$ along with a family of ψ of arrows $\psi_x : F(x) \to N$ for every object x of $\mathcal{C}$, such that for every arrow $f : x \to y$ in $\mathcal{C}$, we have $\psi_y \circ F[f] = \varphi_x$. We say that a cocone (N, ψ) factors through another cocone (L, φ) if there exists an arrow $u : L \to N$ such that $u \circ \varphi_x = \psi_x$ for every x in $\mathcal{C}$. The *colimit* of $F : \mathcal{C} \to \mathcal{D}$, denoted as $\operatorname{colim} F$, is a cocone (L, φ) of F such that for any other cocone (N, ψ) of F, there exists a unique arrow $u : L \to N$ such that (N, ψ) factors through (L, φ). In other words, the diagram of Figure 3 commutes. We often abuse notation by using $\operatorname{colim} F$ to represent just the object L. The colimit is universal; in particular, this means that if the colimit (L, φ) factors through another cocone (M, δ), then L is isomorphic to M and the isomorphism is given by the unique arrow $u' : M \to L$ that defines it. We will use this property in the proof of Lemma 5.

Because we often wish to consider these colimits over a full subcategory $\mathcal{A} \subseteq \mathcal{C}$, we will denote the restriction as $\operatorname{colim}_{A \in \mathcal{A}} F(A)$. The properties of a colimit also imply that if we have nested subcategories $\mathcal{A} \subseteq \mathcal{B}$ $(\subseteq \mathcal{C})$, then there is a unique map $\operatorname{colim}_{A \in \mathcal{A}} F(A) \to \operatorname{colim}_{B \in \mathcal{B}} F(B)$ since we can consider $\operatorname{colim}_{B \in \mathcal{B}} F(B)$ as cocone over $\mathcal{A}$.

4 Main Results Overview

The main focus of this paper is to provide a convergence result between the continuous Reeb space and the discrete mapper. We define their distance as the interleaving distance between their corresponding categorical representations and emphasize that neither the Reeb space nor the interleaving distance must ever be computed for this result. Instead, we provide a theoretical bound on the distance which requires only knowledge of the resolution of the cover. To define the desired distance measure, we use the diagram in Figure 4 as our roadmap. The remainder of this section is dedicated to describing the various categories at the nodes of the diagram as well as the functors that connect them.

Data. In our context, data comes in the form of a topological space $\mathbb{X}$ with an $\mathbb{R}^d$-valued mapping, called an $\mathbb{R}^d$-space. We store such data in the category $\mathbb{R}^d$-**Top**. Specifically,

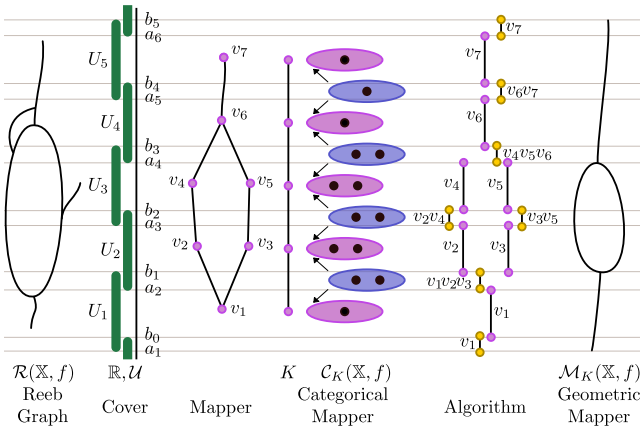


Figure 5 An example of a Reeb space for $d = 1$ (a Reeb graph), denoted as $\mathcal{R}(\mathbb{X}, f)$, is shown on the left. Its associated data $(\mathbb{X}, f)$ is an object in $\mathbb{R}^d$-**Top** with function f given by height. A cover $\mathcal{U}$ is shown by the green intervals, and the corresponding mapper is shown to its right. The mapper data is equivalently stored as the $C_K(\mathbb{X}, f)$ functor defined on an abstract simplicial complex $K = \mathrm{Nrv}(\mathcal{U})$. Note that although we draw K in the same plane as the other objects, it does not have a geometric embedding, nor does it have a natural map to $\mathbb{R}$. This is remedied with the geometric representation of this data, $\mathcal{M}_K(\mathbb{X}, f) := \mathcal{DP}_K\mathcal{C}_K(\mathbb{X}, f)$ which is shown at the far right. Corollary 6 asserts that the interleaving distance between the leftmost and rightmost graphs is bounded by $\varepsilon = \mathrm{res}(\mathcal{U})$.

an object of $\mathbb{R}^d$-**Top** is a pair consisting of a topological space $\mathbb{X}$ with a continuous map $f : \mathbb{X} \to \mathbb{R}^d$, denoted as $(\mathbb{X}, f)$. An arrow in $\mathbb{R}^d$-**Top**, $\nu : (\mathbb{X}, f) \to (\mathbb{Y}, g)$, is a function-preserving map; that is, it is a continuous map on the underlying spaces $\nu : \mathbb{X} \to \mathbb{Y}$ such that $g \circ \nu(x) = f(x)$ for all $x \in \mathbb{X}$. Note that many nice constructions such as PL functions on simplicial complexes or Morse functions on manifolds are objects in $\mathbb{R}^d$-**Top**.

Categorical Reeb space and its contruction. Recall the categorical representation of a Reeb graph is a functor $\mathbf{Open}(\mathbb{R}) \to \mathbf{Set}$. In order to define a categorical representation of the Reeb space, we need a higher dimensional analogue of $\mathbf{Open}(\mathbb{R})$, namely, $\mathbf{Open}(\mathbb{R}^d)$. $\mathbf{Open}(\mathbb{R}^d)$ is a category with open sets $I \subseteq \mathbb{R}^d$ as objects, and a unique arrow $I \to J$ if and only if $I \subseteq J$; that is, $\mathbf{Open}(\mathbb{R}^d)$ is a poset category. The data of the Reeb space can be stored as a functor $\pi_0 f^{-1} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$, defined by sending each open set I to a set $\pi_0 f^{-1}(I)$ representing the path connected components of $f^{-1}(I)$; and by sending the inclusion arrow $I \subseteq J$ to a set map $\pi_0 f^{-1}(I) \to \pi_0 f^{-1}(J)$ induced by the inclusion $f^{-1}(I) \subseteq f^{-1}(J)$. These functors, referred to as the *categorical Reeb spaces*, become objects of the category of functors $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$.

Constructing a Reeb space from the data is now represented by the functor $\mathcal{C} : \mathbb{R}^d$-**Top** $\to \mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ in Figure 4. In particular, $\mathcal{C}$ maps an object $(\mathbb{X}, f)$ in $\mathbb{R}^d$-**Top**, representing the data, to a functor $F : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$, representing its corresponding Reeb space. The functor $\mathcal{C}$ restricts to the Reeb graph construction when $d = 1$ [7]. In addition, from the generalized persistence module framework [1], we can also extend the idea of the interleaving distance between Reeb graphs (in the case $d = 1$) to these categorical Reeb spaces (in the case $d \geq 1$). The definition of functor $\mathcal{C}$ and the Reeb space interleaving distance are covered in Section 5.

Categorical mapper and its construction. Instead of working with continuous objects, we can instead choose a discretization represented by a simplicial complex K. Given a cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ for $\text{image}(f) \subseteq \mathbb{R}^d$, let $K = \text{Nrv}(\mathcal{U})$. Through the machinery detailed in Section 6, we create a categorical representation of the mapper (referred to as the *categorical mapper*) as a functor $F : \mathbf{Cell}(K)^{\text{op}} \to \mathbf{Set}$ (an object of $\mathbf{Set}^{\mathbf{Cell}(K)^{\text{op}}}$); and such a construction is represented by the $\mathcal{C}_K$ functor[3].

Comparing Reeb space and mapper. It should be noted that the Reeb space and the mapper are inherently different objects. The Reeb space comes equipped with an $\mathbb{R}^d$-valued function, while there is no such function built into the mapper even though its construction is highly dependent on the functions chosen to partition the data set [22]. In particular, the two objects are in completely different categories. So, to compare these objects, we study the image of the categorical mapper under the functor $\mathcal{P}_K$, which turns the categorical mapper (a discrete object) into a continuous one comparable with the categorical Reeb space. In particular, for data given as $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$, we compare its image in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ via the functor $\mathcal{C}$, to its image in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ via the functor $\mathcal{P}_K\mathcal{C}_K$. Symbolically, following Figure 4, we are comparing $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ to $\mathcal{C}(\mathbb{X}, f)$. This relationship and the construction of functor $\mathcal{P}_K$ are covered in Section 7.

We then prove our main result, the categorical convergence theorem below.

▶ **Theorem 1** (Convergence between Categorical Reeb Space and Categorical Mapper)**.** *Given a multivariate function $f : \mathbb{X} \to \mathbb{R}^d$ defined on a compact topological space*[4]*, the data is represented as an object $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$. Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be a good cover of $f(\mathbb{X}) \subseteq \mathbb{R}^d$, K be the nerve of the cover and $\text{res}(\mathcal{U})$ be the resolution of the cover, that is, the maximum diameter of the sets in the cover $\text{res}(\mathcal{U}) = \sup\{\text{diam}(U_\alpha) \mid U_\alpha \in \mathcal{U}\}$. Then*

$$d_I(\mathcal{C}(\mathbb{X}, f), \mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)) \leq \text{res}(\mathcal{U}).$$

Theorem 1 states that for increasingly refined covers, the image of the categorical mapper converges to the categorical Reeb space in the interleaving distance. In other words, the distance between the mapper and the Reeb space is bounded above by the resolution of the discretization. Thus, we can make approximation guarantees about the accuracy of the mapper based on a property of the chosen discretization.

Summary. The various categorical representations can be summarized in Figure 4, some of which are illustrated in Figure 5 for the case when $d = 1$. The initial data received is an object $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$. Then we can either construct its categorical Reeb space through the functor $\mathcal{C}$, or construct its categorical mapper using the functor $\mathcal{C}_K$. In order to compare these two objects in the same category, we push the mapper along using the $\mathcal{P}_K$ functor, and then compute the distance between $\mathcal{C}(\mathbb{X}, f)$ and $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$. We should stress before we continue that the diagram of Figure 4 does not commute. In a way, the above distance is measuring how far the diagram is from being commutative. Making no assumptions about $\mathcal{U}$, Theorem 1 states that the interleaving distance between the results of the two paths in the diagram is bounded by the resolution of $\mathcal{U}$. Furthermore in Section 8, for the special case when $d = 1$, we turn our categorical convergence theorem, Theorem 1,

[3] A related but slightly different categorical mapper was introduced by Stovner [23], as a functor from the category of covered topological spaces to the category of simplicial complexes.

[4] For simplicity, we assume a combinatorial s-manifold; however this is not necessary for the proof.

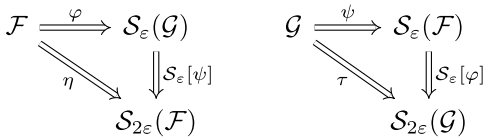
into the geometric convergence theorem, Corollary 6. Finally, we provide an algorithm for producing a geometric representation of the image of categorical mapper, $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$.

5 Interleaving Distance between Reeb Spaces

As described in Section 4, we start by generalizing the categorical Reeb graph to the categorical Reeb space. Given the data received as a topological space $\mathbb{X}$ equipped with an $\mathbb{R}^d$-valued function $f : \mathbb{X} \to \mathbb{R}^d$, denoted as $(\mathbb{X}, f)$, we define the functor $\mathcal{C} : \mathbb{R}^d\text{-}\mathbf{Top} \to \mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ as follows: $\mathcal{C}$ maps an object $(\mathbb{X}, f)$ in $\mathbb{R}^d$-**Top** to a functor $\mathcal{C}(\mathbb{X}, f) := \pi_0 f^{-1} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$, and an arrow $\nu : (\mathbb{X}, f) \to (\mathbb{Y}, g)$ to a natural transformation $\mathcal{C}[\nu]$ induced by the inclusion $\nu f^{-1}(I) \subseteq g^{-1}(I)$. The functor $\mathcal{C}$ turns the given data into the categorical representation of the Reeb space, and the functoriality of π_0 makes it a well-defined functor.

Our first goal is to define the interleaving distance for these categorical Reeb spaces. Denote the ε-thickening of a open set $I \in \mathbf{Open}(\mathbb{R}^d)$ to be the set $I^\varepsilon := \{x \in \mathbb{R}^d \mid \|x - I\| < \varepsilon\}$. Using this, we can define a thickening functor $T_\varepsilon : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Open}(\mathbb{R}^d)$ by $T_\varepsilon(I) := I^\varepsilon$, and $T_\varepsilon[I \subseteq J] := \{I^\varepsilon \subseteq J^\varepsilon\}$. Let $\mathcal{S}_\varepsilon$ be the functor from $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ to itself defined by $\mathcal{S}_\varepsilon(\mathcal{F}) := \mathcal{F}T_\varepsilon$, for every functor $\mathcal{F} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$. Given the two functors $\mathcal{F}$ and $\mathcal{S}_{2\varepsilon}(\mathcal{F})$, both of which are defined on $\mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$, there is an obvious natural transformation $\eta : \mathcal{F} \Rightarrow \mathcal{S}_{2\varepsilon}\mathcal{F}$ defined by $\eta_I = \mathcal{F}[I \subseteq I^{2\varepsilon}]$. We write $\tau : \mathcal{G} \Rightarrow \mathcal{S}_{2\varepsilon}(\mathcal{G})$ for the analogous natural transformation for $\mathcal{G}$.

▶ **Definition 2** (Interleaving distance between Categorical Reeb spaces)**.** An ε-interleaving between functors $\mathcal{F}, \mathcal{G} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ is a pair of natural transformations, $\varphi : \mathcal{F} \Rightarrow \mathcal{S}_\varepsilon(\mathcal{G})$ and $\psi : \mathcal{G} \Rightarrow \mathcal{S}_\varepsilon(\mathcal{F})$ such that the diagrams below commute.



Given two functors $\mathcal{F}, \mathcal{G} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$, the interleaving distance is defined to be

$$d_I(\mathcal{F}, \mathcal{G}) = \inf\{\varepsilon \in \mathbb{R}_{\geq 0} \mid \mathcal{F}, \mathcal{G} \text{ are } \varepsilon\text{-interleaved}\}.$$

We define $d_I(F, G) = \infty$ if the set on the right-hand side is empty.

We prove in the full version [16] the following property of d_I using [1].

▶ **Theorem 3.** *The interleaving distance d_I, between two categorical representations of Reeb spaces, is an extended pseudometric on $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$.*

Special case for Reeb graphs. When $d = 1$ we have much more control of the situation. In particular, [7] gives us that the category of Reeb graphs, defined to be finite graphs with real valued functions that are strictly monotone on the edges, is equivalent to a well-behaved subcategory of $\mathbf{Set}^{\mathbf{Open}(\mathbb{R})}$. Theorem 4 (as a direct consequence of Corollary 4.9 in [7]) says that the above defined interleaving distance d_I is an extended metric, not just a pseudometric, when restricted to these objects.

▶ **Theorem 4** ([7])**.** *When $d = 1$, $d_I(\mathcal{C}(\mathbb{X}, f), \mathcal{C}(\mathbb{Y}, g))$ is an extended metric on the categorical Reeb spaces.*

Theorem 4 means that for $d = 1$, if $d_I(\mathcal{C}(\mathbb{X}, f), \mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)) = 0$ (that is, when the categorical mapper converges to the categorical Reeb graph), then $\mathcal{C}(\mathbb{X}, f)$ and $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ are isomorphic as functors. This implies that, in the special case when $d = 1$, mapper converges to the Reeb graph as spaces, not just in the interleaving distance. While recent work is beginning to elucidate the case where $d > 1$, the technical finesse needed to make a similar statement to Theorem 4 is beyond the scope of this paper. Thus, we will stick to statements about the categorical representations for Reeb spaces when $d > 1$, and make concrete geometric statements when they are available for $d = 1$ (see Section 8).

6 Categorical Representation of Mapper and its Construction

The beauty of working with category theory is that we can store a categorical representation of the mapper as sets over the nerve of a cover, rather than working directly with its complicated topological definition (given in Section 2). Given a choice of finite open cover for image$(f) \subseteq \mathbb{R}^d$, $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$, let $K = \mathrm{Nrv}(\mathcal{U})$. In order to ensure that K faithfully represents the underlying structure, we will assume that $\mathcal{U}$ is a *good* cover. This ensures that the nerve lemma applies; that is, K has the homotopy type of image$(f) \subseteq \mathbb{R}^d$ (see, e.g., Corollary 4G.3 [12] or Theorem 15.21 [13]).

For simplicity of notation, we denote $\mathcal{U}_\sigma = \bigcap_{\alpha \in \sigma} U_\alpha$ to be the open set in $\mathbb{R}^d$ associated to the simplex $\sigma \in K$. One important property of this construction is that for $\sigma \leq \tau$ in K, the associated inclusion of spaces is reversed: $\mathcal{U}_\sigma \supseteq \mathcal{U}_\tau$. So, if we wish to represent the connected components for a particular $\mathcal{U}_\sigma$ for $\sigma \in K$, we can still consider $\pi_0 f^{-1}(\mathcal{U}_\sigma)$, however, the face relation $\sigma \leq \tau$ induces a "backwards" mapping $\pi_0 f^{-1}(\mathcal{U}_\tau) \to \pi_0 f^{-1}(\mathcal{U}_\sigma)$. We keep track of this switch using the opposite category. Recall $\mathbf{Cell}(K)$ is a category with simplices of K as objects and a unique arrow $\sigma \to \tau$ given by the face relation $\sigma \leq \tau$. Then the opposite category, $\mathbf{Cell}(K)^{\mathrm{op}}$, has the simplices of K as objects and a unique arrow $\tau \to \sigma$ given by the face relation $\sigma \leq \tau$.

Thus, given an object $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$, we have a functor $\mathcal{C}_K^f : \mathbf{Cell}(K)^{\mathrm{op}} \to \mathbf{Set}$ that maps every σ to $\mathcal{C}_K^f(\sigma) := \pi_0 f^{-1}(\mathcal{U}_\sigma)$. We are required to use the opposite cell category so that $\mathcal{C}_K^f$ maps the morphism $\sigma \leq \tau$ (equivalently notated $\tau \to \sigma$ in the opposite category) to the set map $\pi_0 f^{-1}(\mathcal{U}_\tau) \to \pi_0 f^{-1}(\mathcal{U}_\sigma)$ induced by the inclusion $\mathcal{U}_\tau \subseteq \mathcal{U}_\sigma$ as discussed above. This functor is used to represent the categorical mapper of $(\mathbb{X}, f)$ for the cover $\mathcal{U}$.

Note that the functor $\mathcal{C}_K^f$ is an object of the category of functors $\mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}}$. The process of building the mapper is thus represented itself by the functor $\mathcal{C}_K : \mathbb{R}^d$-$\mathbf{Top} \to \mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}}$, which is defined as follows. For the objects, $\mathcal{C}_K$ maps an $\mathbb{R}^d$-space $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$ to the functor $\mathcal{C}_K(\mathbb{X}, f) := \mathcal{C}_K^f$ as given above. For the morphisms, it sends a function preserving map $\nu : (\mathbb{X}, f) \to (\mathbb{Y}, g)$ to a natural transformation (which is an arrow in $\mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}}$), $\mathcal{C}_K[\nu] : \mathcal{C}_K^f \to \mathcal{C}_K^g$. Technical details in checking that $\mathcal{C}_K[\nu]$ is indeed a natural transformation are deferred to the full version [16].

7 Convergence between Mapper and Reeb Space

In order to compare the discrete mapper with the continuous Reeb space, we must move them both into the same category. At the moment, for data given as $(\mathbb{X}, f)$ in $\mathbb{R}^d$-$\mathbf{Top}$, we have the categorical Reeb space representation $\mathcal{C}(\mathbb{X}, f)$ in $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$, and the categorical mapper representation $\mathcal{C}_K(\mathbb{X}, f)$ in $\mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}}$. Thus we must first define the functor $\mathcal{P}_K$ in order to push the mapper representation into the $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ category, then prove the convergence result there using the interleaving distance from Section 5. Here, we will give

the definition of $\mathcal{P}_K$ coming from the categorical setting, and then give an equivalent functor $\mathcal{F}$ in Lemma 5 which is more intuitive to work with.

Given an abstract simplicial complex K which is the nerve of the cover $\mathcal{U}$, we define K_A for a open set $A \subseteq \mathbb{R}^d$ to be the collection of simplices in K such that the associated intersection $\mathcal{U}_\sigma$ intersects A, $K_A = \{\sigma \in K \mid \mathcal{U}_\sigma \cap A \neq \emptyset\}$ (see the full version [16] for an example when $d = 2$). Now we can construct the functor $\mathcal{P}_K : \mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}} \to \mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ as follows. Given a functor $F : \mathbf{Cell}(K)^{\mathrm{op}} \to \mathbf{Set}$, $\mathcal{P}_K$ sends it to a functor $\mathcal{P}_K(F) : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ by defining

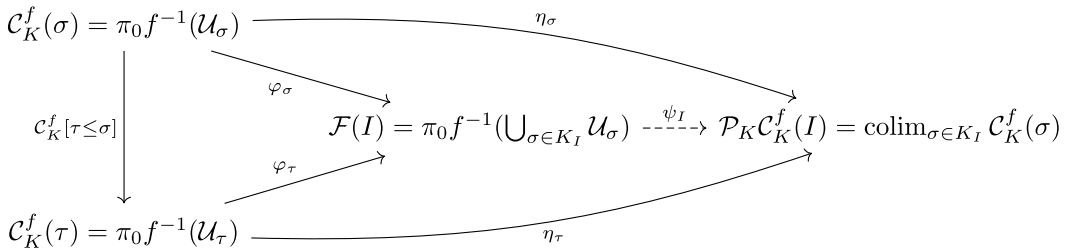
$$\mathcal{P}_K(F)(I) = \operatorname*{colim}_{\sigma \in K_I} F(\sigma)$$

for every I in $\mathbf{Open}(\mathbb{R}^d)$. Here, the colimit construction can be thought of as a set representing the connected components over the collection of open sets $\mathcal{U}_\sigma$ for the simplices $\sigma \in K_I$, or equivalently, over the union $\bigcup_{\sigma \in K_I} \mathcal{U}_\sigma$. The morphisms in the two functor categories $\mathbf{Set}^{\mathbf{Cell}(K)^{\mathrm{op}}}$ and $\mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ are natural transformations; $\mathcal{P}_K$ sends arrows to arrows in a well-defined way via the colimit as discussed at the end of Section 3, since if $I \subseteq J$, then $K_I \subseteq K_J$. Additionally, we must check that $\mathcal{P}_K$ sends a natural transformation $\eta : F \Rightarrow G$ to a natural transformation $\mathcal{P}_K(F) \to \mathcal{P}_K(G)$; we omit this bookkeeping here. Since mapper depends on the choice of a cover, it makes sense that the cover and, in particular, its resolution will be a key factor in understanding the convergence. With all of this machinery, we have our main result, Theorem 1.

Theorem 1 implies that if we have a sequence of covers $\mathcal{U}_i$ such that $\mathrm{res}(\mathcal{U}_i) \to 0$, then the categorical representations of the associated mappers converge to the Reeb space in the interleaving distance. Its proof relies on a main technical result, Lemma 5 below, which relates the functor $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ to one which avoids the combinatorial structure of K as much as possible and instead works with inverse images of subsets of $\mathbb{R}^d$.

▶ **Lemma 5.** *Let $\mathcal{F} : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ be a functor which maps an open set I to a set $\pi_0 f^{-1}(\bigcup_{\sigma \in K_I} \mathcal{U}_\sigma)$ with morphisms induced by π_0 on the inclusions. Then, the functor $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ is equivalent to $\mathcal{F}$.*

Proof. The functor $\mathcal{C}_K(\mathbb{X}, f) = \mathcal{C}_K^f : \mathbf{Cell}(K)^{\mathrm{op}} \to \mathbf{Set}$ is given by sending a cell σ to $\pi_0 f^{-1}(\mathcal{U}_\sigma)$, and its composition with $\mathcal{P}_K$ is given by $\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f) = \mathcal{P}_K(\mathcal{C}_K^f) : \mathbf{Open}(\mathbb{R}^d) \to \mathbf{Set}$ defined by $\mathcal{P}_K(\mathcal{C}_K^f)(I) = \mathrm{colim}_{\sigma \in K_I} \mathcal{C}_K^f(\sigma)$. To establish a natural equivalence of functors, we will construct a natural transformation $\psi : \mathcal{F} \Rightarrow \mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ which is an isomorphism for each ψ_I. As a roadmap, we can refer to the following diagram:



By definition of $\mathcal{F}$, $\mathcal{F}(I) = \pi_0 f^{-1}(\bigcup_{\sigma \in K_I} \mathcal{U}_\sigma)$ so there are obvious maps induced by inclusions $\varphi_\sigma : \pi_0 f^{-1}(\mathcal{U}_\sigma) \to \mathcal{F}(I)$ which all commute; this gives us a cone $(\mathcal{F}(I), \varphi_\sigma)$ for the diagram $\{\mathcal{C}_K^f(\sigma)\}_{\sigma \in K_I}$. The colimit of this same diagram is a cocone denoted by $(\mathcal{P}_K\mathcal{C}_K^f(I), \eta_\sigma)$. We will construct a map $\psi_I : \pi_0 f^{-1}(\bigcup_{\sigma \in K_I} \mathcal{U}_\sigma) \to \mathrm{colim}_{\sigma \in K_I} \mathcal{C}_K^f(\sigma)$ such that the colimit cocone factors through the cocone $(\mathcal{F}(I), \varphi_\sigma)$ using ψ_I; that is, $\psi_I \circ \varphi_\sigma = \eta_\sigma$ for all $\sigma \in K_I$. The universality of the colimit then implies that ψ_I is an isomorphism.

$$\begin{array}{ccc} \pi_0 f^{-1}(\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma) & \xrightarrow{\psi_I} & \mathcal{P}_K \mathcal{C}_K^f(I) \\ \downarrow & & \downarrow \\ \pi_0 f^{-1}(\bigcup_{\sigma\in K_J} \mathcal{U}_\sigma) & \xrightarrow{\psi_J} & \mathcal{P}_K \mathcal{C}_K^f(J) \end{array}$$

Figure 6 The diagram showing that $\varphi = \{\varphi_I\}$ defines a natural transformation.

$$\begin{array}{ccc} \mathcal{F}(I) & \xrightarrow{\varphi_I} & \mathcal{C}(I^\varepsilon) \\ \downarrow{\scriptstyle \mathcal{F}[I\subseteq J]} & & \downarrow{\scriptstyle \mathcal{C}[I^\varepsilon \subseteq J^\varepsilon]} \\ \mathcal{F}(J) & \xrightarrow{\varphi_J} & \mathcal{C}(J^\varepsilon) \end{array} \qquad \begin{array}{ccc} \mathcal{C}(I) & \xrightarrow{\psi_I} & \mathcal{F}(I^\varepsilon) \\ \downarrow{\scriptstyle \mathcal{C}[I\subseteq J]} & & \downarrow{\scriptstyle \mathcal{F}[I^\varepsilon \subseteq J^\varepsilon]} \\ \mathcal{C}(J) & \xrightarrow{\psi_J} & \mathcal{F}(J^\varepsilon) \end{array}$$

$$\begin{array}{ccc} \mathcal{F}(I) & \xrightarrow{\varphi_I} & \mathcal{C}(I^\varepsilon) \\ & {\scriptstyle \mathcal{F}[I\subseteq I^{2\varepsilon}]}\searrow & \downarrow{\scriptstyle \psi_{I^\varepsilon}} \\ & & \mathcal{F}(I^{2\varepsilon}) \end{array} \qquad \begin{array}{ccc} \mathcal{C}(I) & \xrightarrow{\psi_I} & \mathcal{F}(I^\varepsilon) \\ & {\scriptstyle \mathcal{C}[I\subseteq I^{2\varepsilon}]}\searrow & \downarrow{\scriptstyle \varphi_{I^\varepsilon}} \\ & & \mathcal{C}(I^{2\varepsilon}) \end{array}$$

Figure 7 Communicative diagrams showing φ and ψ being natural transformations and ε-interleaved.

To construct ψ_I, consider any u in $\pi_0 f^{-1}(\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma)$. This set element represents a connected component in $f^{-1}(\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma)$, and thus there is at least one σ with an element $v \in \mathcal{C}_K^f(\sigma)$ such that $\varphi_\sigma(v) = u$. Now we define $\psi_I(u) = \eta_\sigma(v)$. Ensuring that ψ_I above is well defined corresponds to ensuring that if there are $v \in \mathcal{C}_K^f(\sigma)$ and $v' \in \mathcal{C}_K^f(\sigma')$ with $\varphi_\sigma(v) = \varphi_\sigma(v') = u$, then $\eta_\sigma(v) = \eta_{\sigma'}(v')$. Note that v and v' represent (path) connected components in $f^{-1}(\mathcal{U}_\sigma)$ and $f^{-1}(\mathcal{U}_{\sigma'})$ respectively. Let x and x' be points in these respective connected components. Since these points are in the same connected component of $f^{-1}(\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma)$, there is a path connecting them, and thus a finite sequence of $\tau_i \in K_I$ with $\tau_0 = \sigma$ and $\tau_n = \sigma'$, such that $f^{-1}(\mathcal{U}_{\tau_i})$ covers the path. We can additionally assume that the τ_i give the maximal simplex containing the path at each location, so that $\tau_i \leq \tau_{i+1}$ or $\tau_i \geq \tau_{i+1}$ for each i. Let $v_i \in \pi_0 f^{-1}\mathcal{U}_{\tau_i}$ represent the connected component of the path. Then we must have $\mathcal{C}_K^f[\tau_i \leq \tau_{i+1}](v_{i+1}) = v_i$ or $\mathcal{C}_K^f[\tau_{i+1} \leq \tau_i](v_i) = v_{i+1}$ for each i. By the colimit properties, this implies that $\eta_{\tau_i}(v_i) = \eta_{\tau_j}(v_j)$ for all i and j, and thus that $\eta_\sigma(v) = \eta_{\sigma'}(v')$ as desired.

Finally, we prove that the collection $\{\psi_I\}$ defines a natural transformation. Since if $I \subseteq J$, then $K_I \subseteq K_J$. Then an exercise in colimit properties ensures that the diagram in Figure 6 commutes, where the arrow on the left is the map induced by inclusions, and the map on the right is induced by the colimit definition. ◀

Proof of Theorem 1. Let $\varepsilon = \mathrm{res}(\mathcal{U})$. Combined with Lemma 5, we will construct, $\varphi : \mathcal{F} \Rightarrow \mathcal{C}(\mathbb{X}, f) \circ T_\varepsilon$ and $\psi : \mathcal{C}(\mathbb{X}, f) \Rightarrow \mathcal{F} \circ T_\varepsilon$, and show that they constitute an ε-interleaving by showing the diagrams of Figure 7 commute following Definition 2.

First, we prove the following statement: if $\mathcal{U}_\sigma \cap I \neq \emptyset$, then $\mathcal{U}_\sigma \subset I^\varepsilon$. Indeed, for any $x \in \mathcal{U}_\sigma$, if $x \in I$ then $x \in I^\varepsilon$. If $x \notin I$, then because there exists a $y \in \mathcal{U}_\sigma \cap I$, such that $\|x - y\| \leq \mathrm{diam}(\mathcal{U}_\sigma) \leq \mathrm{res}(\mathcal{U}) = \varepsilon$, so $x \in I^\varepsilon$. This statement implies that we have the inclusion $\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma \hookrightarrow I^\varepsilon$. We define $\varphi_I : \pi_0 f^{-1}\left(\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma\right) \to \pi_0 f^{-1}(I^\varepsilon)$.

We also have inclusions $I \cap f(\mathbb{X}) \hookrightarrow \bigcup_{\sigma\in K_I} \mathcal{U}_\sigma \hookrightarrow \bigcup_{\sigma\in K_{I^\varepsilon}} \mathcal{U}_\sigma$, since any point $x \in I \cap f(\mathbb{X})$ is contained in some $\mathcal{U}_\alpha$, for some vertex $\alpha \in K_I \subseteq K_{I^\varepsilon}$. Additionally, since $f^{-1}(I) = f^{-1}(I \cap f(\mathbb{X}))$, we define $\psi_I : \pi_0 f^{-1}(I) \to \pi_0 f^{-1}\left(\bigcup_{\sigma\in K_{I^\varepsilon}} \mathcal{U}_\sigma\right)$ to be the composition of the ismorphism $\pi_0 f^{-1}(I) \cong \pi_0 f^{-1}(I \cap f(\mathbb{X}))$ and the map induced by the inclusion $I \cap f(\mathbb{X}) \hookrightarrow \bigcup_{\sigma\in K_{I^\varepsilon}} \mathcal{U}_\sigma$.

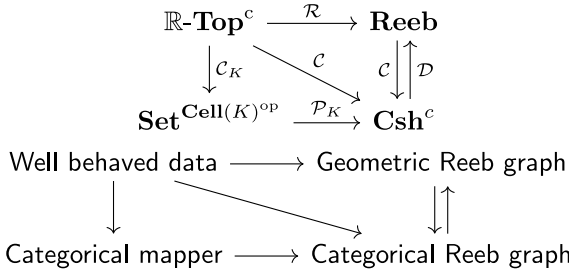


Figure 8 The diagram for connecting geometric representations of the Reeb graph and the mapper.

The top left square of Figure 7 comes from applying the functor $\pi_0 f^{-1}$ to the inclusions $\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma \subseteq I^\varepsilon \subseteq J^\varepsilon$ and $\bigcup_{\sigma\in K_I} \mathcal{U}_\sigma \subseteq \bigcup_{\sigma\in K_J} \mathcal{U}_\sigma \subseteq J^\varepsilon$ for any $I \subseteq J$. Applying $\pi_0 f^{-1}$ to the inclusions $I \cap f(\mathbb{X}) \subseteq \bigcup_{\sigma\in K_{I^\varepsilon}} \mathcal{U}_\sigma \subseteq \bigcup_{\sigma\in K_{J^\varepsilon}} \mathcal{U}_\sigma$ and $I \cap f(\mathbb{X}) \subseteq J \cap f(\mathbb{X}) \subseteq \bigcup_{\sigma\in K_{J^\varepsilon}} \mathcal{U}_\sigma$ for $I \subseteq J$, then replacing $\pi_0 f^{-1}(I \cap f(\mathbb{X}))$ and $\pi_0 f^{-1}(J \cap f(\mathbb{X}))$ with the isomorphic $\mathcal{C}(I)$ and $\mathcal{C}(J)$ respectively gives the diagram of the top right. A similar argument implies that the diagrams in Figure 7 bottom also commute, hence φ and ψ are an ε-interleaving. ◀

8 Geometric Representations

We now leverage the results of [7] to make geometric statements connecting the mapper and the Reeb space for $d = 1$. The main idea is to define a mapping that recovers the geometric representation of the mapper from its categorical representation, and to establish convergence between the mapper and the Reeb graph geometrically. Such a mapping relies on well behaved data, made precise by the notion of constructibility.

Review of prior results. We will follow the notations of [7] which occasionally can be technical. The categories and functors we will discuss can be summed up in the roadmap of Figure 8. Notice its lower left triangle resembles that of Figure 4 with further restrictions. Recall the notation from Section 4; when $d = 1$, the category $\mathbb{R}$-**Top** is exactly the category $\mathbb{R}^d$-**Top**: an object of $\mathbb{R}$-**Top** is an $\mathbb{R}$-space (a pair of a topological space $\mathbb{X}$ and a continuous map $f : \mathbb{X} \to \mathbb{R}$), and an arrow in $\mathbb{R}$-**Top** is a function-preserving map.

Since the geometric Reeb graph of a general $\mathbb{R}$-space may be badly behaved, we restrict to special classes of spaces [7], that is, we focus on well behaved subcategories. In particular, we define the full subcategory $\mathbb{R}$-**Top**^{c} of $\mathbb{R}$-**Top** where the objects are *constructible* $\mathbb{R}$-spaces (see Section 2.2 and Figure 5 of [7] for illustrations and technical details). This collection includes, e.g., PL functions on triangulations of manifolds and Morse functions. Then we define the full subcategory **Reeb** of $\mathbb{R}$-**Top**^{c} (in the finite, discrete setting), which is exactly the category of Reeb graphs, viewed as a graph with a real valued function which is monotone on edges, with arrows given by function preserving maps. Subsequently, the construction of a (geometric) Reeb graph from well behaved data (a constructible $\mathbb{R}$-space) is captured by the functor $\mathcal{R} : \mathbb{R}$-**Top**$^c \to$ **Reeb**.

We can similarly restrict our objects of interest in **Set**$^{\mathbf{Open}(\mathbb{R})}$ to be well behaved. A *cosheaf* is a functor $F :$ **Open**$(\mathbb{R}) \to$ **Set** such that for any open cover $\mathcal{U}$ of a set U, the unique map $\mathrm{colim}_{U_\alpha\in\mathcal{U}} F(U_\alpha) \to F(U)$ is an isomorphism. We further restrict the cosheaves to constructible cosheaves; a cosheaf is *constructible* if there is a finite set $S \subset \mathbb{R}$ such that if $A, B \in$ **Open**$(\mathbb{R})$ with $A \subseteq B$ and $S \cap A = S \cap B$, then $F(A) \to F(B)$ is an isomorphism.

In addition, we require that if $A \cap S = \emptyset$ then $F(A) = \emptyset$. The category of constructible cosheaves with natural transformations is denoted $\mathbf{Csh}^c$.

The work of [7] gives the equivalence of categories $\mathbf{Reeb} \equiv \mathbf{Csh}^c$. In Figure 8, when $d = 1$, the functor $\mathcal{C} : \mathbb{R}^d\text{-}\mathbf{Top} \to \mathbf{Set}^{\mathbf{Open}(\mathbb{R}^d)}$ (given in Figure 4) restricts to a functor $\mathcal{C} : \mathbb{R}\text{-}\mathbf{Top}^c \to \mathbf{Csh}^c$. Its further restriction $\mathcal{C} : \mathbf{Reeb} \to \mathbf{Csh}^c$ is exactly the functor used in [7] to give the equivalence of categories. In addition, $\mathcal{C}$ has an "inverse" functor $\mathcal{D} : \mathbf{Csh}^c \to \mathbf{Reeb}$ which can turn a constructible cosheaf back into a geometric object through the display locale construction [24]. This construction also satisfies the equality $\mathcal{R} = \mathcal{D}\mathcal{C}$ due to the commutativity of the upper right triangle in Figure 8 (as proved in Section 3.5 of [7]). Therefore constructing the (geometric) Reeb graph from well behaved data is the same as creating its categorical representation, and then turning it back into a geometric object.

Our result. The above result implies that because we can turn *any* constructible cosheaf back into a geometric Reeb graph, we can now turn the mapper, defined previously as a categorical object, back into a geometric object. In this spirit, let $\mathcal{M}_K(\mathbb{X}, f) := \mathcal{D}\mathcal{P}_K\mathcal{C}_K(\mathbb{X}, f)$ be the geometric representation of the mapper object, referred to as the *geometric mapper* (following the rectangular diagram in Figure 8), and let $\mathcal{R}(\mathbb{X}, f)$ be the geometric Reeb graph. Then, the equivalence of categories gives us the following immediate corollary to Theorem 1.

▶ **Corollary 6.** *Given a constructible $\mathbb{R}$-space $(\mathbb{X}, f)$ with $f : \mathbb{X} \to \mathbb{R}$, let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be a good cover of $f(\mathbb{X}) \subseteq \mathbb{R}$, and let K be the nerve of the cover. Then*

$$d_I(\mathcal{R}(\mathbb{X}, f), \mathcal{M}_K(\mathbb{X}, f)) \leq \mathrm{res}(\mathcal{U}).$$

In particular, because the interleaving distance is an extended metric when $d = 1$, this implies that a sequence of mappers for more refined covers $\mathcal{U}$ converges to the Reeb graph geometrically. Recent work has also investigated this convergence problem using the bottleneck distance for the extended persistence diagrams [4]; instead, we use the interleaving distance.

Algorithm for geometric mapper. Constructing the geometric representation of 1-dimensional mapper from its categorical representation follows a simple algorithm (as illustrated in Figure 5). For the purpose of exposition, we assume that the mapper is constructed with a finite, connected, minimal cover (a cover with no subcover) and that the number of connected components over each cover element is finite. We further assume that the open sets (intervals) in $\mathcal{U} = \{U_i = (a_i, b_i)\}_{i=1}^n$ can be ordered and satisfy $a_1 < a_2 < b_1 < a_3 < b_2 < \cdots < a_{n-1} < b_{n-1} < b_n$. For ease of notation, we assume there are extra intervals $U_0 = (a_0, b_0)$ with $a_0 < a_1 < b_0 < b_1$ and $U_{n+1} = (a_{n+1}, b_{n+1})$ with $b_{n-1} < a_{n+1} < b_n < b_{n+1}$ and such that $f^{-1}(U_0) = f^{-1}(U_{n+1}) = \emptyset$. Let $M := M(\mathcal{U}, f)$ be the mapper with the added property that for any cover element U_i, we store the vertices corresponding to connected components of $f^{-1}(U_i)$ in the set $F(i)$. Furthermore, let $M[i]$ be the subgraph of M induced by the collection of vertices $F(i)$, and let $M[i, i+1]$ be the subgraph of M induced by the vertices $F(i) \cup F(i+1)$. Note that for any small enough interval $I \subset (a_{i+1}, b_i)$, the colimit construction for I gives exactly the connected components over the union $U_i \cup U_{i+1}$, which is equivalently represented by the connected components of $M[i, i+1]$. For any small enough interval $I \subset (b_{i-1}, a_{i+1})$, the colimit construction for I gives the connected components over U_i, and thus is represented by the connected components of $M[i]$, which are just the vertices.

Thus, the geometric mapper, $\mathcal{M}_K(\mathbb{X}, f) = (\mathbb{X}', f')$, a graph $\mathbb{X}'$ equipped with a function f', can be constructed based on a combinatorial structure described below. For each interval

$[b_{i-1}, a_{i+1}]$, add an edge uv with two new pink vertices for each vertex in $M[i]$ (see Figure 5 Algorithm). Set $f'(u) = b_{i-1}$ and set $f'(v) = a_{i+1}$. For each interval $[a_{i+1}, b_i]$, add an edge wx with two new yellow vertices for each connected component in $M[i, i+1]$. Set $f'(w) = a_{i+1}$ and $f'(x) = b_i$. Now, we have a combinatorial structure which consists of a collection of disjoint edges spread across each of the intervals defined by the cover, and each edge has a top vertex and a bottom vertex given by the function values. A pink and a yellow vertex are called equivalent if the vertex sets corresponding to them in $M[i]$ and $M[i, i+1]$ respectively have a nontrivial intersection. The graph $\mathbb{X}'$ resulting from identifying (i.e. gluing) equivalent vertices with the same function value of f' is the geometric mapper. Such an algorithm relies on subroutines of union-find, therefore it inherits the complexity of union-find that varies depending on naive or advanced implementations.

9 Discussion

The authors of [4] asked whether it is possible to describe the mapper as a particular constructible cosheaf. We addressed this question for $d = 1$ in Section 8: we described the mapper as a constructible cosheaf when it is passed to the continuous version. We suspect that our geometric results hold in the case $d > 1$. That is, with the proper notion of constructibility for $\mathbb{R}^d$-spaces and cosheaves, we will have both an equivalence of categories, and a proof that the interleaving distance is an extended metric, not just a pseudometric; and therefore the mapper converges to the Reeb space on the space level. Our results are first steps towards providing a theoretical justification for the use of discrete objects (mapper and JCN) as approximations to the Reeb space with guarantees. Some future directions include creating categorical interpretation of multiscale mapper [8] and studying distance metrics between Jacobi sets in the categorical setting.

Acknowledgements. BW would like to thank the support from NSF IIS-1513616.

References

1 Peter Bubenik, Vin de Silva, and Jonathan Scott. Metrics for generalized persistence modules. *Foundations of Computational Mathematics*, 15(6):1501–1531, 2015.

2 Hamish Carr and David Duke. Joint contour nets. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1100–1113, 2014.

3 Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry*, 43:42–58, 2010.

4 Mathieu Carriére and Steve Oudot. Structure and stability of the 1-dimensional mapper. *Symposium on Computational Geometry (to appear); arXiv:1511.05823*, 2016.

5 Frédéric Chazal and Jian Sun. Gromov-Hausdorff approximation of filament structure using Reeb-type graph. *Proceedings 13th Annual Symposium on Computational Geometry*, pages 491–500, 2014.

6 Justin Curry. *Sheaves, Cosheaves and Applications*. PhD thesis, University of Pennsylvania, 2014.

7 Vin de Silva, Elizabeth Munch, and Amit Patel. Categorification of Reeb graphs. *Discrete and Computational Geometry (to appear); arXiv:1501.04147*, 2016.

8 Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Mutiscale mapper: A framework for topological summarization of data and maps. arXiv:1504.03763, 2015.

9 Herbert Edelsbrunner and John Harer. Jacobi sets of multiple Morse functions. In F. Cucker, R. DeVore, P. Olver, and E. Süli, editors, *Foundations of Computational Mathematics, Minneapolis 2002*, pages 37–57. Cambridge University Press, 2002.

10 Herbert Edelsbrunner, John Harer, and Amit K. Patel. Reeb spaces of piecewise linear mappings. *Proceedings 24th Annual Symposium on Computational Geometry*, pages 242–250, 2008.

11 William Harvey, Yusu Wang, and Rephael Wenger. A randomized $O(m \log m)$ algorithm for computing Reeb graphs of arbitrary simplicial complexes. *ACM Symposium on Computational Geometry*, pages 267–276, 2010.

12 Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

13 Dimitry Kozlov. *Combinatorial Algebraic Topology*. Springer, 2008.

14 P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Vejdemo-Johansson, M. Alagappan, J. Carlsson, and G. Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3, 2013.

15 Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, New York, NY, 2nd edition, 1978.

16 Elizabeth Munch and Bei Wang. Convergence between categorical representations of Reeb space and mapper. arXiv:1307.7760, 2016.

17 Monica Nicolaua, Arnold J. Levineb, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings National Academy of Sciences of the United States of America*, 108(17):7265–7270, 2011.

18 Salman Parsa. A deterministic $O(m \log m)$ time algorithm for the Reeb graph. *Proceedings 29th Annual Symposium on Computational Geometry*, pages 269–276, 2012.

19 Amit Patel. *Reeb Spaces and the Robustness of Preimages*. PhD thesis, Duke University, 2010.

20 Michael A. Penna. On the geometry of combinatorial manifolds. *Pacific Journal of Mathematics*, 77(2):499–522, 1978.

21 G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [on the singular points of a complete integral pfaff form or of a numerical function]. *Comptes Rendus Acad.Science Paris*, 222:847–849, 1946.

22 Gurjeet Singh, Facundo Mémoli, and Gunnar Carlsson. Topological methods for the analysis of high dimensional data sets and 3D object recognition. In *Eurographics Symposium on Point-Based Graphics*, 2007.

23 Roar Bakken Stovner. On the mapper algorithm: A study of a new topological method for data analysis. Master's thesis, Norwegian University of Science and Technology, 2012.

24 Jon Woolf. The fundamental category of a stratified space. arXiv:0811.2580, 2013.

New Lower Bounds for ϵ-Nets

Andrey Kupavskii[*1], **Nabil H. Mustafa**[†2], **and János Pach**[‡3]

1 **Moscow Institute of Physics and Technology, Moscow, Russia; and École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland**
kupavskii@yandex.ru

2 **Université Paris-Est, LIGM, Equipe A3SI, ESIEE Paris, France**
mustafan@esiee.fr

3 **Rényi Institute, Budapest, Hungary; and École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland**
pach@cims.nyu.edu

Abstract

Following groundbreaking work by Haussler and Welzl (1987), the use of small ϵ-nets has become a standard technique for solving algorithmic and extremal problems in geometry and learning theory. Two significant recent developments are: (i) an upper bound on the size of the smallest ϵ-nets for set systems, as a function of their so-called shallow-cell complexity (Chan, Grant, Könemann, and Sharpe); and (ii) the construction of a set system whose members can be obtained by intersecting a point set in $\mathbb{R}^4$ by a family of half-spaces such that the size of any ϵ-net for them is $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ (Pach and Tardos).

The present paper completes both of these avenues of research. We (i) give a lower bound, matching the result of Chan *et al.*, and (ii) generalize the construction of Pach and Tardos to half-spaces in $\mathbb{R}^d$, for any $d \geq 4$, to show that the general upper bound, $O(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$, of Haussler and Welzl for the size of the smallest ϵ-nets is tight.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases ϵ-nets, lower bounds, geometric set systems, shallow-cell complexity, half-spaces.

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.54

1 Introduction

Let X be a finite set and let $\mathcal{R}$ be a system of subsets of an underlying set containing X. In computational geometry, the pair $(X, \mathcal{R})$ is usually called a *range space*. A subset $X' \subseteq X$ is called an *ϵ-net* for $(X, \mathcal{R})$ if $X' \cap R \neq \emptyset$ for every $R \in \mathcal{R}$ with at least $\epsilon|X|$ elements. The use of small-sized ϵ-nets in geometrically defined range spaces has become a standard technique in discrete and computational geometry, with many combinatorial and algorithmic consequences. In most applications, ϵ-nets precisely and provably capture the most important quantitative and qualitative properties that one would expect from a random sample. Typical applications include the existence of spanning trees and simplicial partitions

* The work of Andrey Kupavskii has been supported in part by the Swiss National Science Foundation Grants 200021-137574 and 200020-14453 and by the grant N 15-01-03530 of the Russian Foundation for Basic Research.

† The work of Nabil H. Mustafa in this paper has been supported by the grant ANR SAGA (JCJC-14-CE25-0016-01).

‡ The work of János Pach has been partially supported by Swiss National Science Foundation Grants 200020-144531 and 200020-162884.

© Andrey Kupavskii, Nabil H. Mustafa, and János Pach;
licensed under Creative Commons License CC-BY

with low crossing number, upper bounds for discrepancy of set systems, LP rounding, range searching, streaming algorithms; see [13, 18].

For any subset $Y \subseteq X$, define the *projection* of $\mathcal{R}$ on Y to be the set system

$$\mathcal{R}|_Y := \big\{Y \cap R : R \in \mathcal{R}\big\}.$$

The *Vapnik-Chervonenkis dimension* or, in short, the *VC-dimension* of the range space $(X, \mathcal{R})$ is the minimum integer d such that $|\mathcal{R}|_Y| < 2^{|R|}$ for any subset $Y \subseteq X$ with $|Y| > d$. According to the Sauer–Shelah lemma [21, 23] (discovered earlier by Vapnik and Chervonenkis [24]), for any range space $(X, \mathcal{R})$ whose VC-dimension is at most d and for any subset $Y \subseteq X$, we have $|\mathcal{R}|_Y| = O(|Y|^d)$.

A straightforward sampling argument shows that every range space $(X, \mathcal{R})$ has an ϵ-net of size $O(\frac{1}{\epsilon} \log |\mathcal{R}|_X|)$. The remarkable result of Haussler and Welzl [10], based on the previous work of Vapnik and Chervonenkis [24], shows that much smaller ϵ-nets exist if we assume that our range space has small VC-dimension. Haussler and Welzl [10] showed that if the VC-dimension of a range space $(X, \mathcal{R})$ is at most d, then by picking a random sample of size $\Theta(\frac{d}{\epsilon} \log \frac{d}{\epsilon})$, we obtain an ϵ-net with positive probability. Actually, they only used the weaker assumption that $|\mathcal{R}|_Y| = O(|Y|^d)$ for every $Y \subseteq X$. This bound was later improved to $(1 + o(1))(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$, as $d, \frac{1}{\epsilon} \to \infty$ [11]. In the sequel, we will refer to this result as the *ϵ-net theorem*. The key feature of the ϵ-net theorem is that it guarantees the existence of an ϵ-net whose size is *independent* of both $|X|$ and $|\mathcal{R}|_X|$. Furthermore, if one only requires the VC-dimension of $(X, \mathcal{R})$ to be bounded by d, then this bound cannot be improved. It was shown in [11] that given any $\epsilon > 0$ and integer $d \geq 2$, there exist range spaces with VC-dimension at most d, and for which any ϵ-net must have size at least $\big(1 - \frac{2}{d} + \frac{1}{d(d+2)} + o(1)\big)\frac{d}{\epsilon} \log \frac{1}{\epsilon}$.

The effectiveness of ϵ-net theory in geometry derives from the fact that most "geometrically defined" range spaces $(X, \mathcal{R})$ arising in applications have bounded VC-dimension and, hence, satisfy the preconditions of the ϵ-net theorem.

There are two important types of geometric set systems, both involving points and geometric objects in $\mathbb{R}^d$, that are used in such applications. Let $\mathcal{R}$ be a family of possibly unbounded geometric objects in $\mathbb{R}^d$, such as the family of all half-spaces, all balls, all polytopes with a bounded number of facets, or all *semialgebraic sets* of bounded complexity, i.e., subsets of $\mathbb{R}^d$ defined by at most D polynomial equations or inequalities in the d variables, each of degree at most D. Given a finite set of points $X \subset \mathbb{R}^d$, we define the *primal range space* $(X, \mathcal{R})$ as the set system "induced by containment" in the objects from $\mathcal{R}$. Formally, it is a set system with the set of elements X and sets $\{X \cap R : R \in \mathcal{R}\}$. The combinatorial properties of this range space depend on the projection $\mathcal{R}|_X$. Using this terminology, Radon's theorem [13] implies that the primal range space on a ground set X, induced by containment in half-spaces in $\mathbb{R}^d$, has VC-dimension at most $d + 1$ [18]. Thus, by the ϵ-net theorem, this range space has an ϵ-net of size $O(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$.

In many applications, it is natural to consider the dual range space, in which the roles of the points and ranges are swapped. As above, let $\mathcal{R}$ be a family of geometric objects (ranges) in $\mathbb{R}^d$. Given a finite set of objects $\mathcal{S} \subseteq \mathcal{R}$, the *dual range space* "induced" by them is defined as the set system (hypergraph) on the ground set $\mathcal{S}$, consisting of the sets $S_x := \{S \in \mathcal{S} : x \in S\}$ for all $x \in \mathbb{R}^d$. It can be shown that if for any $X \subset \mathbb{R}^d$ the VC-dimension of the range space $(X, \mathcal{R})$ is less than d, then the VC-dimension of the dual range space induced by any subset of $\mathcal{R}$ is less than 2^d [13].

Recent progress

In many geometric scenarios, however, one can find smaller ϵ-nets than those whose existence is guaranteed by the ϵ-net theorem. It has been known for a long time that this is the case, e.g.,

for primal set systems induced by containment in balls in $\mathbb{R}^2$ and half-spaces in $\mathbb{R}^2$ and $\mathbb{R}^3$. Over the past two decades, a number of specialized techniques have been developed to show the existence of small-sized ϵ-nets for such set systems [3, 4, 5, 6, 7, 8, 11, 12, 14, 16, 20, 25, 26]. Based on these successes, it was generally believed that in most geometric scenarios one should be able to substantially strengthen the ϵ-net theorem, and obtain perhaps even a $O\big(\frac{1}{\epsilon}\big)$ upper bound for the size of the smallest ϵ-nets. In this direction, there have been two significant recent developments: one positive and one negative.

Upper bounds. Following the work of Clarkson and Varadarajan [8], it has been gradually realized that if one replaces the condition that the range space $(X, \mathcal{R})$ has bounded VC-dimension by a more refined combinatorial property, one can prove the existence of ϵ-nets of size $o(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$. To formulate this property, we need to introduce some terminology.

Given a function $\varphi : \mathbb{N} \to \mathbb{R}^+$, we say that the primal range space $(X, \mathcal{R})$ has *shallow-cell complexity* φ if there exists a constant $c = c(\mathcal{R}) > 0$ such that, for every $Y \subseteq X$ and for every positive integer l, the number of at most l-element sets in $\mathcal{R}|_Y$ is $O\big(|Y| \cdot \varphi(|Y|) \cdot l^c\big)$. Note that if the VC-dimension of $(X, \mathcal{R})$ is d, then for every $Y \subseteq X$, the number of elements of the projection of the set system $\mathcal{R}$ to Y satisfies $|\mathcal{R}|_Y| = O(|Y|^d)$. However, the condition that $(X, \mathcal{R})$ has *shallow-cell complexity* φ for some function $\varphi(n) = O(n^{d'}), 0 < d' < d - 1$ and some constant $c = c(\mathcal{R})$, implies not only that $|\mathcal{R}|_Y| = O(|Y|^{1+d'+c})$, but it reveals some nontrivial finer details about the distribution of the sizes of the smaller members of $\mathcal{R}|_Y$.

Several of the range spaces mentioned earlier turn out to have low shallow-cell complexity. For instance, the primal range spaces induced by containment of points in disks in $\mathbb{R}^2$ or half-spaces in $\mathbb{R}^3$ have shallow-cell complexity $\varphi(n) = O(1)$. In general, it is known [13] that the primal range space induced by containment of points by half-spaces in $\mathbb{R}^d$ has shallow-cell complexity $\varphi(n) = O\big(n^{\lfloor d/2 \rfloor - 1}\big)$.

Define the *union complexity* of a family of objects $\mathcal{R}$, as the maximum number of *faces* (boundary pieces) of all dimensions that the union of any n members of $\mathcal{R}$ can have; see [1]. Applying a simple probabilistic technique developed by Clarkson and Shor [9], one can find an interesting relationship between the union complexity of a family of objects $\mathcal{R}$ and the shallow-cell complexities of the *dual* range spaces induced by subsets $\mathcal{S} \subset \mathcal{R}$. Suppose that the union complexity of a family $\mathcal{R}$ of objects in the plane is $O\big(n\varphi(n)\big)$, for some "well-behaved" non-decreasing function φ. Then the number of at most l-element subsets in the dual range space induced by any $\mathcal{S} \subset \mathcal{R}$ is $O\big(l^2 \cdot \frac{|\mathcal{S}|}{l}\varphi(\frac{|\mathcal{S}|}{l})\big) = O\big(|\mathcal{S}|\varphi(|\mathcal{S}|)l\big)$ [22]; i.e., the dual range space induced by $\mathcal{S}$ has shallow-cell complexity $O\big(\varphi(n)\big)$. According to the above definitions, this means that for any $\mathcal{S} \subset \mathcal{R}$ and for any positive integer l, the number of l-element subsets $\mathcal{S}' \subseteq \mathcal{S}$ for which there is a point $p' \in \mathbb{R}^2$ contained in all elements of $\mathcal{S}'$, but in none of the elements of $\mathcal{S} \setminus \mathcal{S}'$, is at most $O\big(|\mathcal{S}|\varphi(|\mathcal{S}|)l\big)$. Note that for small values of l, the points p' are not heavily covered (l times). Thus, the corresponding *cells* $\bigcap_{S\in\mathcal{S}'} S \setminus \bigcup_{T\in\mathcal{S}\setminus\mathcal{S}'} T$ of the arrangement $\mathcal{S}$ are "shallow," and the number of these shallow cells is bounded from above. This explains the use of the term "shallow-cell complexity".

A series of elegant results [3, 6, 26] illustrate that if the shallow-cell complexity of a set system is $\varphi(n) = o(n)$, then it permits smaller ϵ-nets than what is guaranteed by the ϵ-net theorem. The following theorem represents the current state of the art (see [15] for a simple proof).

▶ **Theorem 1.** Let $(X, \mathcal{R})$ be a range space with shallow-cell complexity φ, where $\varphi(n) = O(n^d)$ for some constant d. Then, for every $\epsilon > 0$, it has an ϵ-net of size $O\big(\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon})\big)$, where the constant hidden in the O-notation depends on d.

Proof. (Sketch.) The main result in [6] shows the existence of ϵ-nets of size $O\big(\frac{1}{\epsilon}\log\varphi(|X|)\big)$

for any non-decreasing function φ[1]. To get a bound independent of $|X|$, first compute a small $(\epsilon/2)$-approximation $A \subseteq X$ for $(X, \mathcal{R})$ [13]. It is known that there is such an A with $|A| = O\big(\frac{d}{\epsilon^2}\log\frac{1}{\epsilon}\big) = O(\frac{1}{\epsilon^3})$, and for any $R \in \mathcal{R}$, we have $\frac{|R\cap A|}{|A|} \geq \frac{|R|}{|X|} - \frac{\epsilon}{2}$. In particular, any $R \in \mathcal{R}$ with $|R| \geq \epsilon|X|$ contains at least an $\frac{\epsilon}{2}$-fraction of the elements of A. Therefore, an $(\epsilon/2)$-net for $(A, \mathcal{R}|_A)$ is an ϵ-net for $(X, \mathcal{R})$. Computing an $(\epsilon/2)$-net for $(A, \mathcal{R}|_A)$ gives the required set of size $O\big(\frac{2}{\epsilon}\log\varphi(|A|)\big) = O\big(\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon^3})\big) = O\big(\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon})\big)$. ◀

Lower bounds. It was conjectured for a long time [14] that most geometrically defined range spaces of bounded Vapnik-Chervonekis dimension have "linear-sized" ϵ-nets, i.e., ϵ-nets of size $O\big(\frac{1}{\epsilon}\big)$. These hopes were shattered by Alon [2], who established a superlinear (but barely superlinear!) lower bound on the size of ϵ-nets for the primal range space induced by straight lines in the plane. Shortly after, Pach and Tardos [19] managed to establish a tight lower bound, $\Omega(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ for the size of ϵ-nets in primal range spaces induced by half-spaces in $\mathbb{R}^4$, and in several other geometric scenarios.

▶ **Theorem 2.** [19] Let $\mathcal{F}$ denote the family of half-spaces in $\mathbb{R}^4$. For any $\epsilon > 0$, there exist point sets $X \subset \mathbb{R}^4$ such that in the (primal) range spaces $(X, \mathcal{F})$, the size of every ϵ-net is at least $\frac{1}{9\epsilon}\log\frac{1}{\epsilon}$.

Our contributions

The aim of this paper is to complete both avenues of research opened by Theorems 1 and 2. Our following theorem, proved in Section 2, generalizes Theorem 2 to higher dimensions $d \geq 4$. It provides an asymptotically tight bound in terms of both ε and d, and hence completely settles the ϵ-net problem for half-spaces.

▶ **Theorem 3.** For any integer $d \geq 1$ and any $\epsilon > 0$, there exist primal range spaces $(X, \mathcal{F})$ induced by point sets X and collections of half-spaces $\mathcal{F}$ in $\mathbb{R}^{2d+2}$ such that the size of every ϵ-net for $(X, \mathcal{F})$ is at least $\frac{d}{9\epsilon}\log\frac{1}{\epsilon}$. In particular, for any integer $d' \geq 4$, this implies the existence of point sets X in $\mathbb{R}^{d'}$ such that any ϵ-net for the primal set system $\mathcal{F}$ induced by halfspaces in $\mathbb{R}^{d'}$ has size at least $\frac{\lfloor d'/2\rfloor - 1}{9\epsilon}\log\frac{1}{\epsilon}$.

As was mentioned in the first subsection, for any $d \geq 1$, the VC-dimension of any range space induced by points and half-spaces in $\mathbb{R}^d$ is at most $d+1$. Thus, Theorem 3 matches, up to a constant factor independent of d and ϵ, the upper bound implied by the ϵ-net theorem of Haussler and Welzl. Noga Alon pointed out to us that it is very easy to show that for a fixed $\epsilon > 0$, the lower bound for ϵ-nets in range spaces induced by half-spaces in $\mathbb{R}^d$ has to grow at least linearly in d. To see this, suppose that we want to obtain a $\frac{1}{3}$-net, say, for the range space induced by *open* half-spaces on a set X of $3d$ points in general position in $\mathbb{R}^d$. Notice that for this we need at least $d+1$ points. Indeed, any d points of X span a hyperplane, and one of the open half-spaces determined by this hyperplane contains at least $\frac{|X|}{3}$ points.

The key element of the proof of Theorem 2 [19] was to construct a set $\mathcal{B}$ of $(k+3)2^{k-2}$ axis-parallel rectangles in the plane such that for any subset of them there is a set Q of at most 2^{k-1} points that hit none of the rectangles that belong to this subset and all the rectangles in its complement (the precise statement is given in Section 3). In Section 4, we

[1] Their result is in fact for the more general problem of *small weight* ϵ-nets.

generalize this statement to $\mathbb{R}^d$ by constructing roughly d times more axis-parallel boxes[2] than in the planar case, but the size of the set Q remains the same size. We will prove

▶ **Lemma 4.** *Let $k, d \geq 2$ be integers. Then there exists a set $\mathcal{B}$ of $d(k+1)2^{k-2}$ axis-parallel boxes in $\mathbb{R}^{d+1}$ such that for any subset $\mathcal{S} \subseteq \mathcal{B}$ there exists a 2^{k-1}-element set Q of points in $\mathbb{R}^{d+1}$ with the property that*

(i) *$Q \cap B \neq \emptyset$ for any $B \in \mathcal{B} \setminus \mathcal{S}$, and*
(ii) *$Q \cap B = \emptyset$ for any $B \in \mathcal{S}$.*

In the next section we show how this lemma implies the bound of Theorem 3, which is d times better than the bound in Theorem 2. In Section 3 we give a proof of a weaker form of Lemma 4, which is easy to deduce from the main lemma of [19]. Applying this lemma to obtain the lower bounds for ϵ-nets would result in getting bounds that are roughly twice as worse than the ones stated in Theorem 2. The proof of Lemma 4 will be given in Section 4.

In Section 5, we show that the bound in Theorem 1 cannot be improved.

▶ **Definition 5.** A function $\varphi : \mathbb{R}^+ \to \mathbb{R}^+$ is called *submultiplicative* if there exists an $\alpha, 0 < \alpha < 1$ such that

1. $\varphi^\alpha(x) \leq \varphi(x^\alpha)$ for all sufficiently large $x \in \mathbb{R}^+$, and
2. $\varphi(xy) \leq \varphi(x)\varphi(y)$ for all sufficiently large $x, y \in \mathbb{R}^+$.

▶ **Theorem 6.** *Let d be a fixed positive integer and let $\varphi : \mathbb{R}^+ \to \mathbb{R}^+$ be any submultiplicative function with $\varphi(n) = O(n^d)$. Then, for any $\epsilon > 0$ there exist range spaces $(X, \mathcal{F})$ that have*

(i) *shallow-cell complexity φ, and for which*
(ii) *the size of any ϵ-net is at least $\Omega(\frac{1}{\epsilon} \log \varphi(\frac{1}{\epsilon}))$.*

Note that if $\varphi(n) = \Omega(n)$, then this theorem yields a lower bound for the size of the smallest ϵ-nets in the constructed range spaces which is somewhat weaker than the bound $\Omega(\frac{d}{\epsilon} \log \frac{1}{\epsilon})$, valid for the old constructions in [11]. Indeed, the property that the VC-dimension is at most d, imposed on the range spaces constructed in [11], implies that the range space has shallow-cell complexity $O(n^{d-1})$.

Theorem 6 becomes interesting when $\varphi(n) = o(n)$ and the upper bound $O\big(\frac{1}{\epsilon} \log \varphi(\frac{1}{\epsilon})\big)$ in Theorem 1 *improves* on the general upper bound $O\big(\frac{1}{\epsilon} \log \frac{1}{\epsilon}\big)$ guaranteed by the ϵ-net theorem. Theorem 6 shows that, if $\varphi(n) = o(n)$, even this improved bound is asymptotically tight.

The best upper and lower bounds for the size of small ϵ-nets in range spaces with a given shallow-cell complexity φ are based on purely combinatorial arguments, and they imply directly or indirectly all known results on ϵ-nets in geometrically defined range spaces (see [17] for a detailed discussion). This suggests that the introduction of the notion of shallow-cell complexity provided the right framework for ϵ-net theory.

2 Proof of Theorem 3 using Lemma 4

Let $\mathcal{B}$ be a set of $(d+1)$-dimensional axis-parallel boxes in $\mathbb{R}^{d+1}$. We recall that the *dual range space* induced by $\mathcal{B}$ is the set system (hypergraph) on the ground set $\mathcal{B}$ consisting of the sets $\mathcal{B}_p := \{B \in \mathcal{B} : p \in B\}$ for all $p \in \mathbb{R}^{d+1}$.

[2] An *axis-parallel box* in $\mathbb{R}^d$ is the Cartesian product of $d+1$ intervals. For simplicity, in the sequel, they will be called "boxes".

▶ **Lemma 7.** *Consider the dual range space induced by a set of axis-parallel boxes $\mathcal{B}$ in $\mathbb{R}^{d+1}$. Then there exists a function $f : \mathcal{B} \to \mathbb{R}^{2d+2}$ such that for every point $p \in \mathbb{R}^{d+1}$, there is a half-space H in $\mathbb{R}^{2d+2}$ with $\{f(B) : B \in \mathcal{B}_p\} = H \cap \{f(B) : B \in \mathcal{B}\}$.*

Proof. By translation, we can assume that all the boxes in $\mathcal{B}$ lie in the positive orthant of $\mathbb{R}^{d+1}$.

Consider the function $g : \mathcal{B} \to \mathbb{R}^{2d+2}$, which maps a box $B = [x_1^l, x_1^r] \times [x_2^l, x_2^r] \times \cdots \times [x_{d+1}^l, x_{d+1}^r]$ to the point $(x_1^l, 1/x_1^r, x_2^l, 1/x_2^r, \ldots, x_{d+1}^l, 1/x_{d+1}^r)$ lying in the positive orthant of $\mathbb{R}^{2d+2}$. For any $p = (a_1, a_2, \ldots, a_{d+1})$ in the positive orthant, let C_p denote the box $[0, a_1] \times [0, 1/a_1] \times [0, a_2] \times [0, 1/a_2] \times \cdots \times [0, a_{d+1}] \times [0, 1/a_{d+1}]$. Obviously, p lies in a box B if and only if $g(B) \in C_p$.

Thus, g maps the set of boxes in $\mathcal{B}$ to a set of points in $\mathbb{R}^{2d+2}$, such that for any point p in the positive orthant of $\mathbb{R}^{d+1}$, the set of boxes $\mathcal{B}_p \subset \mathcal{B}$ that contain p are mapped to the set of points that belong to the box B_p. (Note that B_p contains the origin.)

We complete the proof by applying the following simple lemma (Lemma 2.3 in [19]) to the set $Q = g(\mathcal{B})$: To each point $q \in Q$ in the positive orthant of $\mathbb{R}^{2d+2}$, we can assign another point q' in the positive orthant of $\mathbb{R}^{2d+2}$ such that for each box in $\mathbb{R}^{2d+2}$ that contains the origin there is a half-space with the property that q belongs to the box if and only if q' belongs to the corresponding half-space. The mapping $f(B) = (g(B))'$ for every $B \in \mathcal{B}$ meets the requirements of the lemma. ◀

Now we are in a position to establish Theorem 3 using Lemma 4. Let $\epsilon = \frac{\alpha}{2^{k-1}}$ with $k \in \mathbb{N}, k \geq 2$, and $\frac{1}{3} \leq \alpha \leq \frac{2}{3}$. Applying Lemma 4, we obtain a set $\mathcal{B}$ of $d(k+1)2^{k-2}$ boxes in $\mathbb{R}^{d+1}$. We claim that the dual range space induced by these boxes does not admit an ϵ-net of size $(1-\alpha)|\mathcal{B}|$.

Assume for contradiction that there is an ϵ-net $\mathcal{S} \subseteq \mathcal{B}$ with $|\mathcal{S}| \leq (1-\alpha)|\mathcal{B}|$. According to Lemma 4, there exists a set Q of 2^{k-1} points in $\mathbb{R}^{d+1}$ with the property that no box in $\mathcal{S}$ contains any point of Q, but every member of $\mathcal{B} \setminus \mathcal{S}$ does. By the pigeonhole principle, there is a point $p \in Q$ contained in at least $|\mathcal{B} \setminus \mathcal{S}|/|Q|$ members of $\mathcal{B} \setminus \mathcal{S}$. We have

$$\frac{|\mathcal{B} \setminus \mathcal{S}|}{|Q|} \geq \frac{\alpha|\mathcal{B}|}{|Q|} = \frac{\alpha|\mathcal{B}|}{2^{k-1}} = \epsilon|\mathcal{B}|.$$

Thus, none of the at least $\epsilon|\mathcal{B}|$ members of $\mathcal{B}$ hit by p belong to $\mathcal{S}$, contradicting the assumption that $\mathcal{S}$ was an ϵ-net.

Hence, the size of any ϵ-net in the dual range space induced by $\mathcal{B}$ is at least $(1-\alpha)|\mathcal{B}| = (1-\alpha)d(k+1)2^{k-2} = \frac{(1-\alpha)\alpha}{2} \cdot \frac{d}{\epsilon}(k+1) \geq \frac{d}{9\epsilon} \log \frac{1}{\epsilon}$.

By Lemma 7, any lower bound for the size of ϵ-nets in the dual range space induced by the set $\mathcal{B}$ of boxes in $\mathbb{R}^{d+1}$ gives the same lower bound for the size of an ϵ-net in the (primal) range space on the set of points $f(\mathcal{B}) \subset \mathbb{R}^{2d+2}$ corresponding to these boxes, in which the ranges are half-spaces in $\mathbb{R}^{d+1}$. Thus, Theorem 3 follows immediately for even values of d, and with a slight loss in the constant, also for odd values (applying the lower bound for $d-1$). ◀

The system of boxes constructed above has a fixed number of elements, depending on the value of $1/\epsilon$. We can obtain arbitrarily large constructions by replacing each box of $B \in \mathcal{B}$ with several slightly translated copies of B (we refer the reader to [19] for details).

3 Proof of a weaker form of Lemma 4

In this section we sketch a proof of a weaker form of Lemma 4.

▶ **Lemma 8.** *Let $k, d \geq 1$ be integers. Then there exists a set $\mathcal{B}$ of $\lfloor \frac{d+1}{2} \rfloor (k+3)2^{k-2}$ axis-parallel boxes in $\mathbb{R}^{d+1}$ such that for any subset $\mathcal{S} \subseteq \mathcal{B}$ there exists a 2^{k-1}-element set Q of points with the property that*

(i) *$Q \cap B \neq \emptyset$ for any $B \in \mathcal{B} \setminus \mathcal{S}$, and*
(ii) *$Q \cap B = \emptyset$ for any $B \in \mathcal{S}$.*

Via the same proof as the one given in Section 2 one can get that there exist primal range spaces $(X, \mathcal{F})$ induced by point sets X and collections of half-spaces $\mathcal{F}$ in $\mathbb{R}^{2d+2}$ such that the size of every ϵ-net for $(X, \mathcal{F})$ is at least $\frac{\lfloor \frac{d+1}{2} \rfloor}{9\epsilon} \log \frac{1}{\epsilon}$. Lemma 8 follows directly from

▶ **Lemma 9** ([19])**.** *Let $k \geq 2$ be an integer. Then there exists a set $\mathcal{R}$ of $(k+3)2^{k-2}$ axis-parallel rectangles in $\mathbb{R}^2$ such that for any $\mathcal{S} \subseteq \mathcal{R}$ there exists a 2^{k-1}-element set Q of points in $\mathbb{R}^2$ with the property that*

(i) *$Q \cap R \neq \emptyset$ for any $R \in \mathcal{R} \setminus \mathcal{S}$, and*
(ii) *$Q \cap R = \emptyset$ for any $R \in \mathcal{S}$.*

Denote the x-coordinate and y-coordinate of a point $p \in \mathbb{R}^2$ by $x(p)$ and $y(p)$ respectively, and set $m = \lfloor \frac{d+1}{2} \rfloor$. Let $\mathcal{R} = \{R_1, \dots, R_t\}$, $t = (k+3)2^{k-2}$, be a set of rectangles satisfying the conditions of Lemma 9. By scaling, one can assume that $R \subset [0,1]^2$ for every $R \in \mathcal{R}$.

For $i = 1 \dots m$, define the function f_i mapping a point in $\mathbb{R}^2$ to a product of $(d+1)$ intervals in $\mathbb{R}^{d+1}$, as follows.

$$p \in \mathbb{R}^2, \quad f_i(p) \quad = \quad \underbrace{[0,1] \times \dots \times [0,1]}_{2i-2 \text{ intervals}} \times x(p) \times y(p) \times \underbrace{[0,1] \times \dots \times [0,1]}_{d+1-2i \text{ intervals}}.$$

This mapping lifts each rectangle $R \in \mathcal{R}$ to the box $f_i(R) = \{f_i(p) \ : \ p \in R\}$, and each set of rectangles $\mathcal{R}' \subseteq \mathcal{R}$ to the set of boxes $f_i(\mathcal{R}') = \{f_i(R) \mid R \in \mathcal{R}'\}$.

Let $\mathcal{B} = \bigcup_i f_i(\mathcal{R})$ be the required set of $(k+3)\lfloor \frac{d+1}{2} \rfloor 2^{k-2}$ boxes in $\mathbb{R}^{d+1}$. Given a set $\mathcal{S} \subseteq \mathcal{B}$, let $\mathcal{R}_i \subseteq \mathcal{R}$, $i = 1 \dots m$, be such that $\mathcal{S} \cap f_i(\mathcal{R}) = f_i(\mathcal{R}_i)$. Using Lemma 9, let $Q_i = \{q_1^i, \dots, q_{2^{k-1}}^i\} \subset \mathbb{R}^2$ be the set of points hitting all rectangles in $\mathcal{R} \setminus \mathcal{R}_i$, and no rectangle in $\mathcal{R}_i$. Now observe that the set

$$Q = \begin{cases} \left\{ \left(x(q_j^1), y(q_j^1), \dots, x(q_j^m), y(q_j^m) \right) \ : \ j \in [1, 2^{k-1}] \right\} & \text{if } d \text{ is odd.} \\ \left\{ \left(x(q_j^1), y(q_j^1), \dots, x(q_j^m), y(q_j^m), 1 \right) \ : \ j \in [1, 2^{k-1}] \right\} & \text{if } d \text{ is even.} \end{cases}$$

of 2^{k-1} points in $\mathbb{R}^{d+1}$ is the required set for $\mathcal{S}$: any rectangle $R \in \mathcal{R} \setminus \mathcal{R}_i$ contains a unique point $q \in Q_i$, and thus $f(R)$ contains the unique point of Q with $x(q)$ and $y(q)$ in its $(2i-1)$-th and $2i$-th coordinates. Similarly, a rectangle $R \in \mathcal{R}_i$ is not hit by any point of Q_i, and thus is not hit by any point of Q.

Instead of lifting the rectangles in $\lfloor \frac{d+1}{2} \rfloor$ disjoint coordinates in $\mathbb{R}^{d+1}$, the proof of Lemma 4 shows how to pack them more carefully into d coordinates, thus improving the above bound by a factor of two.

4 Proof of Lemma 4

The desired family $\mathcal{B}$ of axis-parallel boxes will be contained in $K = [0,1]^{d+1}$, where each box in $\mathcal{B}$ is a Cartesian product of $d+1$ intervals. For ease of exposition, we will identify intervals with binary sequences; namely, a binary sequence $0.l_1 l_2 \dots l_s$ will correspond to the interval $(0.l_1 l_2 \dots l_s 000 \dots, 0.l_1 l_2 \dots l_s 111 \dots) \subset (0,1)$. For example, the sequence 0 corresponds to the interval $(0,1)$, the sequence 0.0 corresponds to the interval $(0, 1/2)$ and so on. We call s the *size* of the sequence. The "trivial" sequence 0 is of size 0, 0.0 of size 1 and so on. Note

that sequences of size s correspond to intervals of Euclidean length 2^{-s}. We denote both sequences and the corresponding intervals by capital letters X, Y with subscripts, and we denote the size of any sequence X by $\text{size}(X)$.

We have $\mathcal{B} := \bigcup_{i=1}^{d} \mathcal{B}^i$, where each $B \in \mathcal{B}^i$ has the form

$$B = \underbrace{0 \times 0 \times \ldots \times X_i \times X_{i+1} \times 0 \times \ldots \times 0}_{d+1 \text{ intervals}}.$$

The only "non-trivial" intervals in B – that is, not equal to $(0, 1)$ – are the i-th and the $(i+1)$-th ones. When clear from the context, we will omit the $(d-1)$ trivial intervals, and simply write $B = X_i \times X_{i+1}$ for $B \in \mathcal{B}^i$ and where X_i, X_{i+1} are binary sequences representing the corresponding intervals. Set $\mathcal{B}^i := \mathcal{B}^i_1 \bigcup \cdots \bigcup \mathcal{B}^i_k$, where

$$\mathcal{B}^i_j := \big\{X_i \times X_{i+1} : \ X_i = 0.l_1 \ldots l_{k-j}, \ X_{i+1} = 0.m_1 \ldots m_j, \ l_{k-j} = m_j = 1\big\}$$

for $1 \le j \le k-1$ and

$$\mathcal{B}^i_k := \big\{X_i \times X_{i+1} : \ X_i = 0.l_1 \ldots l_k, l_k = 1, \ X_{i+1} = 0\big\}.$$

The construction of $\mathcal{B}$ is complete. For every i and $1 \le j \le k-1$, we have $|\mathcal{B}^i_j| = 2^{k-2}$. We also have $|\mathcal{B}^i_k| = 2^{k-1}$. Then, $|\mathcal{B}| = \sum_{i=1}^{d} \sum_{j=1}^{k} |\mathcal{B}^i_j| = d(k+1)2^{k-2}$. It remains to show the existence of the desired set Q for any set $\mathcal{S} \subseteq \mathcal{B}$. We start with the following crucial observation, stated without proof.

▶ **Observation 10.** *The two boxes $X = X_1 \times X_2 \times \ldots \times X_{d+1}$ and $Y = Y_1 \times Y_2 \times \ldots \times Y_{d+1}$ intersect if and only if for each $i \in [1, d+1]$, one of X_i or Y_i is a subsequence of the other (we will consider 0 to be a subsequence of every other sequence). Moreover, if this is the case, then we have $X \cap Y = Z_1 \times \ldots \times Z_{d+1}$, where $Z_i = \arg\max \big\{\text{size}(X_i), \text{size}(Y_i)\big\}$.*

It will be useful to define the following larger set of boxes:

$$(i,j)\text{-level} := \big\{X_i \times X_{i+1} : X_i \text{ is a sequence of size } k-j, X_{i+1} \text{ is a sequence of size } j\big\}.$$

Note that the length of the interval in the i-th and $(i+1)$-th coordinates is 2^{-k+j} and 2^{-j}, respectively, for the (i,j)-level. Also, for any i and j, the boxes from the (i,j)-level are disjoint, with their closures forming a cover of the cube K.

Fix some $i \in [1, d]$ and $j \in [1, k-1]$. We say *four* boxes from the (i,j)-level are *grouped* if the corresponding sequences for the i-th and $(i+1)$-th coordinate of these boxes differ only in the last bit. This provides us with the partition of the boxes on the (i,j)-level into 2^{k-2} groups. Denote this set of groups by $\mathcal{G}(i,j)$. Note that for every group G, we have $|G \cap \mathcal{B}| = 1$. Given $\mathcal{S} \subseteq \mathcal{B}$, we define the following set of boxes:

$$\mathcal{H}(i,j) := \bigcup_{\substack{G \in \mathcal{G}_{i,j} \\ |G \cap \mathcal{S}| = 0}} \{B \in G : B = X_i \times X_{i+1}, \text{ sum of last digits of } X_i, X_{i+1} \text{ is even}\} \bigcup \bigcup_{\substack{G \in \mathcal{G}_{i,j} \\ |G \cap \mathcal{S}| = 1}} \{B \in G : B = X_i \times X_{i+1}, \text{ sum of last digits of } X_i, X_{i+1} \text{ is odd}\}. \quad (1)$$

For $j = k$, pair the boxes into 2^{k-2} groups of two boxes each, where the two boxes $X_i \times 0$ and $X'_i \times 0$ are paired together if X_i, X'_i differ in the last bit and choose one box from each pair into $\mathcal{H}(i,k)$ such that $\mathcal{H}(i,k) \cap \mathcal{B} = \mathcal{B} \setminus \mathcal{S}$.

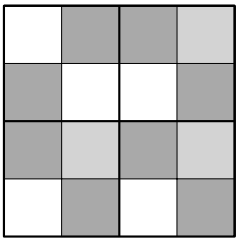
Note that each box $B \in \mathcal{H}(i,j)$ belongs to the (i,j)-level, and so is of the form $B = X_i \times X_{i+1}$, where X_i has size $k-j$ and X_{i+1} has size j. Set

$$\mathcal{H} = \bigcup_{\substack{i\in[1,d]\\ j\in[1,k-1]}} \mathcal{H}(i,j).$$

For each $B = X_i \times X_{i+1} \in \mathcal{B}^i_j$, $1 \le j \le k-1$, the sum of the last digits of X_i and X_{i+1} is even, and so a simple but crucial property of the system of boxes $\mathcal{H}$ is that

$$\mathcal{H} \cap \mathcal{B} = \mathcal{B} \setminus \mathcal{S}. \tag{2}$$

The construction of the set $\mathcal{H}(i,j)$ is illustrated below. The groups on the (i,j)-level are bounded by thick lines, and the rectangles from the (i,j)-level that belong to $\mathcal{H}(i,j)$ are shaded dark gray, while the rectangles of $\mathcal{S}$ are shaded light gray. In each group the upper right box belongs to $\mathcal{B}$. We choose two rectangles from each group to add to $\mathcal{H}(i,j)$, depending on whether the rectangle of $\mathcal{B}$ in that group belongs to $\mathcal{S}$ or not.



The set Q we are going to construct will be a hitting set for $\mathcal{H}$. This is sufficient to prove the lemma: note that $|Q| = |\mathcal{H}(i,j)| = 2^{k-1}$ for each i,j, and since the boxes at the (i,j)-level are disjoint, each point from Q must hit exactly one box from $\mathcal{H}(i,j)$ and hence no box of $\mathcal{S}$ (by equation (2)).

Before we describe the construction of Q, we define the set of *hitting boxes* $\mathcal{A}(i,j)$:

1. $\mathcal{A}(1,1) = \mathcal{H}(1,1)$,
2. For $i \in [1,d]$, $j \in [2,k]$

$$\mathcal{A}(i,j) = \{A \cap H : A \in \mathcal{A}(i,j-1),\ H \in \mathcal{H}(i,j), A \cap H \neq \emptyset\},$$

3. For $i \in [2,d]$

$$\mathcal{A}(i,1) = \{A \cap H : A \in \mathcal{A}(i-1,k),\ H \in \mathcal{H}(i,1), A \cap H \neq \emptyset\}.$$

The key properties of the sets of hitting boxes are formulated in the following lemma.

▶ **Lemma 11.** *Let $\mathcal{A}(\cdot,\cdot)$ be as defined above. Then*

(i) *For $i \in [2,d]$, each $A \in \mathcal{A}(i-1,k)$ intersects exactly one box from $\mathcal{H}(i,1)$. Moreover, each box $H \in \mathcal{H}(i,1)$ is intersected by some $A \in \mathcal{A}(i-1,k)$.*

(ii) *Let $i \in [1,d]$, and $j \in [2,k]$. Then each $A \in \mathcal{A}(i,j-1)$ intersects exactly one box from $\mathcal{H}(i,j)$. Moreover, each box $H \in \mathcal{H}(i,j)$ is intersected by some $A \in \mathcal{A}(i,j-1)$.*

Proof. The proof of the lemma is by induction on the pair (i,j) with lexicographic ordering. By construction of $\mathcal{A}(\cdot,\cdot)$, for each box $A \in \mathcal{A}(i,j)$:

$$A = (H_{i,j} \cap \ldots \cap H_{i,1}) \bigcap (H_{i-1,k} \cap \ldots \cap H_{i-1,1}) \bigcap \ldots \bigcap (H_{1,k} \cap \ldots \cap H_{1,1}) \tag{3}$$

where $H_{i,j} \in \mathcal{H}(i,j)$.

Proof of (i). By equation (3) and Observation 10, each box $A \in \mathcal{A}(i-1,k)$ has the form $A = X_1 \times \ldots \times X_i \times 0 \times \ldots \times 0$, where for each $j \in [1,i]$, X_j has size k. In particular, X_i is of size k. On the other hand, for $H \in H(i,1)$ we have $H = 0 \times \ldots \times 0 \times Y_i \times Y_{i+1} \times 0 \times \ldots \times 0$, where Y_i is a sequence of size $k-1$ and Y_{i+1} is a sequence of size 1. Moreover, for each sequence X'_i of size $k-1$ there is exactly one $H \in \mathcal{H}(i,1)$ such that $H = 0 \times \ldots \times 0 \times X'_i \times Y_{i+1} \times 0 \times \ldots \times 0$. To see that, one has to note that after fixing a sequence X'_i we determine the last digit of Y_{i+1} in a unique way based on the even/odd sum criterion from (1). But the last digit is the whole sequence Y_{i+1}. Thus, defining X'_i as X_i without the last bit, we get the first part of (i).

On the other hand, by induction, each of the elements from $\mathcal{H}(i-1,k)$ contains one box from $\mathcal{A}(i-1,k)$. This implies that among the elements of $\mathcal{A}(i-1,k)$ all sequences X'_i of length $k-1$ are present. Therefore, for each $H \in \mathcal{H}(i,1)$, $H = 0 \times \ldots \times Y_i \times Y_{i+1} \times 0 \times \ldots \times 0$, there exists a box $A \in \mathcal{A}(i-1,k)$ where $A = X_1 \times \ldots \times X_{i-1} \times Y_i \times 0 \times \ldots \times 0$; by Observation 10, H intersects A.

Proof of (ii). The proof of this part is similar to the previous one. By equation (3) and Observation 10, each box $A \in \mathcal{A}(i,j-1)$ has the form $A = X_1 \times \ldots \times X_{i+1} \times 0 \times \ldots \times 0$, where $X_1, \ldots, X_i$ are sequences of size $k-1$ and X_{i+1} is of size $j-1$. Let $X_i = 0.l_1 \ldots l_{k-1}, X_{i+1} = 0.m_1 \ldots m_{j-1}$. We claim that there is a unique element $H \in \mathcal{H}(i,j)$, such that $H = 0 \times \ldots \times Y_i \times Y_{i+1} \times 0 \times \ldots \times 0$, where $Y_i = 0.l_1 \ldots l_{k-j}, Y_{i+1} = 0.m_1 \ldots m_{j-1}x$, where x is either 0 or 1. Indeed, there are two such boxes in the (i,j)-level, but the value of x is again uniquely determined based on the even/odd condition from (1) or the simpler condition for $j = k$. It is easy to see that H is the only element from $\mathcal{H}(i,j)$ that satisfies the containment relation from Observation 10 with A.

To prove the second part of the claim, we again use induction. For every box $H' \in \mathcal{H}(i,j-1)$ there is an element $A \in \mathcal{A}(i,j-1)$ contained in it. Therefore, for each sequence $Y_i = 0.l_1 \ldots l_{k-j}, Y_{i+1} = 0.m_1 \ldots m_{j-1}$ there is an element $A \in \mathcal{A}(i,j-1)$ that contains these two sequences as subsequences on the i-th and $(i+1)$-st coordinate. On the other hand, each $H \in \mathcal{H}(i,j)$ is determined by such sequences Y_i, Y_{i+1}. Therefore each H intersects some A. ◀

It is easy to deduce from Lemma 11 that $|\mathcal{A}(i,j)| = 2^{k-1}$ for each $i \in [1,d]$ and $j \in [1,k]$. Moreover, each box of $\mathcal{H}$ is hit by one of the boxes of $\mathcal{A}(d,k)$. Choose one point from each box of $\mathcal{A}(d,k)$ to get the desired set Q.

5 Proof of Theorem 6

The goal of this section is to establish lower bounds on the sizes of ϵ-nets in range spaces with given shallow-cell complexity φ. Theorem 6 is a consequence of the following more precise statement.

▶ **Theorem 12.** *Let $\varphi : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically increasing submultiplicative function*[3]*, which tends to infinity and is bounded from above by a polynomial of constant degree.*

For any $0 < \delta < \frac{1}{10}$ one can find an $\epsilon_0 > 0$ with the following property: for any $0 < \epsilon < \epsilon_0$, there exists a range space on a set of $n = \frac{\log \varphi(\frac{1}{\epsilon})}{\epsilon}$ elements with shallow-cell complexity φ, in which the size of every ϵ-net is at least $\frac{(1-4\delta)}{\epsilon} \log \varphi(\frac{1}{\epsilon})$.

[3] Compare with Definition 5.

Proof. The parameters of the range space are as follows:

$$n = \frac{\log \varphi(\frac{1}{\epsilon})}{\epsilon}, \quad m = \epsilon n = \log \varphi\big(\frac{1}{\epsilon}\big), \quad p = \frac{n\varphi^{1-2\delta}(n)}{\binom{n}{m}}.$$

Let d be the smallest integer such that $\varphi(n) = O(n^d)$. In fact, we will assume that $n^{d-1} \le \varphi(n) \le c_1 n^d$, for a suitable constant $c_1 \ge 1$, provided that n is large enough. In the most interesting case, when $\varphi(n) = o(n)$, we have $d = 1$. Using that $n \ge \frac{\log \varphi(1/\epsilon)}{\epsilon}$, if $\epsilon < \epsilon_0$, we have the following logarithmic upper bound on m.

$$m = \log \varphi\big(\frac{1}{\epsilon}\big) \le \log\big(c_1 \epsilon^{-d}\big) \le d \log \frac{c_1}{\epsilon} \le d \log n \tag{4}$$

Consider a range space $([n], \mathcal{F})$ with a ground set $[n]$ and with a system of m-element subsets $\mathcal{F}$, where each m-element subset of $[n]$ is added to $\mathcal{F}$ independently with probability p. The next claim follows by a routine application of the Chernoff bound.

▶ **Claim 13.** *With high probability,* $|\mathcal{F}| \le 2n\varphi^{1-2\delta}(n)$.

Theorem 12 follows by combining the next two lemmas that show that, with high probability, the range space $([n], \mathcal{F})$

(i) does not admit an ϵ-net of size less than $\frac{(1-4\delta)}{\epsilon} \log \varphi(\frac{1}{\epsilon})$, and

(ii) has shallow-cell complexity φ.

For the proofs, we need to assume that $n = n(\delta, d, \varphi)$ is a sufficiently large constant, or, equivalently, that $\epsilon_0 = \epsilon_0(\delta, d)$ is sufficiently small.

▶ **Lemma 14.** *With high probability, the range space* $([n], \mathcal{F})$ *has shallow-cell complexity* φ.

Proof. It is enough to show that for all sufficiently large $x \ge x_0$, every $X \subseteq [n], |X| = x$, the number of sets of size *exactly* l in $\mathcal{F}|_X$ is $O(x\varphi(x))$, as this implies that the number of sets in $\mathcal{F}|_X$ of size at most l is $O(x\varphi(x)l)$. In the computations below, we will also assume that $l \ge d + 1 \ge 2$; otherwise if $l \le d$, and assuming $x \ge x_0 \ge 2d$, we have

$$\binom{x}{l} \le \binom{x}{d} \le x^d \le x\varphi(x)$$

where the last inequality follows by the assumption on $\varphi(x)$, provided that x is sufficiently large. We distinguish two cases.

Case 1: $x > \frac{n}{\varphi^{\delta/d}(x)}$**.** In this case, we trivially upper-bound $|\mathcal{F}|_X|$ by $|\mathcal{F}|$. By Claim 13, with high probability, we have

$$\begin{aligned}
|\mathcal{F}| &\le 2n \cdot \varphi^{1-2\delta}(n) \le 2n \cdot \Big(\varphi(x) \cdot \varphi\big(\frac{n}{x}\big)\Big)^{1-2\delta} \quad \text{(by the submultiplicativity of } \varphi) \\
&\le 2n \cdot \Big(\varphi(x) \cdot \varphi\big(\varphi^{\delta/d}(x)\big)\Big)^{1-2\delta} \quad \text{(as } n/x \le \varphi^{\delta/d}(x)) \\
&\le 2n \cdot \Big(c_1 \varphi(x) \varphi^{\delta}(x)\Big)^{1-2\delta} \quad \text{(using } \varphi(t) \le c_1 t^d) \\
&\le 2c_1' n \varphi(x)^{1-\delta} \le 2c_1' x \varphi(x)^{1-\delta+\delta/d} = O(x\varphi(x)).
\end{aligned}$$

Case 2: $x \le \frac{n}{\varphi^{\delta/d}(x)}$. Denote the largest integer x that satisfies this inequality by x_1. It is clear that $x_1 = o(n)$ (recall that φ is monotonically increasing and tends to infinity). We also denote the system of all l-element subsets of $\mathcal{F}|_X$ by $\mathcal{F}|_X^l$ and the set of all l-element subsets of X by $\binom{X}{l}$. Let E be the event that $\mathcal{F}$ does not have the required $\varphi(\cdot)$-shallow-cell complexity property. Then $\Pr[E] \le \sum_{l=2}^{m} \Pr[E_l]$, where E_l is the event that for some $X \subset [n]$, $|X| = x$, there are more than $x\varphi(x)$ elements in $\mathcal{F}|_X^l$. Then, for any fixed $l \ge d+1 \ge 2$, we have

$$\begin{aligned}
\Pr[E_l] &\le \sum_{x=x_0}^{x_1} \Pr\Big[\exists X \subseteq [n], |X| = x, |\mathcal{F}|_X^l| > x\varphi(x)\Big] \\
&\le \sum_{x=x_0}^{x_1} \binom{n}{x} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \Pr\Big[\text{For a fixed } X, |X| = x, |\{S \in \mathcal{F}|_X, |S| = l\}| = s\Big] \\
&\le \sum_{x=x_0}^{x_1} \binom{n}{x} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \binom{\binom{x}{l}}{s} \Pr\Big[\text{For a fixed } X, |X| = x, \mathcal{S} \subseteq \binom{X}{l}, |\mathcal{S}| = s, \\
&\qquad\qquad \text{we have } \mathcal{F}|_X^l = \mathcal{S}\Big] \\
&\le \sum_{x=x_0}^{x_1} \binom{n}{x} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \binom{\binom{x}{l}}{s} \Big(1 - (1-p)^{\binom{n-x}{m-l}}\Big)^s (1-p)^{\binom{n-x}{m-l}\left(\binom{x}{l}-s\right)} \qquad (5) \\
&\le \sum_{x=x_0}^{x_1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \Big(\frac{en}{x}\Big)^x \left(\frac{e\left(\frac{ex}{l}\right)^l}{s}\right)^s \Big(p\binom{n-x}{m-l}\Big)^s \qquad (6) \\
&\le \sum_{x=x_0}^{x_1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \Big(\frac{en}{x}\Big)^x \left(\frac{e^{l+1}x^{l-1}}{l^l\varphi(x)} p\binom{n}{m} \frac{m^l}{(n-x-m)^l}\right)^s \qquad (7) \\
&\le \sum_{x=x_0}^{x_1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \Big(\frac{en}{x}\Big)^x \left(\Big(\frac{emx}{n}\Big)^{l-1} \frac{e^2 m \varphi^{1-2\delta}(n)}{\varphi(x)}\right)^s \qquad (8)
\end{aligned}$$

In the transition to the expression (6), we used several times (i) the bound $\binom{a}{b} \le \left(\frac{ea}{b}\right)^b$ for any $a, b \in \mathbb{N}$; (ii) the inequality $(1-p)^b \ge 1 - bp$ for any integer $b \ge 1$ and real $0 \le p \le 1$; and (iii) we upper-bounded the last factor of (5) by 1.

In the transition from (6) to (7) we lower-bounded s by $x\varphi(x)$. We also used the estimate $\binom{n-x}{m-l} \le \binom{n}{m}\frac{m^l}{(n-x-m)^l}$, which can be verified as follows.

$$\begin{aligned}
\binom{n-x}{m-l} &= \binom{n-x}{m} \prod_{i=0}^{l-1} \frac{m-i}{n-x-m+(i+1)} \\
&\le \binom{n-x}{m} \left(\frac{m}{n-x-m}\right)^l \le \binom{n}{m} \frac{m^l}{(n-x-m)^l}.
\end{aligned}$$

Finally, to obtain (8), we substituted the formula for p and used the fact that

$$l^l(n-x-m)^l = \big(l \cdot (n-x-m)\big)^l \ge \big(l \cdot \frac{n}{2}\big)^l \ge n^l,$$

as $x \le x_1 = o(n)$, $m = \epsilon n \le n/4$ for $\epsilon < \epsilon_0 \le 1/4$ and $l \ge 2$.

Denote $x_2 = \lceil n^{1-\delta} \rceil$. We split the expression (8) into two sums Σ_1 and Σ_2. Let

$$\Sigma_1 := \sum_{x=x_0}^{x_2-1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(\frac{emx}{n}\right)^{l-1} \frac{e^2 m\varphi^{1-2\delta}(n)}{\varphi(x)}\right)^s$$

$$\Sigma_2 := \sum_{x=x_2}^{x_1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(\frac{emx}{n}\right)^{l-1} \frac{e^2 m\varphi^{1-2\delta}(n)}{\varphi(x)}\right)^s$$

These two sums will be bounded separately. We have

$$\Sigma_1 \le \sum_{x=x_0}^{x_2-1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(\frac{emx}{n}\right)^{l-1} \frac{c_1^{1-2\delta} e^2 m n^{d-2d\delta}}{x^{d-2d\delta}\varphi^{2\delta}(x)}\right)^s \tag{9}$$

$$\le \sum_{x=x_0}^{x_2-1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(\frac{emx}{n}\right)^{l-1-d+2d\delta} C m^{d+1-2d\delta}\right)^s \quad \text{(for some } C>0\text{)}$$

$$\le \sum_{x=x_0}^{x_2-1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(n^{-\delta/2}\right)^{l-1-d+2d\delta} C m^{d+1}\right)^s \tag{10}$$

$$\le \sum_{x=x_0}^{x_2-1} x^l \left(\frac{en}{x}\right)^x \left(n^{-\frac{\delta}{2}\cdot 2d\delta} n^{\frac{\delta^2}{2}}\right)^{x\varphi(x)} \le \sum_{x=x_0}^{x_2-1} x^l \left(\frac{en}{x}\right)^x n^{-\frac{x\varphi(x)d\delta^2}{2}} \tag{11}$$

$$\le \sum_{x=x_0}^{x_2-1} n^{2x-\frac{x\varphi(x)d\delta^2}{2}} \le \sum_{x=x_0}^{x_2-1} n^{-2x} \le \frac{n}{n^{2x_0}} = o\left(\frac{1}{m}\right). \tag{12}$$

To obtain (9), we used the property that $\varphi(n) \le \varphi(x)\varphi(n/x) \le c_1\varphi(x)(n/x)^d$, provided that $n, x, n/x$ are sufficiently large. To establish (10), we used the fact that $x \le x_2 = n^{1-\delta}$ and that $em \le ed\log n \le n^{\delta/2}$ (this follows from (4). In the transition to (11), we needed that $l \ge d+1$, $d \ge 1$ and that $Cm^{d+1} \le C(d\log n)^{d+1} = o(n^{\delta^2/2})$, by (4). Then we lower-bounded s by $x\varphi(x)$. To arrive at (12), we used that $l \le x$. The last inequality follows from the facts that x_0 is large enough, so that $\varphi(x) \ge \varphi(x_0) \ge 8/(d\delta^2)$ and that $m = o(n)$.

Next, we turn to bounding Σ_2. First observe that

$$\varphi^{1-2\delta}(n) \le \varphi^{\frac{1-2\delta}{1-\delta}}(n^{1-\delta}) \le \varphi^{\frac{1-2\delta}{1-\delta}}(x) \le \varphi^{1-\delta}(x),$$

where we used the submultiplicativity and monotonicity of the function $\varphi(n)$ and the fact that $x \ge x_2 = n^{1-\delta}$. Substituting the bound for $\varphi^{1-2\delta}(n)$ in Σ_2 and putting $C = e^2 m$, we obtain

$$\Sigma_2 \le \sum_{x=x_2}^{x_1} \sum_{s=\lceil x\varphi(x)\rceil}^{\binom{x}{l}} \left(\frac{en}{x}\right)^x \left(\left(\frac{emx}{n}\right)^{l-1} C\varphi^{-\delta}(x)\right)^s$$

$$\le \sum_{x=x_2}^{x_1} x^l \left(\frac{en}{x}\right)^x \left(\frac{emx}{n} C\varphi^{-\delta}(x)\right)^{x\varphi(x)} \tag{13}$$

$$\le \sum_{x=x_2}^{x_1} \left(\frac{n}{x}\right)^{x-x\varphi(x)} \left(e^{1+x/(x\varphi(x))} m x^{l/(x\varphi(x))} C\varphi^{-\delta}(x)\right)^{x\varphi(x)}$$

$$\leq \sum_{x=x_2}^{x_1} \left(\frac{n}{x}\right)^{x-x\varphi(x)} \left(C'\varphi^{-\delta/2}(x)\right)^{x\varphi(x)} \quad \text{(for some constant } C'>0) \tag{14}$$

$$\leq n\left(\frac{n}{x_1}\right)^{x_2-x_2\varphi(x_2)} \left(C\varphi^{-\delta/2}(x_2)\right)^{x_2\varphi(x_2)} \leq \left(\frac{n}{x_1}\right)^{x_2-x_2\varphi(x_2)} \tag{15}$$

$$= \left(\frac{x_1}{n}\right)^{x_2\varphi(x_2)-x_2} = o(1/m).$$

In the transition to (13), we used that $emx \leq em^2 \leq ed^2\log^2 n < n$ and $l \geq 2$. To get (14), we used that for some constant $c>1$ we have $x^{l/(x\varphi(x))} \leq c^{m/\varphi(x)} \leq c^{\log\varphi(x)/\varphi(x)} = O(1)$ and that $m \leq \varphi^{\delta/2}(x)$ for $x \geq x_0$. To obtain (15), we noticed that $n^{1/(x_2\varphi(x_2))} = O(1)$. At the last equation, we used that $x_1 = o(n)$, $ne/x_1 \to \infty$ as $n\to\infty$ and $x_2\varphi(x_2) - x_2 = \Omega(n^{1-\delta/2})$.

We have shown that for every $l = 2,\ldots,m$, $\Pr[E_l] = o(1/m)$. We conclude that $\Pr[E] \leq \sum_{l=2}^{m}\Pr[E_l] = o(1)$ and, hence, with high probability, the range space $([n],\mathcal{F})$ has shallow-cell complexity φ. ◀

Now we are in a position to prove that with high probability, the range space $([n],\mathcal{F})$ does not admit a small ϵ-net.

▶ **Lemma 15.** *With high probability, the size of any ϵ-net of the range space $([n],\mathcal{F})$ is at least $\frac{(1-4\delta)}{\epsilon}\log\varphi(\frac{1}{\epsilon})$.*

Proof. Assume without loss of generality that $\delta < 1/10$. Denote by μ the probability that the range space has an ϵ-net of size $t = (1-4\delta)\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon}) = (1-4\delta)n$. Then

$$\mu \leq \sum_{\substack{X\subseteq[n]\\|X|=t}} \Pr\left[X \text{ is an } \epsilon\text{-net for } \mathcal{F}\right] \leq \binom{n}{t}(1-p)^{\binom{n-t}{m}} \leq \binom{n}{t}e^{-p\binom{n-t}{m}} \tag{16}$$

$$\leq \left(\frac{en}{t}\right)^t e^{-n\varphi^\delta(n)} \leq 5^n e^{-n\varphi^\delta(n)} = o(1). \tag{17}$$

Here, the crucial transition from (16) to (17) uses the inequality below. Since $1-ax > e^{-bx}$ for $b>a,\ 0<x<1/a-1/b$, we obtain that

$$p\binom{n-t}{m} \geq p\binom{n}{m}\left(\frac{n-m-t}{n-t}\right)^t \geq n\varphi^{1-2\delta}(n)\left(1-\frac{m}{n-t}\right)^t$$

$$\geq n\varphi^{1-2\delta}(n)\left(1-\frac{(1+\delta/2)m}{n}\right)^t \geq n\varphi^{1-2\delta}(n)e^{-\frac{(1+\delta)mt}{n}}$$

$$\geq n\varphi^{1-2\delta}(n)e^{-(1-3\delta)\log\varphi(\frac{1}{\epsilon})} \geq n\varphi^{1-2\delta}(n)\varphi^{-1+3\delta}\left(\frac{1}{\epsilon}\right) \geq n\varphi^\delta(n).$$ ◀

Thus, Lemma 14 and Lemma 15 imply that with high probability the range space $([n],\mathcal{F})$ has shallow-cell complexity φ and it admits no ϵ-net of size less than $(1-4\delta)\frac{1}{\epsilon}\log\varphi(\frac{1}{\epsilon})$. This completes the proof of the theorem. ◀

Acknowledgements. We thank the anonymous reviewers for feedback that improved the content and style of this paper, and for suggesting a slightly weaker but simpler proof of Lemma 4.

References

1. P. K. Agarwal, J. Pach, and M. Sharir. State of the union (of geometric objects): A review. In J. Goodman, J. Pach, and R. Pollack, editors, *Computational Geometry: Twenty Years Later*, pages 9–48. American Mathematical Society, 2008.
2. N. Alon. A non-linear lower bound for planar epsilon-nets. *Discrete & Computational Geometry*, 47(2):235–244, 2012.
3. B. Aronov, E. Ezra, and M. Sharir. Small-size ϵ-nets for axis-parallel rectangles and boxes. *SIAM J. Comput.*, 39(7):3248–3282, 2010.
4. P. Ashok, U. Azmi, and S. Govindarajan. Small strong epsilon nets. *Comput. Geom.*, 47(9):899–909, 2014.
5. N. Bus, S. Garg, N. H. Mustafa, and S. Ray. Tighter estimates for epsilon-nets for disks. *Comput. Geom.*, 53:27–35, 2016.
6. T. M. Chan, E. Grant, J. Könemann, and M. Sharpe. Weighted capacitated, priority, and geometric set cover via improved quasi-uniform sampling. In *Proceedings of Symposium on Discrete Algorithms (SODA)*, 2012.
7. B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.
8. K. Clarkson and K. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37:43–58, 2007.
9. K. L. Clarkson and P. W. Shor. Application of random sampling in computational geometry, II. *Discrete & Computational Geometry*, 4:387–421, 1989.
10. D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.
11. J. Komlós, J. Pach, and G. J. Woeginger. Almost tight bounds for epsilon-nets. *Discrete & Computational Geometry*, 7:163–173, 1992.
12. J. Matoušek. On constants for cuttings in the plane. *Discrete & Computational Geometry*, 20(4):427–448, 1998.
13. J. Matoušek. *Lectures in Discrete Geometry*. Springer-Verlag, New York, NY, 2002.
14. J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: Small epsilon-nets for disks and halfspaces. In *Proceedings of Symposium on Computational Geometry*, pages 16–22, 1990.
15. N. H. Mustafa, K. Dutta, and A. Ghosh. A simple proof of optimal epsilon-nets. *Submitted*, 2016.
16. N. H. Mustafa and S. Ray. Near-optimal generalisations of a theorem of Macbeath. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 578–589, 2014.
17. N. H. Mustafa and K. Varadarajan. Epsilon-approximations and epsilon-nets. In J. E. Goodman, J. O'Rourke, and C. D. Tóth, editors, *Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2016, to appear.
18. J. Pach and P. K. Agarwal. *Combinatorial Geometry*. John Wiley & Sons, New York, NY, 1995.
19. J. Pach and G. Tardos. Tight lower bounds for the size of epsilon-nets. *Journal of the AMS*, 26:645–658, 2013.
20. E. Pyrga and S. Ray. New existence proofs for epsilon-nets. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 199–207, 2008.
21. N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13:145–147, 1972.
22. Micha Sharir. On k-sets in arrangement of curves and surfaces. *Discrete & Computational Geometry*, 6:593–613, 1991.

23 S. Shelah. A combinatorial problem; stability and order for models and theories in infinitary languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

24 V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

25 K. Varadarajan. Epsilon nets and union complexity. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 11–16, 2009.

26 K. Varadarajan. Weighted geometric set cover via quasi uniform sampling. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 641–648, New York, USA, 2010. ACM.

On Computing the Fréchet Distance Between Surfaces

Amir Nayyeri[1] **and Hanzhong Xu**[2]

1 **Oregon State University, EECS Department, USA**
nayyeria@eecs.oregonstate.edu
2 **Oregon State University, EECS Department, USA**
xuhanz@oregonstate.edu

Abstract

We describe two $(1+\varepsilon)$-approximation algorithms for computing the Fréchet distance between two homeomorphic piecewise linear surfaces $\mathcal{R}$ and $\mathcal{S}$ of genus zero and total complexity n, with Fréchet distance δ.

1. A $2^{O\left(\left(n+\frac{\text{Area}(\mathcal{R})+\text{Area}(\mathcal{S})}{(\varepsilon\delta)^2}\right)^2\right)}$ time algorithm if $\mathcal{R}$ and $\mathcal{S}$ are composed of fat triangles (triangles with angles larger than a constant).
2. An $O(D/(\varepsilon\delta)^2)\cdot n + 2^{O(D^4/(\varepsilon^4\delta^2))}$ time algorithm if $\mathcal{R}$ and $\mathcal{S}$ are polyhedral terrains over $[0,1]^2$ with slope at most D.

Although, the Fréchet distance between curves has been studied extensively, very little is known for surfaces. Our results are the first algorithms (both for surfaces and terrains) that are guaranteed to terminate in finite time. Our latter result, in particular, implies a linear time algorithm for terrains of constant maximum slope and constant Fréchet distance.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Surfaces, terrains, Fréchet distance, normal coordinates, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.55

1 Introduction

Measuring similarity between geometric objects is a fundamental problem that appears in several applications in computer science (see, for example, references [27, 16]). A natural way of estimating such a similarity is formalized via the notion of the Fréchet distance. The Fréchet distance takes into account the endowed topology of the objects, which makes it a more desirable choice compared to other classical measures like the Hausdorff distance for many applications.

The Fréchet distance between curves can be computed efficiently [2], and it has been successfully used in many applications [20, 25, 21, 5, 28, 6, 7]. Surprisingly though, computing the Fréchet distance becomes much harder for higher dimensional objects, even surfaces. It is NP-hard to compute the Fréchet distance between a triangle and a (self-crossing) surface [17], between two terrains [8], and between two polygons with (the same number of) holes [8]. Furthermore, to the best of our knowledge, no exact or approximation algorithm exists for computing the Fréchet distance between two surfaces, even two terrains, that is guaranteed to terminate in finite time.

In this paper, we study the problem of computing the Fréchet distance between surfaces of genus zero (punctured spheres). Our input is composed of two homeomorphic piecewise


© Amir Nayyeri and Hanzhong Xu;
licensed under Creative Commons License CC-BY

linear orientable surfaces, $\mathcal{R}$ and $\mathcal{S}$, each constructed from a set of Euclidean triangles by identifying pairs of equal-length edges. The input also includes immersions $\varphi_{\mathcal{R}} : \mathcal{R} \to \mathbb{R}^3$ and $\varphi_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}^3$ that are isometric on each triangle of $\mathcal{R}$ and $\mathcal{S}$, respectively. The Fréchet length of a homeomorphism $\sigma : \mathcal{R} \to \mathcal{S}$, is defined to be $\delta_F(\sigma) = \max_{x \in \mathcal{R}} ||\varphi_{\mathcal{R}}(x) - \varphi_{\mathcal{S}}(\sigma(x))||_2$. The Fréchet distance between (immersed) $\mathcal{R}$ and $\mathcal{S}$ is defined to be $\delta_F(\mathcal{R}, \mathcal{S}) = \inf_\sigma \delta_F(\sigma)$, where σ ranges over all homeomorphisms between $\mathcal{R}$ and $\mathcal{S}$. Note that $\delta_F(\mathcal{R}, \mathcal{S})$ depends on the immersions $\varphi_{\mathcal{R}}$ and $\varphi_{\mathcal{S}}$, which are not explicitly mentioned here and throughout the paper to simplify the notation.

1.1 Previous work

The Fréchet distance and its variants between curves have been extensively studied [3, 13, 12, 19, 4, 11], resulting in very efficient exact or approximation algorithms. In contrast, very little is known about computing the Fréchet distance between surfaces. Alt and Buchin [1] show that the Fréchet distance between triangulated surfaces is uppersemicomputable: there is an algorithm that generates an infinite sequence of real numbers that converges to the Fréchet distance. Buchin et al. [9] describe a polynomial time exact algorithm for computing the Fréchet distance between simple polygons. Their work is generalized to exact polynomial time algorithms for folded polygons and polygons with constant numbers of holes by Cook et al. [10], and Nayyeri and Sidiropoulos [22], respectively. To our knowledge, there is no exact or approximation algorithm that is guaranteed to terminate in finite time for surfaces or even polyhedral terrains.

1.2 Our contribution

In this paper, we describe $(1+\varepsilon)$-approximation algorithms for computing the Fréchet distance between two triangulated surfaces $\mathcal{R}$ and $\mathcal{S}$ (composed of fat triangles) immersed in $\mathbb{R}^3$.

▶ **Theorem 1.** *Let $\mathcal{R} = (\mathcal{R}_V, \mathcal{R}_E, \mathcal{R}_T)$ and $\mathcal{S} = (\mathcal{S}_V, \mathcal{S}_E, \mathcal{S}_T)$ be two triangulated surfaces composed of fat triangles, let $n = |\mathcal{R}_V| + |\mathcal{S}_V|$, and let $\varepsilon > 0$. There exists a $(1+\varepsilon)$-approximation algorithm for computing the Fréchet distance between $\mathcal{R}$ and $\mathcal{S}$ with running time*

$$2^{O\left(\left(|\mathcal{R}_V|+|\mathcal{S}_V|+\frac{Area(\mathcal{R})+Area(\mathcal{S})}{(\varepsilon\delta)^2}\right)^2\right)}.$$

Note that $\frac{\text{Area}(\mathcal{R})+\text{Area}(\mathcal{S})}{\delta^2}$ is invariant up to scaling, therefore, scaling the surfaces does not change the running time of our algorithm. We view this result as a first step towards designing efficient approximation algorithms for computing the Fréchet distance between surfaces.

We describe a significantly more efficient algorithm if $\mathcal{R}$ and $\mathcal{S}$ are polyhedral terrains over $[0,1]^2$. This algorithm works in linear time for surfaces of constant maximum slope with constant Fréchet distance.

▶ **Theorem 2.** *Let $\mathcal{R}$ and $\mathcal{S}$ be polyhedral terrains over $[0,1]^2$ of maximum slope D, and let $n = |\mathcal{R}_V| + |\mathcal{S}_V|$. There exists a $(1+\varepsilon)$-approximation algorithm for computing the Fréchet distance between $\mathcal{R}$ and $\mathcal{S}$ with running time*

$$O(D/(\varepsilon\delta)^2) \cdot n + 2^{O(D^4/(\varepsilon^4\delta^2))}.$$

1.3 Overview

First, we consider the problem for piecewise linear triangulations $\mathcal{R} = (\mathcal{R}_V, \mathcal{R}_E, \mathcal{R}_T)$ and $\mathcal{S} = (\mathcal{S}_V, \mathcal{S}_E, \mathcal{S}_T)$, composed of triangles of diameter at most r. We design an algorithm that, for each $\delta > 0$, returns an $(\mathcal{R}, \mathcal{S})$-homeomorphism of Fréchet length $\delta + O(r)$ if $\delta \geq \delta_F(\mathcal{R}, \mathcal{S})$. If $\delta < \delta_F(\mathcal{R}, \mathcal{S})$, our algorithm either returns an $(\mathcal{R}, \mathcal{S})$-homeomorphism of Fréchet length $\delta + O(r)$, or it correctly decides that the $\delta_F(\mathcal{R}, \mathcal{S}) > \delta$. We use binary search with this algorithm to approximate $\delta_F(\mathcal{R}, \mathcal{S})$.

Consider a homeomorphism $g : \mathcal{R} \to \mathcal{S}$ of Fréchet length at most $\delta + r$. As the first step, our algorithm computes f_0, an approximation of $g|_{\mathcal{R}_V \cup g^{-1}(\mathcal{S}_V)}$ (the restriction of g into $\mathcal{R}_V \cup g^{-1}(\mathcal{S}_V)$). Precisely, for each vertex $u \in \mathcal{R}_V$ (resp. each vertex $v \in \mathcal{S}_V$), $f_0(u)$ and $g(u)$ (resp. $f_0^{-1}(v)$ and $g^{-1}(v)$) are in the same triangle of $\mathcal{S}_T$ (resp. $\mathcal{R}_T$). To compute $f_0(u)$, for each $u \in \mathcal{R}_V$ (resp. for each $v \in \mathcal{S}_V$), our algorithm enumerates over all possible triangles that can contain $g(u)$ (resp. $g^{-1}(v)$): triangles of $\mathcal{S}_T$ (resp. $\mathcal{R}_T$) whose distance from u (resp. v) is at most δ. Our algorithm refines $\mathcal{R}$ and $\mathcal{S}$ with regards to f_0. Let $\widetilde{\mathcal{R}}_V = \mathcal{R}_V \cup f_0^{-1}(\mathcal{S}_V)$, and $\widetilde{\mathcal{S}}_V = \mathcal{S}_V \cup f_0(\mathcal{R}_V)$. Then, let $\widetilde{\mathcal{R}} = (\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E, \widetilde{\mathcal{R}}_T)$ and $\widetilde{\mathcal{S}} = (\widetilde{\mathcal{S}}_V, \widetilde{\mathcal{S}}_E, \widetilde{\mathcal{S}}_T)$ be arbitrary refinements of $\mathcal{R}$ and $\mathcal{S}$, respectively.

Our algorithm seeks to extend f_0 to a homeomorphim $f : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$, with Fréchet length at most $\delta + O(r)$. Let the homeomorphism $h : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$ be an extension of f_0 of Fréchet length $\delta + O(r)$ (we show such a homeomorphism exists). So, $h_1 = h|_{\widetilde{\mathcal{R}}_E}$ is a one-to-one continuous map with the following properties: (1) $f_0 = h_1|_{\widetilde{\mathcal{R}}_V}$, (2) h_1 maps the boundary vertices to boundary vertices, and boundary edges to boundary edges, (3) h_1 preserves the cyclic order of edges around each vertex (the combinatorial embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$), and (4) $\delta_F(h_1) = \delta + O(r)$.

Moreover, we observe that any map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ with properties (1) to (4) can be extended to a homeomorphism between $\widetilde{\mathcal{R}}$ and $\widetilde{\mathcal{S}}$ of Fréchet length $\delta + O(r)$. As a result, our problem of extending f_0 to $(\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$-homeomorphism f reduces to finding an f_1 with properties (1) to (4), or equivalently finding the image of each edge $e \in \widetilde{\mathcal{R}}_E$ under f_1.

Let $e \in \widetilde{\mathcal{R}}_E$, and let $\gamma = h_1(e)$, so, $\delta_F(e, \gamma) \leq \delta_F(h_1) \leq \delta + O(r)$. The intersection of γ with any triangle $t \in \widetilde{\mathcal{S}}_T$ is a collection of paths $\{\gamma_1, \ldots, \gamma_k\}$. As the diameter of t is at most r, modifying γ_i's inside t can only increase the Fréchet length by $O(r)$. Since we allow an $O(r)$ additive approximation factor, we can overlook the exact positions of γ_i's in t, and focus only on their intersection pattern. When the intersection patterns in all triangles of $\widetilde{\mathcal{S}}_T$ are viewed as a whole, they specify the sequence of edges in $\widetilde{\mathcal{S}}_E$ that γ crosses, that is the homotopy class of γ in $\widetilde{\mathcal{S}} \backslash \widetilde{\mathcal{S}}_V$.

Potentially, the image of an edge $e \in \widetilde{\mathcal{R}}_E$ may belong to infinitely many homotopy classes (in $\widetilde{\mathcal{S}} \backslash \widetilde{\mathcal{S}}_V$). We bound the number of possible homotopy classes, by considering the interaction of $h_1(\widetilde{\mathcal{R}}_E)$ with $\widetilde{\mathcal{S}}_E$. We view $h_1(\widetilde{\mathcal{R}}_E)$ as an embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$ on $\widetilde{\mathcal{S}}$ that is combinatorially equivalent with its triangulated embedding on $\widetilde{\mathcal{R}}$. We show that if an edge $s \in \widetilde{\mathcal{S}}_E$ is crossed a sufficiently large number of times by $h_1(\widetilde{\mathcal{R}}_E)$, then we can shortcut the curves in $h_1(\widetilde{\mathcal{R}}_E)$ along s to obtain a different embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$ on $\widetilde{\mathcal{S}}$. Following Cook et al. [10], we observe that such shortcutting operations decrease the crossing number on s, but do not increase the Fréchet distance. We conclude the existence of an $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ of Fréchet length $\delta + O(r)$ such that each edge $s \in \widetilde{\mathcal{S}}_E$ is crossed by $f_1(\widetilde{\mathcal{R}}_E)$ a bounded number of times.

Our algorithm uses normal coordinates to enumerate the set of homotopy classes for the curves of $f_1(\widetilde{\mathcal{R}}_E)$ in $\widetilde{\mathcal{S}} \backslash \widetilde{\mathcal{S}}_V$. Normal coordinates record, for each edge $s \in \widetilde{\mathcal{S}}_T$, the number of times it is crossed by $f_1(\widetilde{\mathcal{R}}_E)$. Our crossing bound implies a bound on the maximum coordinate of any set of normal coordinates that must be considered, thus, a bound on all possible normal coordinates.

Provided f_0 and a set of normal coordinates, our algorithm constructs an $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ of Fréchet length $\delta + O(r)$ via constructing the images of all edges. Finally, our algorithm extends f_1 to include the interior of triangles, $\widetilde{\mathcal{R}}_T$, to obtain $f : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$ of Fréchet length at most $\delta + O(r)$. The running time of our algorithm depends on the complexity of $\mathcal{R}$ and $\mathcal{S}$, as well as their vertex densities: the maximum number of their vertices in any ball of radius $\max(\delta, r)$.

To obtain our $(1+\varepsilon)$-approximation algorithm for two general triangulations $\mathcal{R}$ and $\mathcal{S}$, our algorithm refines them into triangulations composed of triangles of diameter $O(\varepsilon\delta)$. Then, the above algorithm can be applied to find a homeomorphism of Fréchet length $\delta + O(\varepsilon\delta)$. If $\mathcal{R}$ and $\mathcal{S}$ are terrains with slope at most D, our algorithms can find triangulations $\mathcal{R}'$ and $\mathcal{S}'$ within the Fréchet distance $\varepsilon\delta$ of $\mathcal{R}$ and $\mathcal{S}$, respectively. The complexity of $\mathcal{R}'$ and $\mathcal{S}'$, and their vertex densities are bounded by functions of ε, δ and D, therefore, we obtain a linear time $(1+\varepsilon)$-approximation algorithm if all these parameters are constant.

2 Preliminaries

Maps

Let $f : A \to B$ be a function, and let $U \subseteq A$. We define $f(U)$ to be $\{f(u) | u \in A\}$. The function $f|_U : U \to B$, the **restriction** of f to the subset U, is defined as for all $u \in U$, $f|_U(u) = f(u)$. In this case, we also say, that f is an **extension** of $f|_U$ to A. For topological spaces A and B, f is a **homeomorphism** if (1) it is a bijection, (2) it is continuous, and (3) its inverse is continuous.

Surfaces

A **surface** $\mathcal{Q}$ (or a 2-manifold) is a space, in which every point has a neighborhood that is homeomorphic to the plane or half-plane. An **embedding** $\Phi : \mathcal{Q} \to \mathbb{R}^3$ is a continuous one-to-one map. An **immersion** $\varphi : \mathcal{Q} \to \mathbb{R}^3$ is a continuous map, such that for any $x \in \mathcal{Q}$ there is a neighborhood N_x of x, on which f is an embedding.

A **piecewise linear surface** is a surface $\mathcal{Q}$ that is constructed from a set of Euclidean polygons by identifying pairs of equal-length edges. In this paper, we assume that all the constituent polygons are triangles. We denote the constituent vertices, edges, and triangles of $\mathcal{Q}$ by $\mathcal{Q}_V$, $\mathcal{Q}_E$, and $\mathcal{Q}_T$, in order, thus, we write $\mathcal{Q} = (\mathcal{Q}_V, \mathcal{Q}_E, \mathcal{Q}_T)$. A **locally isometric immersion** is an immersion $\varphi : \mathcal{Q} \to \mathbb{R}^3$ such that for each $t \in \mathcal{Q}_T$, $\varphi|_t$ is an isometric map. In particular, t and $\varphi(t)$ are congruent triangles.

Embedded curves and graphs

Let $\mathcal{Q}$ be a surface and let $\alpha, \beta : [0,1] \to \mathcal{Q}$ be curves embedded on $\mathcal{Q}$ with the same endpoints, $\alpha(0) = \beta(0)$ and $\alpha(1) = \beta(1)$. A **homotopy** $h : [0,1] \times [0,1] \to \mathcal{Q}$ is a continuous map such that $h[x,0] = \alpha$, $h[x,1] = \beta$, $h[0,t] = \alpha(0) = \beta(0)$, and $h[1,t] = \alpha(1) = \beta(1)$.

An **embedding** of a graph $G = (V, E)$ into a surface $\mathcal{Q}$ is a continuous function that maps vertices in V into distinct points in $\mathcal{Q}$ and edges in E into disjoint paths except for their endpoints. The **faces** of the embedding are maximal subsets of $\mathcal{Q}$ that are disjoint from the image of the graph. An embedding is **cellular** if all its faces are topological disks. In particular, each boundary component in a cellular embedding is covered by the image of the graph. A cellular embedding on an orientable surface can be described by a **rotation system**. A rotation system is composed of a cyclic (clockwise) order of edges around vertices. A rotation system is a combinatorial description of the embedding of a graph. Equivalent

rotation systems of G on two different surfaces $\mathcal{Q}$ and $\mathcal{Q}'$ induce a one-to-one correspondence between the vertices, edges, and faces of the different embeddings. Therefore, they can be extended to a homeomorphism between $\mathcal{Q}$ and $\mathcal{Q}'$. Note that the homotopy class of an edge might be completely different in two combinatorially equivalent embeddings. For example, one can apply Dehn Twists on a cycle that avoids vertices of the embedding to change the homotopy class of edges without affecting the rotation system.

Fréchet distance

Let $\mathcal{R}$ and $\mathcal{S}$ be homeomorphic piecewise linear triangulations, and let $\varphi_{\mathcal{R}} : \mathcal{R} \to \mathbb{R}^3$ and $\varphi_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}^3$ be locally isometric immersions. The Fréchet length of a homeomorphism $\sigma : \mathcal{R} \to \mathcal{S}$, is defined to be $\delta_F(\sigma) = \max_{x \in \mathcal{R}} ||\varphi_{\mathcal{R}}(x) - \varphi_{\mathcal{S}}(\sigma(x))||_2$. The Fréchet distance between (immersed) $\mathcal{R}$ and $\mathcal{S}$ is defined to be $\delta_F(\mathcal{R}, \mathcal{S}) = \inf_\sigma \delta_F(\sigma)$, where σ ranges over all homeomorphisms between $\mathcal{R}$ and $\mathcal{S}$. Note that $\delta_F(\mathcal{R}, \mathcal{S})$ depends on the immersions $\varphi_{\mathcal{R}}$ and $\varphi_{\mathcal{S}}$, which are not explicitly mentioned here and throughout the paper to simplify the notation. Also, note that, a homeomorphism that realizes $\delta_F(\mathcal{R}, \mathcal{S})$ does not necessarily exist as the Fréchet distance is defined as an infimum.

Neighborhoods

Given $U \subset \mathbb{R}^3$, and $r \in \mathbb{R}_{\geq 0}$, we define the set $B_r(U)$ to be composed of all the points in $\mathbb{R}^3$ that are not farther than r from U. In particular, for a $p \in \mathbb{R}^3$, $B_r(p)$ is a closed ball of radius r, and for a curve $\gamma \subseteq \mathbb{R}^3$, $B_r(\gamma)$ is the union of balls of radius r centered on all points of γ. Let $\mathcal{Q} = (\mathcal{Q}_V, \mathcal{Q}_E, \mathcal{Q}_T)$ be a triangulated surface. We use $U \cap \mathcal{Q}_V$ to denote the set of vertices inside U, $U \cap \mathcal{Q}_E$ to denote the set of edges of $\mathcal{Q}_E$ that intersect U, and $U \cap \mathcal{Q}_T$ to denote the set of triangles of $\mathcal{Q}_T$ that intersect U.

3 Fréchet distance between fine triangulaions

In this section, let $\mathcal{R}$ and $\mathcal{S}$ be two triangulated surfaces (of genus zero) composed of triangles with diameter at most r. Also, let $\varphi_{\mathcal{R}} : \mathcal{R} \to \mathbb{R}^3$ and $\varphi_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}^3$ be immersions. We consider the Fréchet length and Fréchet distance with respect to $\varphi_{\mathcal{R}}$ and $\varphi_{\mathcal{S}}$, but to simplify the exposition we do not explicitly mention them when it is clear from the context. We describe an algorithm that, given $\delta > 0$, returns an $(\mathcal{R}, \mathcal{S})$-homeomorphism of Fréchet length $\delta + O(r)$ if $\delta \geq \delta_F(\mathcal{R}, \mathcal{S})$. If $\delta < \delta_F(\mathcal{R}, \mathcal{S})$, our algorithm either returns a homeomorphism of Fréchet length $\delta + O(r)$ or it correctly decides that $\delta < \delta_F(\mathcal{R}, \mathcal{S})$.

3.1 Mapping vertices

A bijection $f_0 : \widetilde{\mathcal{R}}_V \to \widetilde{\mathcal{S}}_V$ is an **approximate vertex map** if and only if it has the following properties.

1. $\widetilde{\mathcal{R}}_V = \mathcal{R}_V \cup \mathcal{R}'_V$, $\widetilde{\mathcal{S}}_V = \mathcal{S}_V \cup \mathcal{S}'_V$, $f_0(\mathcal{R}_V) = \mathcal{S}'_V$, and $f_0(\mathcal{R}'_V) = \mathcal{S}_V$.
2. f_0 maps boundary vertices to boundary vertices, and it preserves the cyclic order of boundary vertices on each boundary component.
3. There exists an $f : \mathcal{R} \to \mathcal{S}$ of Fréchet length at most $\delta + r$ such that
   (a) for each $u \in \mathcal{R}_V$, $f(u)$ and $f_0(u)$ are in the same triangle of $\mathcal{S}_T$, and
   (b) for each $v \in \mathcal{S}_V$, $f^{-1}(v)$ and $f_0^{-1}(v)$ are in the same triangle of $\mathcal{R}_T$.
   We say that f_0 **agrees** with f.

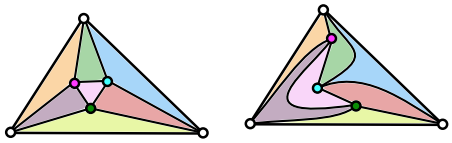


Figure 1 H and H'; corresponding faces and vertices have the same colors.

We show that an approximate vertex map can be extended to an $(\mathcal{R}, \mathcal{S})$-homeomorphism with Fréchet length $\delta + O(r)$. To that end, we find the following lemma helpful.

▶ **Lemma 3.** *Let t be a triangle with diameter r, and let $P, P' \subseteq int(t)$ be finite point sets with the same cardinality. Also, let $g : P \to P'$ be a bijection. There exists a homeomorphism $h : t \to t$ such that*

1. *$h|_{\partial(t)}$ is the identity map.*
2. *$h|_P = g$.*
3. *$\delta_F(h) \leq r$.*

Proof. Let (x, y, z) be vertices of t. Let $g' : \{x, y, z\} \cup P \to \{x, y, z\} \cup P'$ be a bijection that is the identity map for $\{x, y, z\}$ and g for P. Let H be a triangulation (a plane graph) with vertex set $\{x, y, z\} \cup P$. Let H' be a graph with vertex set $V' = \{x, y, z\} \cup P'$, where $v, v' \in V'$ are adjacent if any only if there corresponding vertices via g' are adjacent in H. The isomorphism between H and H' naturally gives rise to a combinatorial embedding of H' that is equivalent to the embedding of H. The isomorphism between H and H' and their equivalent embedding provides bijections between vertex sets, edge sets, and face sets of H and H'. Let h be any homeomorphism that respects these bijections and that is identity on the boundary. By the construction, h has properties (1) and (2). Additionally, $\delta_F(h) \leq r$, as h maps points within t that has diameter r. ◀

▶ **Lemma 4.** *Let $\mathcal{R} = (\mathcal{R}_V, \mathcal{R}_E, \mathcal{R}_T)$ and $\mathcal{S} = (\mathcal{S}_V, \mathcal{S}_E, \mathcal{S}_T)$ be triangulated surfaces composed of triangles of diameter at most r. Any approximate vertex map $f_0 : \mathcal{R} \to \mathcal{S}$ can be extended to a $(\mathcal{R}, \mathcal{S})$-homeomorphism of Fréchet length $\delta + 3r$.*

Proof. Let $g : \mathcal{R} \to \mathcal{S}$ be a homeomorphism of Fréchet length $\delta + r$ that agrees with f_0. We construct homeomorphisms $g' : \mathcal{R} \to \mathcal{R}$ and $g'' : \mathcal{S} \to \mathcal{S}$, each of Fréchet length at most r, such that $g'' \circ g \circ g'$ is an extension of f_0.

We construct g' in three steps as follows. (1) For each $v \in \mathcal{R}_V$, we define $g'(v) = v$. (2) For each $e \in \mathcal{R}_E$, if e is not a boundary edge we define $g'|_e$ to be the identity map, otherwise $g'|_e$ is any homeomorphism, in which for any $p \in f_0^{-1}(\mathcal{S}_V) \cap e$ we have $g'|_e(p) = g^{-1}(f_0(p))$. Such a homeomorphism exists because both f_0 and g preserve the order of boundary edges along boundary cycles. (3) For each $t \in \mathcal{R}_T$, we define $g'|_t$ to be the extension of $g'|_{\partial(t)}$ to t such that for any $p \in f_0^{-1}(\mathcal{S}_V) \cap t$ we have $g'|_t(p) = g^{-1}(f_0(p))$. Lemma 3 implies that such an extension exists.

The construction of g'' is very similar. (1) For each $v \in \mathcal{S}_V$, we define $g''(v) = v$. (2) For each $e \in \mathcal{S}_E$, if e is not a boundary edge we define $g''|_e$ to be the identity map, otherwise $g''|_e$ is any homeomorphism, in which for any $p \in f_0(\mathcal{R}_V) \cap e$ we have $g''|_e(p) = g(f_0^{-1}(p))$. (3) For each $t \in \mathcal{S}_T$, we define $g''|_t$ to be the extension of $g''|_{\partial(t)}$ to t such that for any $p \in f_0(\mathcal{R}_V) \cap t$ we have $g''|_t(p) = g(f_0^{-1}(p))$. Lemma 3 implies that such an extension exists. ◀

Our algorithm computes a set of candidates for f_0 via guessing the triangles that contain the images and preimages of $\mathcal{R}_V$ and $\mathcal{S}_V$, respectively, under a homeomorphism of Fréchet length at most $\delta + r$.

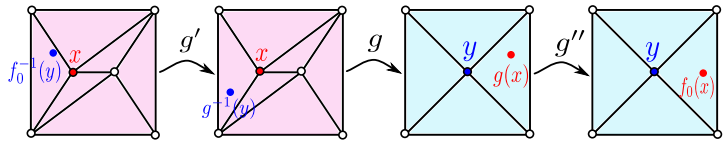


Figure 2 Proof of Lemma 4; $f_0 = g'' \circ g \circ g'$ on vertices $x \in \mathcal{R}_V$ (red) and $y \in \mathcal{S}_V$ (blue).

▶ **Lemma 5.** *Let $\mathcal{R}$ and $\mathcal{S}$ be triangulated surfaces composed of triangles of diameter at most r. There exists a set F_0 of size*

$$\prod_{u \in \mathcal{R}_V} |B_{\delta+r}(u) \cap \mathcal{S}_T| \cdot \prod_{v \in \mathcal{S}_V} |B_{\delta+r}(v) \cap \mathcal{R}_T|$$

that contains an approximate vertex map. ($B_{\delta+r}(u) \cap \mathcal{S}_T$ denotes the set of triangles of $\mathcal{S}_T$ that intersect $B_{\delta+r}(u)$, and $B_{\delta+r}(v) \cap \mathcal{R}_T$ denotes the set of triangles of $\mathcal{R}_T$ that intersect $B_{\delta+r}(v)$.)

Proof. Let f_0 be an approximate vertex map, and let f be the homeomorphism of Fréchet length $\delta + r$ that agrees with f_0. For each $u \in \mathcal{R}_V$, we need to guess the triangle $t \in \mathcal{S}_T$ that contains $f_0(u)$ or equivalently $f(u)$. Since $||f(u) - u|| \leq \delta + r$, t should intersect $B_{\delta+r}(u)$, thus, there are $|B_{\delta+r}(u) \cap \mathcal{S}_T|$ choices for the triangle that contains $f_0(u)$. Similarly, for each $v \in \mathcal{S}_V$, there are $|B_{\delta+r}(v) \cap \mathcal{R}_T|$ choices for the triangle that contains $f_0^{-1}(v)$. ◀

3.2 Mapping edges

Let $f_0 : \widetilde{\mathcal{R}}_V \to \widetilde{\mathcal{S}}_V$ be an approximate vertex map. Let $\widetilde{\mathcal{R}} = (\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E, \widetilde{\mathcal{R}}_T)$ and $\widetilde{\mathcal{S}} = (\widetilde{\mathcal{S}}_V, \widetilde{\mathcal{S}}_E, \widetilde{\mathcal{S}}_T)$ be refinements of $\mathcal{R}$ and $\mathcal{S}$, respectively.

A **scaffold map** is a continuous one-to-one map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ with the following properties.

1. $f_1(\widetilde{\mathcal{R}}_V) = \widetilde{\mathcal{S}}_V$.
2. f_1 is a cellular embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$ on $\widetilde{\mathcal{S}}$.
3. f_1 maps boundary edges to boundary edges (so, it preserves the cyclic order of boundary edges around boundary components).
4. f_1 preserves the cyclic order of edges around each vertex: for any $u \in \widetilde{\mathcal{R}}_V$ with neighbors $\{w_1, \ldots, w_k\}$, the cyclic order of the edges $\{(u, w_1), (u, w_2), \ldots, (u, w_k)\}$ around u is identical to the cyclic order of curves $\{f_1(u, w_1), \ldots, f_1(u, w_k)\}$ around $f_1(u)$.
5. For each $e \in \widetilde{\mathcal{R}}_E$ and each $t \in \widetilde{\mathcal{S}}_T$, $f_1(e) \cap t$ is a collection of straight line segments that intersect $\partial(t)$ at their endpoints.

3.2.1 Sufficiency of scaffold maps

We show that a scaffold map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ can be extended to a $(\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$-homeomorphism of Fréchet length arbitrarily close to $\delta_F(f_1)$. As Properties (2) to (4) of a scaffold map f_1 imply, $f_1(\widetilde{\mathcal{R}}_E)$ is an embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$ on the surface $\widetilde{\mathcal{S}}$ that is combinatorially equivalent to the embedding of $(\widetilde{\mathcal{R}}_V, \widetilde{\mathcal{R}}_E)$ on $\widetilde{\mathcal{R}}$. In particular, f_1 gives a one-to-one correspondence between the triangles in $\widetilde{\mathcal{R}}_T$ and the faces of $f_1(\widetilde{\mathcal{R}}_E)$. Property (5) of a scaffold map implies that each triangle corresponds to a **folded polygon**: a piecewise linear triangulated (sub-)surface (of $\widetilde{\mathcal{S}}$) whose interior is disjoint from $\widetilde{\mathcal{S}}_V$.

To extend f_1 to a homeomorphism, for each triangle $t \in \widetilde{\mathcal{R}}_T$ and its corresponding folded polygon p_t with boundary $f_1(\partial(t))$, we extend $f_1|_{\partial(t)}$ to a (t, p_t)-homeomorphism. The proof

of Lemma 1 of Cook et al. [10] actually proves the following stronger statement, which facilitates the extension of $f_1|_{\partial(t)}$.

▶ **Lemma 6** (Cook et al. [10]). *Let t be a triangle, p be a folded polygon with n triangles, and $g : \partial(t) \to \partial(p)$ a homeomorphism. For any $\epsilon > 0$, the map g can be extended to a homeomorphism, $h : t \to p$, for which $\delta_F(h) \leq \delta_F(g) + \epsilon$, in polynomial time in n.*

The one-to-one correspondence of triangles to folded polygons and Lemma 6 imply that a scaffold map can be extended to an $(\widetilde{\mathcal{R}}, \widetilde{\mathcal{S}})$-homeomorphism of arbitrary close Fréchet length.

▶ **Lemma 7.** *Let $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ be a scaffold map that maps each triangle to a folded polygon composed of at most m triangles. For any $\epsilon > 0$, the map f_1 can be extended to a homeomorphism $f : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$, for which $\delta_F(f) \leq \delta_F(f_1) + \epsilon$, in $O(m|\widetilde{\mathcal{R}}_V + \widetilde{\mathcal{S}}_V|)$ time.*

In light of Lemma 7 we focus on computing a scaffold map with Fréchet length $\delta + O(r)$.

3.2.2 Crossing bounds

We show that any approximate vertex map can be extended to a scaffold map of Fréchet length $\delta + O(r)$, whose image intersects each edge $s \in \widetilde{\mathcal{S}}$ a bounded number of times. (Note that, a priori, these intersection numbers can be arbitrarily large, giving rise to infinitely many scaffold maps.) To this end, we consider a homeomorphism $h : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$ of Fréchet length $\delta + O(r)$ that is an extension of an approximate vertex map, which exists by Lemma 4. We modify $h_1 = h|_{\widetilde{\mathcal{R}}_E}$ via a sequence of shortcutting operations (defined below) to obtain a scaffold map of Fréchet length $\delta + O(r)$, whose image intersects each edge of $\widetilde{\mathcal{S}}$ a bounded number of times.

Let $\alpha : [0,1] \to \mathbb{R}^d$ be an immersed curve, let $0 \leq t_1 < t_2 \leq 1$, and let $\ell : [t_1, t_2] \to \mathbb{R}^d$ be a line segment with endpoints $\alpha[t_1]$ and $\alpha[t_2]$. Finally, let $\alpha' : [0,1] \to \mathbb{R}^d$ be $\alpha[0,t_1) \cup \ell[t_1,t_2] \cup \alpha(t_2,1]$, that is α' coincides with α in $[0,t_1) \cup (t_2,1]$, and coincides with the line segment ℓ on $[t_1,t_2]$. We say that α' is obtained from α via a **shortcutting operation**. The following lemma is key in our arguments.

▶ **Lemma 8** (Lemma 3 of Buchin et al. [9]). *Let $\alpha : [0,1] \to \mathbb{R}^d$ and $\alpha' : [0,1] \to \mathbb{R}^d$ be two curves, and let s be a line segment. If α' is obtained from α via a sequence of shortcutting operations then $\delta_F(\alpha', s) \leq \delta_F(\alpha, s)$.*

The following lemma follows from Lemma 4.1 and Lemma 4.2 of Erickson and Nayyeri [14]. Also, see Schaefer and Štefankovič [24] for similar shortcutting arguments.

▶ **Lemma 9** (Erickson and Nayyeri [14]). *Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ be a set of non-crossing curves on a triangulated surface (of genus zero) $\mathcal{Q} = (\mathcal{Q}_V, \mathcal{Q}_E, \mathcal{Q}_T)$. There exists a set of non-crossing curves $\Gamma' = \{\gamma'_1, \gamma'_2, \ldots, \gamma'_k\}$ with the following properties.*

1. *For each i, γ'_i is obtained from γ_i via a sequence of shortcutting operations along the edges in $\mathcal{Q}_E$.*
2. *For each $\gamma' \in \Gamma'$ and $t \in \mathcal{Q}_T$, each connected component of $\gamma' \cap t$ is*
   - **(a)** *a path with endpoints on different sides of t, or*
   - **(b)** *a path with one point being a vertex of t and the other on its opposite side, or*
   - **(c)** *a side of t, in this case γ' coincide with the side of t.*
3. *For each $e \in \mathcal{Q}_E$, if e is crossed by m different curves of Γ' then it is crossed at most 2^m times.*

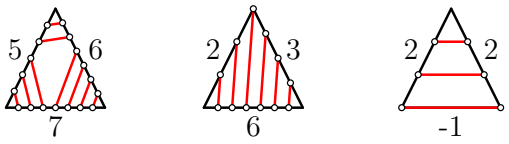


Figure 3 Examples for normal coordinates in triangles.

We remark that if Γ is composed of edges of a cellularly embedded graph on $\mathcal{Q}$, Γ' is the set of edges of a combinatorially equivalent cellular embedding on $\mathcal{Q}$. This is because the shortcutting operation does not change the boundary edges or the cyclic order of edges around vertices. Now, we are ready to show the existence of our desired scaffold map.

▶ **Lemma 10.** *Any approximate vertex map $f_0 : \widetilde{\mathcal{R}}_V \to \widetilde{\mathcal{S}}_V$ can be extended to a scaffold map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ of Fréchet length $\delta + 3r$ such that any edge $s \in \widetilde{\mathcal{S}}_E$ is crossed by $f_1(\widetilde{\mathcal{R}}_E)$ at most $2^{|B_{\delta+3r}(s) \cap \widetilde{\mathcal{R}}_E|}$ times. ($B_{\delta+3r}(s) \cap \widetilde{\mathcal{R}}_E$ denotes the subset of edges of $\widetilde{\mathcal{R}}_E$ that intersect $B_{\delta+3r}(s)$.)*

Proof. By Lemma 4, f_0 can be extended to a homeomorphism h of Fréchet length at most $\delta+3r$. Let $\Gamma = \{\gamma_e = h(e) | e \in \widetilde{\mathcal{R}}_E\}$, and note that $\delta_F(e, \gamma_e) \leq \delta+3r$. Because $\delta_F(h) \leq \delta+3r$, for any $s \in \widetilde{\mathcal{S}}_E$, if γ_e intersects s then e must intersect $B_{\delta+3r}(s)$. Thus, s is crossed by at most $|B_{\delta+3r}(s) \cap \widetilde{\mathcal{R}}_E|$ different γ_e's. Now, let $\Lambda = \{\lambda_e | e \in \widetilde{\mathcal{R}}_E\}$ be the set of paths obtained from Γ via Lemma 9 that has properties (1), (2), and (3). In particular, any segment $s \in \widetilde{\mathcal{S}}$ is crossed at most $2^{|B_{\delta+3r}(s) \cap \widetilde{\mathcal{R}}_E|}$ times.

We further modify Λ to obtain a set of piecewise linear paths $\Omega = \{\omega_e | e \in \widetilde{\mathcal{R}}_E\}$. For each $\lambda_e \in \Lambda$ and each $t \in \widetilde{\mathcal{S}}_T$, let $\{\lambda_e^1, \ldots, \lambda_e^k\}$ be the connected components of $\lambda_e \cap t$. For each $1 \leq i \leq k$, we define ω_e^i to be the straight line segment between the endpoints of λ_e^i. Property (2) of Lemma 9 implies that each ω_e^i is (a) a line segment between different sides of t, (b) a line segment between a vertex of t and its opposite side, or (c) a side of t (in this case $\omega_e = \omega_e^i$). Since each ω_e is obtained from λ_e via a sequence of shortcutting operations, which itself is obtained from γ_e via a sequence of shortcutting operations, we have $\delta_F(e, \omega_e) \leq \delta_F(e, \lambda_e) \leq \delta_F(e, \gamma_e) \leq \delta + 3r$ (Lemma 8). ◀

3.2.3 Enumeration via normal coordinates

Let $g_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ be a scaffold map, and let $t \in \widetilde{\mathcal{S}}_T$. The intersection of $g_1(\widetilde{\mathcal{R}}_E)$ with t is a collection of **elementary segments**: straight line segments with endpoints on $\partial(t)$. The intersection pattern of $g_1(\widetilde{\mathcal{R}}_E) \cap t$ can be presented (up to continuous deformation) with three numbers, one per edge. For each edge $s \in \mathcal{S}_E$ we define its **(extended) normal coordinate**, denoted by $N(e)$, as follows: (1) $N(e) = -1$ if $e \in g_1(\widetilde{\mathcal{R}}_E)$, and (2) $N(e)$ is the number of elementary segments intersecting the interior of e, otherwise. See Figure 3 for examples of extended normal coordinates in triangles. (Our (extended) normal coordinates are straight forward extensions of normal coordinates defined for normal curves in surfaces, or normal surfaces in 3-manifolds. See references [23, 26, 15] for a detailed exposition of normal curves, the two dimensional variant of standard normal surfaces introduced by Haken [18].)

The set of normal coordinates of $g_1(\widetilde{\mathcal{R}}_E)$ is a vector of $|\widetilde{\mathcal{S}}_E|$ numbers, one per edge in $\widetilde{\mathcal{S}}_E$. Provided the normal coordinates, there is a unique way of locating the elementary segments inside each $t \in \widetilde{\mathcal{S}}_T$ (up to a continuous deformation) so that they do not cross. Hence, the normal coordinates specify, for every $e \in \widetilde{\mathcal{R}}_E$, the sequence of edges that $g_1(e)$ crosses (its homotopy class in $\widetilde{\mathcal{S}} \backslash \widetilde{\mathcal{S}}_E$). The following corollary is implied by Lemma 10.

▶ **Corollary 11.** *There is a collection $\mathcal{N}$ of sets of normal coordinates of size at most*

$$\prod_{s \in \widetilde{\mathcal{S}}_E} 2^{|B_{\delta+3r}(s) \cap \widetilde{\mathcal{R}}_E|},$$

such that for any approximate vertex map $f_0 : \widetilde{\mathcal{R}}_V \to \widetilde{\mathcal{S}}_V$, $\mathcal{N}$ contains the set of normal coordinates of a scaffold map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ with the following properties:
1. *f_1 is an extension of f_0.*
2. *$\delta_F(f_1) \leq \delta + 3r$.*

Next, we show that provided the normal coordinates of a scaffold map g_1, we can build another scaffold map f_1 with the same normal coordinates within the Fréchet distance $O(r)$ of g_1. In fact, we show the following stronger lemma.

▶ **Lemma 12.** *Let $g_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ be a scaffold map of Fréchet length δ', and let $e \in \widetilde{\mathcal{R}}_E$. Let $T \subseteq \widetilde{\mathcal{S}}_T$ be the set of all triangles that intersect $g_1(e)$. For any point $x \in e$ and any point $y \in t \in T$, we have $||x - y|| \leq \delta' + 2r$.*

Proof. Let $z \in g(e) \cap t$, and let $x' = g^{-1}(z)$. Because x and x' are both on e, we have $||x - x'|| \leq r$. Since y and z are in the same triangle t, we have $||z - y|| \leq r$. Therefore,

$$||x - y|| \leq ||x - x'|| + ||x' - z|| + ||z - y|| \leq r + \delta' + r \leq \delta' + 2r\,. \qquad \blacktriangleleft$$

The following corollary is immediate observing that the normal coordinates uniquely specify the sequence of triangles each curves cross.

▶ **Corollary 13.** *Let $g_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ and $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ be two scaffold maps with identical sets of normal coordinates. For each $e \in \widetilde{\mathcal{R}}$, we have $\delta_F(e, f_1(e)) \leq \delta_F(e, g_1(e)) + 2r$.*

▶ **Lemma 14.** *Let N be the set of normal coordinates of a scaffold map $g_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$. Provided N, there is an algorithm to compute a scaffold map $f_1 : \widetilde{\mathcal{R}}_E \to \widetilde{\mathcal{S}}$ such that $\delta_F(f_1) \leq \delta_F(g_1) + 2r$.*

Proof. By Corollary 13, it suffices to find any scaffold map with normal coordinates N. For each $s \in \widetilde{\mathcal{S}}_E$, we arbitrarily select $N(s)$ points on s. N uniquely determines, for each $t \in \widetilde{\mathcal{S}}_T$, which points should be connected with elementary segments. For each edge $e \in \widetilde{\mathcal{R}}_E$, we obtain a path γ_e composed of elementary segments. Lemma 12 implies that any homeomorphism between e and γ_e has Fréchet length at most $\delta_F(g_1) + 2r$. Our f_1 will be the union of all such homeomorphisms, one per each $e \in \widetilde{\mathcal{R}}_E$. $\blacktriangleleft$

3.3 Summing up

Now, we are ready to prove the main lemmas of this section. The following lemma describes an algorithm that for a sufficiently large δ returns a homeomorphism of Fréchet length close to δ, and for a sufficiently small δ it decides that no homeomorphism with Fréchet length δ exists.

▶ **Lemma 15.** *Let $\mathcal{R}$ and $\mathcal{S}$ be two triangulated surfaces composed of triangles of diameter at most r, and let $\delta > 0$. Let ρ be the maximum number of (immersed) vertices of $\mathcal{R}_V \cup \mathcal{S}_V$ in any ball of radius $\max(\delta, r)$. There exists a $2^{O(\rho \cdot (|\mathcal{R}_V| + |\mathcal{S}_V|))}$ time algorithm with the following properties.*
1. *If $\delta \geq \delta_F(\mathcal{R}, \mathcal{S})$, it computes a homeomorphism, $f : \mathcal{R} \to \mathcal{S}$, of Fréchet length at most $\delta + 5r$,*

2. *If $\delta < \delta_F(\mathcal{R},\mathcal{S}) - 5r$, it decides that $\delta < \delta_F(\mathcal{R},\mathcal{S})$.*
3. *If $\delta_F(\mathcal{R},\mathcal{S}) - 5r \le \delta < \delta_F(\mathcal{R},\mathcal{S})$, either it computes a homeomorphism, $f : \mathcal{R} \to \mathcal{S}$, of Fréchet length at most $\delta + 5r$, or it decides that $\delta \le \delta_F(\mathcal{R},\mathcal{S})$.*

Proof. First, consider the case $\delta \ge \delta_F(\mathcal{R},\mathcal{S})$. By Lemma 5, there is a set of size $O(\rho^{|\mathcal{R}_V|+|\mathcal{S}_V|})$ that contains an approximate vertex map. Let $f_0 : \widetilde{\mathcal{R}} \to \widetilde{\mathcal{S}}$ be an approximate vertex map. Let $\widetilde{\rho}$ be the maximum number of vertices of $\widetilde{\mathcal{R}}_V \cup \widetilde{\mathcal{S}}_V$ in any ball of radius $\max(\delta, r)$. Let B be any ball of radius $\max(\delta, r)$, and let $\widetilde{B}$ be the concentric ball of radius $\max(\delta, r) + (\delta + r)$. For any vertex $u \in \mathcal{R}_V \cap B$, we have $f_0(u) \in \widetilde{B}$, and for any vertex $v \in \mathcal{S}_V \cap B$, we have $f_0^{-1}(v) \in \widetilde{B}$. Since, $\widetilde{B}$ can be covered by a constant number of balls of radii $\max(\delta, r)$, it contains $O(\rho)$ vertices of $\mathcal{R}_V \cup \mathcal{S}_V$. That is, B contains $O(\rho)$ vertices of $\widetilde{\mathcal{R}}_V \cup \widetilde{\mathcal{S}}_V$. We conclude that $\widetilde{\rho} = O(\rho)$.

Let $e \in \mathcal{R}_E$, and consider $B_{\delta+3r}(e) \cap \widetilde{\mathcal{S}}_E$, the set of edges in $\widetilde{\mathcal{S}}_E$ that intersect $B_{\delta+3r}(e)$. Any such an edge must have both of its endpoints in $B_{\delta+4r}(e)$, for the length of each edge is at most r. Now, consider the graph H induced by the vertices in $B_{\delta+4r}(e)$, which includes all edges in $B_{\delta+3r}(e) \cap \widetilde{\mathcal{S}}_E$. H is a collection of planar graphs, thus, its number of vertices and edges are within a constant factor. That is, $|B_{\delta+3r}(e) \cap \widetilde{\mathcal{S}}_E| = O(|B_{\delta+4r}(e) \cap \widetilde{\mathcal{S}}_V|)$. On the other hand, $B_{\delta+4r}(e)$ can be covered by a constant number of balls of radius $\max(\delta, r)$, therefore, $|B_{\delta+4r}(e) \cap \widetilde{\mathcal{S}}_V| = O(\widetilde{\rho}) = O(\rho)$. Overall, $|B_{\delta+3r}(e) \cap \widetilde{\mathcal{S}}_E| = O(\rho)$.

Therefore, Corollary 11 implies that a collection $\mathcal{N}$ of normal coordinate sets exists such that (1) $|\mathcal{N}| = 2^{O(\rho \cdot |\widetilde{\mathcal{S}}_E|)} = 2^{O(\rho \cdot (|\mathcal{R}_V|+|\mathcal{S}_V|))}$, and (2) $\mathcal{N}$ includes the normal coordinates of a scaffold map of Fréchet length $\delta + 3r$ that is an extension of f_0. Provided these normal coordinates, the algorithm of Lemma 14 computes a scaffold map f_1 of Fréchet length $\delta + 5r$. The total running time is dominated by the size of $\mathcal{N}$.

For any value of δ, our algorithm either computes a correct homeomorphism via computing a scaffold map or it fails to compute such a homeomorphism. In the former case, our algorithm returns the homeomorphism only if its Fréchet length is at most $\delta + 5r$. Therefore, if $\delta < \delta_F(\mathcal{R},\mathcal{S}) - 5r$ our algorithms recognizes that $\delta < \delta_F(\mathcal{R},\mathcal{S})$. For intermediate δ values, our algorithm either returns a homeomorphism of Fréchet length $\delta + 5r$, or it recognizes that a homeomorphism of Fréchet length δ does not exist. ◀

Next, we use binary search with Lemma 15 to estimate the value of the Fréchet distance.

▶ **Lemma 16.** *Let $\mathcal{R}$ and $\mathcal{S}$ be two triangulated surfaces composed of triangles of diameter at most r. Let ρ be the maximum number of vertices of $\mathcal{R}_V \cup \mathcal{S}_V$ in any ball of radius $\max(\delta_F(\mathcal{R},\mathcal{S}), r)$. There exists a $2^{O(\rho \cdot (|\mathcal{R}_V|+|\mathcal{S}_V|))} \log(\delta_F(\mathcal{R},\mathcal{S})/r)$ time algorithm to compute an $(\mathcal{R},\mathcal{S})$-homeomorphism of Fréchet length at most $\delta_F(\mathcal{R},\mathcal{S}) + 6r$*

Proof. We use Lemma 15 as a black box in an exponential and a binary search for $\delta_F(\mathcal{R},\mathcal{S})$. Initially, we set $\delta = r$, we perform an exponential search to find the smallest $2^k \cdot r$, for which Lemma 15 returns a homeomorphism. Hence, we know that $2^{k-1} \cdot r \le \delta_F(\mathcal{R},\mathcal{S}) < 2^k \cdot r$. Then, we perform a binary search to reduce the size of this gap from $O(2^k \cdot r)$ to r, to guarantee that our homeomorphism has Fréchet length at most $\delta_F(\mathcal{R},\mathcal{S}) + 6r$. Both the exponential and the binary search can be done in $O(k)$ time. Since $2^k \cdot r > \delta_F(\mathcal{R},\mathcal{S})$, we have $k = O\left(\log(\frac{\delta_F(\mathcal{R},\mathcal{S})}{r})\right)$. Thus, the total running time is

$$2^{O(\rho \cdot (|\mathcal{R}_V|+|\mathcal{S}_V|))} \cdot \log(\frac{\delta_F(\mathcal{R},\mathcal{S})}{r}).$$ ◀

4 General surfaces

In this section, we describe an algorithm to compute the Fréchet distance between two arbitrary surfaces (of genus zero) $\mathcal{R} = (\mathcal{R}_V, \mathcal{R}_E, \mathcal{R}_T)$ and $\mathcal{S} = (\mathcal{S}_V, \mathcal{S}_E, \mathcal{R}_T)$. To simplify our running time analysis, we assume that $\mathcal{R}$ and $\mathcal{S}$ are composed of fat triangles, that is all their angles are larger than a constant $\theta > 0$. In general, our running time would depend on the minimum angle of the constituent triangles of the surfaces.

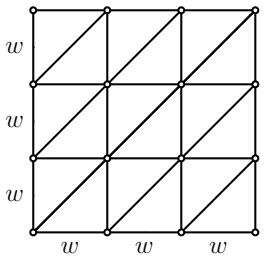
We define an **r-refinement** of a triangulation to be a refinement composed of triangles of diameter at most r. Before refining $\mathcal{R}$ and $\mathcal{S}$, we define triangulated grids, which we find helpful here and in the next section.

Triangulated grids. For any $w \in \mathbb{R}$, let $G_w = (V_w, E_w)$ be a triangulated grid of width w. That is

$$V_w = \{(iw, jw) \,|i, j \in \mathbb{Z}\},$$

and

$$E_w = \{((iw, jw), (i'w, j'w)) \,|i, j, i', j' \in \mathbb{Z} \wedge (i - i', j - j') \in \{(0, 1), (1, 0), (1, 1)\}\}.$$



▶ **Lemma 17.** *Let $\mathcal{Q} = (\mathcal{Q}_V, \mathcal{Q}_E, \mathcal{Q}_T)$ be a triangulated surface composed of fat triangles, and let $r \in \mathbb{R}^+$. There exists an $O(|\mathcal{Q}_V| + Area(\mathcal{Q})/r^2)$ time algorithm to compute an r-refinement of $\mathcal{Q}$ of size $O(|\mathcal{Q}_V| + Area(\mathcal{Q})/r^2)$.*

Proof. For each triangle $t \in \mathcal{Q}$ with diameter larger than r, we show how to refine it into a new triangulation composed of $O(\text{Area}(t)/r^2)$ triangles with diameter r. Note that when we put these triangulations together their vertices do not necessarily match on the border of different triangles. As a result, we may see flat polygons with more than three sides, but, we can triangulate them without introducing new vertices.

Let $t \in \mathcal{Q}_T$ with side lengths ℓ, ℓ' and ℓ'', with $\max(\ell, \ell', \ell'') > r$. Place t on G_r, the triangulated grid of width r. Let $\bar{t}$ be the triangulation of t induced by G_r. The number of triangles in $\bar{t}$ is $|\bar{t}| = O((\ell + \ell' + \ell'')/r + \text{Area}(t)/r^2)$. Since all angles of t are larger than a constant, ℓ, ℓ', and ℓ'' are within a constant factor of each other, and $\text{Area}(t) = \Theta(\ell^2)$. Therefore, $|\bar{t}| = O(\ell/r + \ell^2/r^2) = O(\ell^2/r^2) = O(\text{Area}(t)/r^2)$. ◀

Our algorithm for general surfaces is implied by Lemma 16 and Lemma 17.

Proof of Theorem 1. Let $\delta = \delta_F(\mathcal{R}, \mathcal{S})$, and let $r = (\varepsilon\delta)/6$. Let $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$ be r-refinements of $\mathcal{R}$ and $\mathcal{S}$, respectively, obtained by applying the algorithm of Lemma 17, thus, $|\overline{\mathcal{R}}_V| = O(|\mathcal{R}_V| + \text{Area}(\mathcal{R})/r^2)$, and $|\overline{\mathcal{S}}_V| = O(|\mathcal{S}_V| + \text{Area}(\mathcal{S})/r^2)$. Also, $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$ can be computed in linear time with respect to their sizes. Trivially, the number of vertices of $\overline{\mathcal{R}}_V \cup \overline{\mathcal{S}}_V$ in each ball of radius $\max(\delta, r)$ is at most $\overline{n} = |\overline{\mathcal{R}}_V| + |\overline{\mathcal{S}}_V|$. Thus, by Lemma 16 for $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$, there

is an $2^{O(\overline{n}^2)}\log(1/\varepsilon)$ time algorithm to compute an $(\mathcal{R},\mathcal{S})$-homoemorphism of Fréchet length at most $\delta + 6r = (1+\varepsilon)\delta$. We have,

$$2^{O(\overline{n}^2)}\log(\frac{1}{\varepsilon}) = 2^{O(|\overline{\mathcal{R}}_V|+|\overline{\mathcal{S}}_V|)^2}\log(\frac{1}{\varepsilon}) = 2^{O\left(\left(|\mathcal{R}_V|+|\mathcal{S}_V|+\frac{\text{Area}(\mathcal{R})+\text{Area}(\mathcal{S})}{(\varepsilon\delta)^2}\right)^2\right)}.$$ ◀

5 Terrains

In this section, we describe an algorithm to compute the Fréchet distance between two polyhedral terrains $\mathcal{R}$ and $\mathcal{S}$ over $[0,1]^2$ (i.e. the images of the immersions $\varphi_{\mathcal{R}} : \mathcal{R} \to \mathbb{R}^3$ and $\varphi_{\mathcal{S}} : \mathcal{S} \to \mathbb{R}^3$ are polyhedral terrains over $[0,1]^2$). Let $\delta = \delta_F(\mathcal{R},\mathcal{S})$, and let D be the maximum slope of $\mathcal{R}$ and $\mathcal{S}$ for any point in their domain, $[0,1]^2$.

5.1 Sampling

Let $\mathcal{Q} : [0,1]^2 \to \mathbb{R}$ be a polyhedral terrain, let $1/r \in \mathbb{Z}$, and let $G_r = (V_r, E_r)$ be a grid of width r. Here we use $\mathcal{Q}$ both to refer to the triangulated surface and to the function over $[0,1]^2$. The **r-coarse approximation** of $\mathcal{Q}$, is a polyhedral terrain $\overline{\mathcal{Q}}$, whose vertex set is,

$$\overline{\mathcal{Q}}_V = \{(x,y,\mathcal{Q}(x,y)) \,|\, (x,y) \in V_r\},$$

and whose edge set is

$$\overline{\mathcal{Q}}_E = \{((x,y,\mathcal{Q}(x,y)),(x',y',\mathcal{Q}(x',y'))) \,|\, (x,y),(x',y') \in E_r\}.$$

Again, we view $\overline{\mathcal{Q}}$ as a triangulated surface as well as a function $\overline{\mathcal{Q}} : [0,1]^2 \to \mathbb{R}$, thus, we use $\overline{\mathcal{Q}}(x,y)$ for a point $(x,y) \in [0,1]^2$.

▶ **Lemma 18.** *Let $\mathcal{Q} : [0,1]^2 \to \mathbb{R}$ be a polyhedral terrain with maximum slope D, and let $\overline{\mathcal{Q}}$ be its r-coarse approximation, where $1/r \in \mathbb{Z}$. We have $\delta_F(\mathcal{Q},\overline{\mathcal{Q}}) \le 2\sqrt{2}\cdot rD$.*

Proof. Let $f : \mathcal{Q} \to \overline{\mathcal{Q}}$ be the projection map along the z-axis. That is, for any $(x,y,z) \in \mathcal{Q}$, $f(x,y,z) = (x,y,\overline{\mathcal{Q}}(x,y))$. Let t be the triangle in G_r that contains (x,y), and let (x',y') be any vertex of t. We have $||(x,y)-(x',y')|| \le \sqrt{2}\cdot r$, which implies,

$$||(x,y,\mathcal{Q}(x,y)) - (x',y',\mathcal{Q}(x',y'))|| \le \sqrt{2}\cdot rD,$$

and

$$||(x,y,\overline{\mathcal{Q}}(x,y)) - (x',y',\overline{\mathcal{Q}}(x',y'))|| \le \sqrt{2}\cdot rD,$$

for the maximum slope of $\overline{\mathcal{Q}}$ is bounded by D too.

On the other hand, since (x',y') is a grid point $\overline{\mathcal{Q}}(x',y') = \mathcal{Q}(x',y')$. Thus, by the triangle inequality,

$$||(x,y,\mathcal{Q}) - (x,y,\overline{\mathcal{Q}})|| \le 2\sqrt{2}\cdot rD.$$ ◀

Proof of Theorem 2. Let $r' = \min(\varepsilon\delta/12, \varepsilon\delta/(8\sqrt{2}D))$, let $1/r$ be the smallest integer larger than $1/r'$, and let $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$ be r-refinements of $\mathcal{R}$ and $\mathcal{S}$, respectively. Consider any point $p = (x,y,z) \in \mathbb{R}^3$. The number of vertices of $\overline{\mathcal{R}}_V \cup \overline{\mathcal{S}}_V$ in $B_{\max(\delta,r)}(p)$ is at most the number of grid points, vertices of V_r, in a disk of radius $\max(\delta,r)$ with center (x,y), which is $O(\delta^2/r^2)$. Thus, Lemma 16 implies that an $(\overline{\mathcal{R}},\overline{\mathcal{S}})$-homeomorphism of Fréchet length $\delta+6r$ can be computed in $2^{O(\delta^2/r^4)} = 2^{O(D^4/(\varepsilon^4\delta^2))}$ time. Composing this homeomorphism

with the $(\mathcal{R}, \overline{\mathcal{R}})$-homeomorphism and the $(\overline{\mathcal{S}}, \mathcal{S})$-homeomorphism of Lemma 18, we obtain a homeomorphism of Fréchet length

$$\delta_F(S,T) + 6r + 4\sqrt{2} \cdot rD \leq \delta + \varepsilon\delta/2 + \varepsilon\delta/2 = (1+\varepsilon)\delta.$$

We need to sample $O(D/(\varepsilon\delta)^2)$ points from $\mathcal{R}$ and $\mathcal{S}$ to compute $\overline{\mathcal{R}}$ and $\overline{\mathcal{S}}$, which takes $O(D/(\varepsilon\delta)^2 n)$ time. Therefore, overall, we obtain the desired asymptotic time bound. ◀

Acknowledgement. The authors would like to thank Glencora Borradaile for helpful discussions.

References

1 Helmut Alt and Maike Buchin. Semi-computability of the Fréchet distance between surfaces. In *Proceedings of the 21st European Workshop on Computational Geometery*, pages 45–48, 2005.

2 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geometry Appl.*, 5:75–91, 1995. doi:10.1142/S0218195995000064.

3 Boris Aronov, Sariel Har-Peled, Christian Knauer, Yusu Wang, and Carola Wenk. Fréchet distance for curves, revisited. In *Proceedings of the 14th Conference on Annual European Symposium*, pages 52–63, 2006. doi:10.1007/11841036_8.

4 Rinat Ben Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete fréchet distance with shortcuts via approximate distance counting and selection. In *Proceedings of the 30th Annual Symposium on Computational Geometry*, pages 377–386, 2014. doi:10.1145/2582112.2582155.

5 Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 31st Conference on Very Large Data Bases*, pages 853–864, 2005.

6 Kevin Buchin, Maike Buchin, and Joachim Gudmundsson. Detecting single file movement. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 1–10, 2008.

7 Kevin Buchin, Maike Buchin, Joachim Gudmundsson, Maarten Löffler, and Jun Luo. Detecting commuting patterns by clustering subtrajectories. In *Proceedings of the 19th International Symposium on Algorithms and Computation*, pages 644–655, 2008.

8 Kevin Buchin, Maike Buchin, and André Schulz. Fréchet distance of surfaces: Some simple hard cases. In *Proceedings of the 18th Conference on Annual European Symposium*, pages 63–74, 2010.

9 Kevin Buchin, Maike Buchin, and Carola Wenk. Computing the Fréchet distance between simple polygons. *Comp. Geom. Theo. Appl.*, 41(1-2):2–20, October 2008. doi:10.1016/j.comgeo.2007.08.003.

10 Atlas F. Cook IV, Anne Driemel, Jessica Sherette, and Carola Wenk. Computing the frechet distance between folded polygons. *Computational Geometry*, 50:1–16, 2015. doi:10.1016/j.comgeo.2015.08.002.

11 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: Computing the fréchet distance with shortcuts. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–337, 2012.

12 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, 2012. doi:10.1007/s00454-012-9402-z.

13 Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Tech. Report CD-TR 94/64, Christian Doppler Lab. Expert Sys., TU Vienna, Austria, 1994.

14 Jeff Erickson and Amir Nayyeri. Shortest non-crossing walks in the plane. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 297–308, 2011.

15 Jeff Erickson and Amir Nayyeri. Tracing compressed curves in triangulated surfaces. In *Proceedings of the 28th Annual Symposium on Computational Geometry*, pages 131–140, 2012.

16 Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer Berlin Heidelberg, 2005. doi:10.1007/3-540-26808-1_9.

17 Michael Godau. *On the Complexity of Measuring the Similarity Between Geometric Objects in Higher Dimensions*. PhD thesis, Freie Universität Berlin, 1998.

18 Wolfgang Haken. Theorie der Normalflächen: Ein Isotopiekriterium für den Kreisknoten. *Acta Mathematica*, 105:245–375, 1961.

19 Sariel Har-Peled and Benjamin Raichel. The fréchet distance revisited and extended. *ACM Transactions on Algorithms*, 10(1):3, 2014. doi:10.1145/2532646.

20 Man-Soon Kim, Sang-Wook Kim, and Miyoung Shin. Optimization of subsequence matching under time warping in time-series databases. In *Proceedings of ACM Symposium on Applied Computing*, pages 581–586, 2005.

21 Ariane Mascret, Thomas Devogele, Iwan Le Berre, and Alain Hénaff. Coastline matching process based on the discrete Fréchet distance. In *Proceedings of the12th International symposium on Spatial Data Handling*, pages 383–400, 2006.

22 Amir Nayyeri and Anastasios Sidiropoulos. Computing the fréchet distance between polygons with holes. In *The proceedings of The 42nd International Colloquium on Automata, Languages, and Programming*, pages 997–1009, 2015. doi:10.1007/978-3-662-47672-7_81.

23 Marcus Schaefer, Eric Sedgwick, and Daniel Stefankovic. Algorithms for normal curves and surfaces. In *Proceedings if the 8th Annual International Computing and Combinatorics Conference*, pages 370–380, 2002.

24 Marcus Schaefer and Daniel Štefankovič. Decidability of string graphs. *J. Comput. Syst. Sci.*, 68(2):319–334, 2004.

25 Joan Serrà, Emillia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech & Language Processing*, 16(6):1138–1151, 2008.

26 Daniel Stefankovic. *Algorithms for Simple Curves on Surfaces, String Graphs, and Crossing Numbers*. PhD thesis, University of Chicago, Chicago, IL, USA, 2005. AAI3168396.

27 Oliver van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

28 Carola Wenk, Randall Salas, and Dieter Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management.*, pages 879–888, 2006.

The Farthest-Point Geodesic Voronoi Diagram of Points on the Boundary of a Simple Polygon*

Eunjin Oh[1], Luis Barba[2], and Hee-Kap Ahn[3]

1 **Department of Computer Science and Engineering, POSTECH, 77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, Korea**
jin9082@postech.ac.kr

2 **Département d'Informatique, Université Libre de Bruxelles, Brussels, Belgium; and School of Computer Science, Carleton University, Ottawa, Canada**
lbarbafl@ulb.ac.be

3 **Department of Computer Science and Engineering, POSTECH, 77 Cheongam-Ro, Nam-Gu, Pohang, Gyeongbuk, Korea**
heekap@postech.ac.kr

Abstract

Given a set of sites (points) in a simple polygon, the farthest-point geodesic Voronoi diagram partitions the polygon 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 metric. We present an $O((n+m)\log\log n)$-time algorithm to compute the farthest-point geodesic Voronoi diagram for m sites lying on the boundary of a simple n-gon.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases Geodesic distance, simple polygons, farthest-point Voronoi diagram

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.56

1 Introduction

Let P be a simple polygon with n vertices. Given two points x and y in P, the *geodesic path* $\pi(x,y)$ is the shortest path contained in P connecting x with y. Note that if the straight-line segment connecting x with y is contained in P, then $\pi(x,y)$ is a straight-line segment. Otherwise, $\pi(x,y)$ is a polygonal chain whose vertices (other than its endpoints) are reflex vertices of P. We refer the reader to [10] for more information on geodesic paths.

The *geodesic distance* between x and y, denoted by $d(x,y)$, is the sum of the Euclidean lengths of each segment in $\pi(x,y)$. Throughout this paper, when referring to the distance between two points in P, we mean the geodesic distance between them. To ease the description, we assume that each vertex of P has a unique farthest neighbor. This *general position* condition was also assumed by Aronov et al. [3] and Ahn et al. [2] and can be obtained by applying a slight perturbation to the positions of the vertices [7].

Let S be a set of m *sites* (points) contained in P. Given a point $x \in P$, a (geodesic) *S-farthest neighbor* of x, is a site $\mathrm{N}(P,S,x)$ (or simply $\mathrm{N}(x)$) of S that maximizes the geodesic distance to x. Let $F_S : P \to \mathbb{R}$ be the function that maps each point $x \in P$ to the distance to a S-farthest neighbor of x (i.e., $F_S(x) = d(x, \mathrm{N}(x))$). A point x in P that minimizes $F_S(x)$ is called the *geodesic center* of S (in P).

* This was supported by the NRF grant 2011-0030044 (SRC-GAIA) funded by the government of Korea.

© Eunjin Oh, Luis Barba, and Hee-Kap Ahn;
licensed under Creative Commons License CC-BY

We can decompose P into *Voronoi cells* such that for each site $s \in S$, $\mathsf{Cell}(s)$ is the set of points $x \in P$ such that $d(x, s)$ is strictly larger than $d(x, s')$ for any other site s' of S (some cells might be empty). The set $\text{int}(P) \setminus \cup_{s \in S} \mathsf{Cell}(s)$ defines the (farthest) *Voronoi tree* of S with root at the geodesic center of S and leaves on the boundary of P. Each edge of this diagram consists of a sequence of straight-lines and hyperbolic arcs [3].

The Voronoi tree together with the set of Voronoi cells defines the *farthest-point geodesic Voronoi diagram* of S (in P), denoted by $\mathsf{FVD}[S]$ (or simply FVD if S is clear from context). Thus, we indistinctively refer to FVD as a tree or as a set of Voronoi cells.

There are many similarities between the Euclidean farthest-point Voronoi diagram and the farthest-point geodesic Voronoi diagram (see [3] for further references). In the Euclidean case, a site has a nonempty Voronoi cell if and only if it is extreme, i.e., it lies on the boundary of the convex hull of the set of sites. Moreover, the clockwise sequence of Voronoi cells (at infinity) is the same as the clockwise sequence of sites along the boundary of the convex hull. With these properties, the Euclidean farthest-point Voronoi diagram can be computed in linear time if the convex hull of the sites is known [1].

In the geodesic case, a site with nonempty Voronoi cell lies on the boundary of the geodesic convex hull of the sites. The clockwise order of the Voronoi cells along the boundary of P is a subsequence of the clockwise order of sites along the boundary of the geodesic convex hull. However, the cell of an extreme site may be empty, roughly because the polygon is not large enough for the cell to appear. In addition, the complexity of the bisector between two sites can be linear to the complexity of the polygon.

Previous work. Since the early 1980s many classical geometric problems have been studied in the geodesic setting. The problem of computing the geodesic diameter of the vertices of a simple n-gon P (and its counterpart, the geodesic center) received a lot of attention from the computational geometry community. Chazelle [6] gave the first algorithm for computing the geodesic diameter. This algorithm runs in $O(n^2)$ time using linear space. Suri [13] reduced the complexity to $O(n \log n)$-time without increasing the space complexity. Finally, Hershberger and Suri [8] presented a fast matrix search technique, one application of which is a linear-time algorithm for computing the diameter of P. A key step in this process is the computation of the farthest neighbor of each vertex in P.

The first algorithm for computing the geodesic center 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 $\mathsf{FVD}[V]$, where V is the set of vertices of P. In 1989, Pollack et al. [12] improved the running time to $O(n \log n)$ time. In a recent paper, Ahn et al. [2] settled the complexity of this problem by presenting a $\Theta(n)$-time algorithm to compute the geodesic center of a simple n-gon.

Since the geodesic center and diameter can both be computed from $\mathsf{FVD}[V]$ in linear time, the problem of computing farthest-point geodesic Voronoi diagrams is a strict generalization. For a set S of m points in P, Aronov et al. [3] presented an algorithm to compute $\mathsf{FVD}[S]$ in $O((n+m)\log(n+m))$ time. While a trivial lower bound of $\Omega(n + m \log m)$ is known for this general problem, there has been no progress closing this gap. In other words, it is 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 [10, Chapter 27] in the Handbook of Computational Geometry.

Our result. In this paper, we present an $O((n+m)\log\log n)$-time algorithm to compute FVD of m points on the boundary of a simple n-gon. This is the first improvement on the computation of farthest-point geodesic Voronoi diagrams since 1993 [3]. Indeed, while we

consider sites lying on the boundary of the polygon only, our approach can also be extended to handle arbitrary sites in the polygon. Then the running time becomes $O(n \log\log n + m \log(n+m))$. The details can be found in the full version of this paper. Our result suggests that the computation time of Voronoi diagrams has only a close-to-linear dependence in the complexity of the polygon. We believe our results could be used as a stepping stone to solve the question posed by Mitchell [10, Chapter 27]. Due to lack of space, some of the proofs are omitted. All missing proofs can be found in the full version of this paper.

1.1 Outline

The algorithm consists of three phases. First, we compute the farthest-point geodesic Voronoi diagram restricted to the boundary of the polygon. Then we recursively decompose the interior of the polygon into smaller (non-Voronoi) cells until the complexity of each of them becomes constant. Finally, we explicitly compute the farthest-point geodesic Voronoi diagram in each of the cells and merge them to complete the description of the Voronoi diagram.

In order to compute the Voronoi diagram of S, we start by computing the restriction of $\mathsf{FVD}[S]$ to the boundary of P in linear time. The main tool used to speed up the algorithm is the matrix search technique introduced by Hershberger and Suri [8] which provides a "partial" description of $\mathsf{FVD}[S] \cap \partial P$ (i.e., the restriction of $\mathsf{FVD}[S]$ to the vertices of P.) To extend it to the entire boundary of P, we borrow some tools used by Ahn et al. [2]. This reduces the problem to the computation of upper envelopes of distance functions which can be completed in linear time.

Once $\mathsf{FVD}[S]$ restricted to ∂P is computed, we recursively split the polygon into cells by a closed polygonal path in time linear to the complexity of the cell. By recursively repeating this procedure on each resulting cell, we guarantee that after $O(\log\log n)$ rounds the boundary of each cell consists of a constant number of geodesic paths. In particular, we guarantee that each cell is a pseudo-triangle, a quadrilateral, or a simple polygon enclosed by a convex chain and a concave chain which we call a *lune-cell*.

While decomposing the polygon, we also compute the farthest-point geodesic Voronoi diagram of S restricted to the boundary of each cell. Each round can be completed in linear time which leads to an overall running time of $O((n+m)\log\log n)$.

Finally, we compute the farthest-point geodesic Voronoi diagram restricted to each cell in time linear to the complexity of the cell using the algorithm in [5].

2 Decomposing the boundary

Given a set A of points, let ∂A and $\mathrm{int}(A)$ denote the boundary and the interior of A, respectively. Let P be a simple n-gon and S be a set of m sites (points) contained in ∂P. Throughout most of this paper, we will make the assumption that S is the set of all vertices of P. This assumption is general enough as we show how to extend the result to the case when S is an arbitrary set of sites contained on the boundary of P in Section 6.

The following result was used by Ahn et al. [2] and is based on the matrix search technique developed by Hershberger and Suri [8].

▶ **Lemma 1** (Result from [8]). *We can compute the S-farthest neighbor of each vertex of P in $O(n)$ time.*

Using Lemma 1, we mark the vertices of P that are S-farthest neighbors of at least one vertex of P. Let M denote the set of marked vertices of P (clearly this set can be computed

in $O(n)$ time after applying Lemma 1). In other words, M contains all vertices of P whose Voronoi region contains at least one vertex of P.

For a marked vertex w of P, the vertices of P whose farthest neighbor is w appear contiguously along ∂P [3]. That is, given an edge uv such that $\textsc{n}(u) = \textsc{n}(v)$, we know that $\textsc{n}(x) = \textsc{n}(u) = \textsc{n}(v)$ for each point $x \in uv$. Therefore, after computing all these farthest neighbors, we effectively split ∂P into subchains, each associated with a different vertex of M (see [2] further for the first use of this technique).

Given two points x and y on ∂P, let $C[x, y]$ denote the portion of ∂P from x to y in clockwise order. We say that three (nonempty) disjoint sets A_1, A_2 and A_3 contained in ∂P are in *clockwise order* if $A_2 \subset C[a, c]$ for any $a \in A_1$ and any $c \in A_3$.

▶ **Lemma 2** ([3, Corollary 2.7.4]). *The order of sites with nonempty Voronoi cells along ∂P is the same as the order of Voronoi cells along ∂P.*

We call an edge ab of P a *transition edge* if $\textsc{n}(a) \neq \textsc{n}(b)$. Let ab be a transition edge of P such that b is the clockwise neighbor of a along ∂P. Recall that we have computed $\textsc{n}(a)$ and $\textsc{n}(b)$ and note that $a, b, \textsc{n}(a), \textsc{n}(b)$ are in clockwise order by Lemma 2. Let v be a vertex of P such that $\textsc{n}(a), v, \textsc{n}(b)$ are in clockwise order. If there is a point x on ∂P whose farthest neighbor is v, then x must lie on ab. In other words, the Voronoi cell $\mathsf{Cell}(v)$ restricted to ∂P is contained in ab and hence, there is no vertex u of P such that $\textsc{n}(u) = v$.

Since we know which vertex is the farthest neighbor of each non-transition edge of P, to complete the description of FVD restricted to ∂P it suffices to compute FVD restricted to transition edges. To this end, we need some tools introduced in the following sections.

2.1 The apexed triangles

An *apexed triangle* $\triangle = (a, b, c)$ with *apex* $\textsc{a}(\triangle) = a$ is a triangle contained in P with an associated distance function $g_\triangle(x)$ such that (1) $\textsc{a}(\triangle)$ is a vertex of P, (2) there is an edge of ∂P containing both b and c, and (3) there is a vertex $\textsc{d}(\triangle)$ of P, called the *definer* of $\triangle$, such that

$$g_\triangle(x) = \begin{cases} \|x - \textsc{a}(\triangle)\| + d(\textsc{a}(\triangle), \textsc{d}(\triangle)) = d(x, \textsc{d}(\triangle)) & \text{if } x \in \triangle \\ -\infty & \text{if } x \notin \triangle, \end{cases}$$

where $\|x - y\|$ denote the Euclidean distance between x and y.

Intuitively, $\triangle$ bounds a constant complexity region where the geodesic distance function from $\textsc{d}(\triangle)$ can be obtained by looking only at the distance from $\textsc{a}(\triangle)$. We call the side of an apexed triangle $\triangle$ opposite to the apex the *bottom side* of $\triangle$. Note that the bottom side of $\triangle$ is contained in an edge of P.

The concept of the apexed triangle was introduced by Ahn et al. [2]. After computing the farthest S-neighbor of each vertex, they show how to compute a linear number of apexed triangles in linear time with the following property: for each point $p \in P$, there exists an apexed triangle $\triangle$ such that $p \in \triangle$ and $\textsc{d}(\triangle) = \textsc{n}(p)$. By the definition of the apexed triangle, we have $d(p, \textsc{n}(p)) = g_\triangle(p)$. To summarize the results presented by Ahn et al. [2], we need some definitions. Given a chain C contained in ∂P with endpoints u and v, the *funnel* of a site s to C, denoted by $\gamma_s(C)$, is the weakly simple polygon contained in P bounded by C, $\pi(u, s)$ and $\pi(s, v)$.

▶ **Lemma 3** (Summary of [2]). *Given a simple n-gon P with vertex set S, we can compute a set of $O(n)$ apexed triangles in $O(n)$ time with the property that for any site $s \in S$, the union of all apexed triangles with definer s is a funnel γ_s such that $\mathsf{Cell}(s) \subset \gamma_s$.*

In other words, Lemma 3 states that for each site s of S, the set of apexed triangles with definer s forms a connected component. In particular, the union of their bottom sides is a connected chain along ∂P. Moreover, these apexed triangles are interior disjoint.

2.2 The refined farthest-point geodesic Voronoi diagram

We consider a refined version of FVD which we call the *refined farthest-point geodesic Voronoi diagram* defined as follows: for each site $s \in S$, the Voronoi cell $\mathsf{Cell}(s)$ of FVD is subdivided by the apexed triangles with definer s. That is, for each apexed triangle $\triangle$ with definer s, we define a *refined cell* $\mathsf{rCell}(\triangle) = \mathrm{int}(\triangle) \cap \mathsf{Cell}(s)$. Since any two apexed triangles $\triangle_1$ and $\triangle_2$ with the same definer are interior disjoint, we know that $\mathsf{rCell}(\triangle_1)$ and $\mathsf{rCell}(\triangle_2)$ are also interior disjoint. We denote the set $\mathrm{int}(P) \setminus \cup_{\triangle} \mathsf{rCell}(\triangle)$ by rFVD. Then, rFVD forms a tree consisting of arcs and vertices. Notice that each arc of rFVD is a connected subset of either the bisector of two sites or a side of an apexed triangle. Since the number of the apexed triangles is $O(n)$, the complexity of rFVD is still linear.

▶ **Lemma 4.** *For a point x in $\mathsf{rCell}(\triangle)$ for an apexed triangle $\triangle$, the line segment connecting x and y is contained in $\mathsf{rCell}(\triangle)$, where y is the point on the bottom side of $\triangle$ hit by the ray from $\mathrm{A}(\triangle)$ towards x. Moreover, $\mathsf{rCell}(\triangle)$ is connected.*

3 Computing the farthest-point geodesic Voronoi diagram restricted to the boundary of the polygon

We compute all apexed triangles satisfying the condition in Lemma 3 in $O(n)$ time [2]. Recall that the apexed triangles with the same definer are interior disjoint and have their bottom sides on ∂P whose union forms a connected chain. Moreover, their union is a funnel by Lemma 3. Thus, the apexed triangles with the same definer can be sorted along ∂P.

▶ **Lemma 5.** *Let s be a site in S and let $\tau_s \neq \emptyset$ be the set of all apexed triangles with definer s. We can sort the apexed triangles in τ_s along ∂P with respect to their bottom sides in $O(|\tau_s|)$ time.*

3.1 Computing rFVD restricted to a transition edge

Let uv be a transition edge of P such that u is the clockwise neighbor of v. Without loss of generality, we assume that uv is horizontal and u lies to the left of v. Recall that if there is a site s with $\mathsf{Cell}(s) \cap uv \neq \phi$, then s lies in $C[\mathrm{N}(v), \mathrm{N}(u)]$. Thus, to compute $\mathsf{rFVD} \cap uv$, it is sufficient to consider the apexed triangles with definers in $C[\mathrm{N}(v), \mathrm{N}(u)]$. Let A be the set of apexed triangles with definers in $C[\mathrm{N}(v), \mathrm{N}(u)]$.

In this section, we give a procedure to compute $\mathsf{rFVD} \cap uv$ in $O(|A|)$ time using the sorted lists of the apexed triangles with definers in $C[\mathrm{N}(v), \mathrm{N}(u)]$. Once it is done for all transition edges, we have the refined farthest-point geodesic Voronoi diagram restricted to ∂P. Let $s_1 = \mathrm{N}(u), s_2, \ldots, s_\ell = \mathrm{N}(v)$ be the sites lying on $C[\mathrm{N}(v), \mathrm{N}(u)]$ in counterclockwise order along ∂P.

3.1.1 An upper envelope and rFVD

Consider any t functions $f_1, \ldots, f_t$ with $f_j : D \to \mathbb{R} \cup \{-\infty\}$ for $1 \leq j \leq t$, where D is any point set. We define the *upper envelope* of $f_1, \ldots, f_t$ as the function $f : D \to \mathbb{R} \cup \{-\infty\}$ such that $f(x) = \max_{1 \leq j \leq t} f_j(x)$. Moreover, we say that a function f_j *appears* on the upper envelope if $f_j(x) = f(x) \in \mathbb{R}$ at some point x.

In this subsection, we restrict the domain of the distance functions $g_\triangle$ to uv. By definition, the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ on uv coincides with $\mathsf{rFVD} \cap uv$ in its projection on uv. We consider the sites one by one in order and compute the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ on uv as follows.

While the upper envelope of $g_\triangle$ for all apexed triangles $\triangle \in A$ is continuous on uv, the upper envelope of $g_{\triangle'}$ of all apexed triangles $\triangle'$ with definers from s_1 up to s_k on uv (we simply say the upper envelope for sites from s_1 to s_k) might be discontinuous at some point on uv for $1 \leq k < \ell$. Let w be the leftmost point where the upper envelope for sites from s_1 to s_k is discontinuous. Then we define $U(s_k)$ as the function such that $U(s_k)(x)$ is the value of the upper envelope for sites from s_1 to s_k at x for a point x lying to the left of w, and $U(s_k)(x) = -\infty$ for a point x lying to the right of w. By definition, $U(\textsc{n}(v))$ is the upper envelope of the distance functions of all apexed triangles in A. Note that $\mathsf{rCell}(\triangle) \cap uv = \phi$ for some apexed triangle $\triangle \in A$. Thus the distance function of an apexed triangle might not appear on $U(s_k)$ on uv. Let $\tau_U(s_k)$ be the list of the apexed triangles whose distance functions appear on $U(s_k)$ sorted along uv from u with respect to their bottom sides. Note that if $\textsc{d}(\triangle_i) \neq \textsc{d}(\triangle_{i+1})$, the bisector of $\textsc{d}(\triangle_i)$ and $\textsc{d}(\triangle_{i+1})$ crosses the intersection of the bottom sides of $\triangle_i$ and $\triangle_{i+1}$ for two consecutive apexed triangles $\triangle_i$ and $\triangle_{i+1}$ of $\tau_U(s_k)$.

3.1.2 A procedure for computing $U(s_\ell)$

Suppose that we have already computed $U(s_{k-1})$ and $\tau_U(s_{k-1})$ for some index $2 \leq k \leq \ell$. We show how to compute $U(s_k)$ and $\tau_U(s_k)$ from $U(s_{k-1})$ and $\tau_U(s_{k-1})$ in the following. We use two auxiliary lists U' and τ'_U which are initially set to $U(s_{k-1})$ and $\tau_U(s_{k-1})$. We update U' and τ'_U until they finally become $U(s_k)$ and $\tau_U(s_k)$, respectively.

Let τ_k be the list of the apexed triangles with definer s_k sorted along ∂P with respect to their bottom sides. For any apexed triangle $\triangle$, we denote the list of the apexed triangles in τ_k overlapping with $\triangle$ in their bottom sides by $\tau_O(\triangle)$. Also, we denote the lists of the apexed triangles in $\tau_k \setminus \tau_O(\triangle)$ lying left to $\triangle$ and lying right to $\triangle$ with respect to their bottom sides by $\tau_L(\triangle)$ and $\tau_R(\triangle)$, respectively.

Let $\triangle_a$ denote the last element (the rightmost apexed triangle) of τ'_U. With respect to $\triangle_a$, we partition τ_k into three disjoint sublists $\tau_L(\triangle_a)$, $\tau_O(\triangle_a)$ and $\tau_R(\triangle_a)$. We can compute these sublists in $O(|\tau_k|)$ time.

Case 1: Some apexed triangles in τ_k overlap with $\triangle_a$. If $\tau_O(\triangle_a) \neq \phi$, let $\triangle$ be the leftmost apexed triangle in $\tau_O(\triangle_a)$. We compare the distance functions $g_\triangle$ and $g_{\triangle_a}$ on $\triangle_a \cap \triangle \cap uv$. That is, we compare $d(x, s_k)$ and $d(x, \textsc{d}(\triangle_a))$ for $x \in \triangle_a \cap \triangle \cap uv$.

1. If there is a point on $\triangle_a \cap \triangle \cap uv$ that is equidistant from s_k and $\textsc{d}(\triangle_a)$, $g_\triangle$ appears on $U(s_k)$. Moreover, the distance functions of the apexed triangles in $\tau_R(\triangle)$ also appear on $U(s_k)$, and no apexed triangle in $\tau_L(\triangle)$ appears on $U(s_k)$ by Lemma 2. Thus we append $\triangle$ and the apexed triangles in $\tau_R(\triangle)$ at the end of τ'_U. We also update U' accordingly. Then, τ'_U and U' are $\tau_U(s_k)$ and $U(s_k)$, respectively.
2. If $d(x, \textsc{d}(\triangle_a)) > d(x, s_k)$ for all points $x \in \triangle_a \cap \triangle \cap uv$, then $\triangle$ and its distance function do not appear on $\tau_U(s_k)$ and $U(s_k)$, respectively, by Lemma 2. Thus we do nothing and scan the apexed triangles in $\tau_O(\triangle_a) \cup \tau_R(\triangle_a)$, except $\triangle$, from left to right until we find an apexed triangle $\triangle'$ such that there is a point on $\triangle_a \cap \triangle' \cap uv$ which is equidistant from $\textsc{d}(\triangle_a)$ and s_k. Then we apply the procedure in (1) with $\triangle'$ instead of $\triangle$. If there is no such apexed triangle, we have $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$.

3. Otherwise, we have $d(x, s_k) > d(x, \mathrm{D}(\triangle_a))$ for all points $x \in \triangle_a \cap \triangle \cap uv$. Then the distance function of $\triangle_a$ does not appear on $U(s_k)$. Thus, we remove $\triangle_a$ and its distance function from τ'_U and U', respectively. We consider the apexed triangles in $\tau_L(\triangle_a)$ from right to left. For an apexed triangle $\triangle' \in \tau_L(\triangle_a)$, we do the following. Since τ'_U is updated, we update $\triangle_a$ to the last element of τ'_U. Afterwards, we check whether $d(x, s_k) \geq d(x, \mathrm{D}(\triangle_a))$ for all points $x \in \triangle_a \cap \triangle' \cap uv$ if $\triangle'$ overlaps with $\triangle_a$. If so, we remove $\triangle_a$ from τ'_U and update $\triangle_a$. We do this until we find an apexed triangle $\triangle' \in \tau_L(\triangle_a)$ such that this test fails. Then, there is a point on $\triangle' \cap \triangle_a \cap uv$ which is equidistant from $\mathrm{D}(\triangle_a)$ and s_k. After we reach such an apexed triangle $\triangle'$, we apply the procedure in (1) with $\triangle'$ instead of $\triangle$.

Case 2: No apexed triangle in τ_k overlaps with $\triangle_a$. If $\tau_O(\triangle_a) = \phi$, we cannot compare the distance function of any apexed triangle in τ_k with the distance function of $\triangle_a$ directly, so we need a different method to handle this.

There are two possible subcases: either $\tau_L(\triangle_a) = \phi$ or $\tau_R(\triangle_a) = \phi$. Note that these are the only possible subcases since the union of the apexed triangles with the same definer is connected. For the former subcase, the upper envelope of sites from s_1 to s_k is discontinuous at the right endpoint of the bottom side of $\triangle_a$. Thus $g_\triangle$ does not appear on $U(s_k)$ for any apexed triangle $\triangle \in \tau_k$. Thus $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$.

For the latter subcase, at most one of s_k and $\mathrm{D}(\triangle_a)$ has its Voronoi cell in $\mathsf{FVD}[S_k]$, where $S_k = \{s_1, \ldots, s_k\}$, by Lemma 2. We can find a site (s_k or $\mathrm{D}(\triangle_a)$) which does not have its Voronoi cell in $\mathsf{FVD}[S_k]$ in $O(1)$ time. Due to lack of space, we omit the description of this procedure. It can be found in the full version of this paper.

If s_k does not have its Voronoi cell in $\mathsf{FVD}[S_k]$, then $U(s_k) = U'$ and $\tau_U(s_k) = \tau'_U$. If $\mathrm{D}(\triangle_a)$ does not have its Voronoi cell in $\mathsf{FVD}[S_k]$, we remove all apexed triangles with definer $\mathrm{D}(\triangle_a)$ from τ'_U and their distance functions from U'. Since such apexed triangles lie at the end of τ'_U consecutively, it takes the time proportional to the number of the apexed triangles. Afterwards, we do this until the last element of τ_k and the last element of τ'_U overlap in their bottom sides. When the two elements overlap, we apply the procedure of Case 1.

In total, the running time is bounded by $O(|A|)$.

▶ **Theorem 6.** *The farthest-point geodesic Voronoi diagram of the vertices of a simple n-gon P restricted to the boundary of P can be computed in $O(n)$ time.*

4 Decomposing the polygon into smaller cells

Until now, we have computed $\mathsf{rFVD} \cap \partial P$ of size $O(n)$. We add the points in $\mathsf{rFVD} \cap \partial P$ to the vertex set of P, and apply the algorithm to compute the apexed triangles with respect to the vertex set of P again [2]. Note that now there is no transition edge. Thus all apexed triangles are disjoint in their bottom sides. We have the set of the apexed triangles sorted along ∂P with respect to their bottom sides.

A subset A of P is *geodesically convex* if $\pi(x, y) \subseteq A$ for any $x, y \in A$. We define a *t-path-cell* for some $t \in \mathbb{N}$ as a simple polygon contained in P with all vertices on ∂P which is geodesically convex and has at most t convex vertices.

In the following, for a cell C, $|\partial C|$ denotes the number of edges of C. For a curve γ, $|\mathsf{rFVD} \cap \gamma|$ denotes the number of the refined cells intersecting γ.

Sketch of the algorithm. We subdivide P into t-path-cells recursively for some $t \in \mathbb{N}$ until each cell becomes a base cell. There are three types of base cells. The first type is

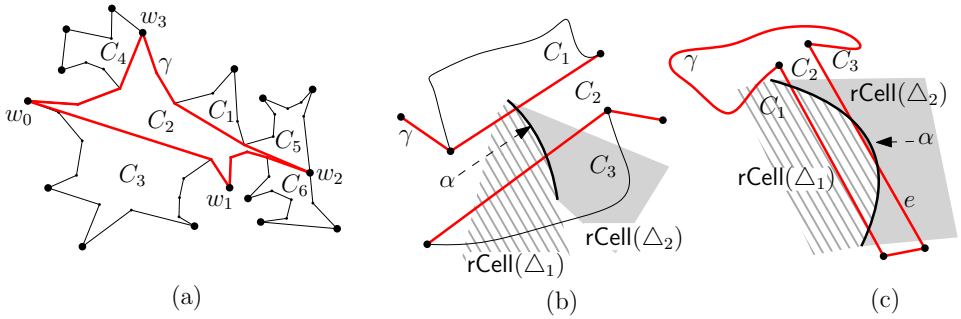


Figure 1 (a) The region bounded by the black curve is a 16-path-cell. All convex vertices are marked with black disks. The region is subdivided into six 5-path-cells by the red thick curve consisting of $\pi(w_0, w_1), \pi(w_1, w_2), \pi(w_2, w_3)$ and $\pi(w_3, w_0)$. (b) The arc α of rFVD intersects C_1, C_2, C_3 and crosses C_2. (c) The arc α of rFVD intersects C_1, C_2, C_3 and crosses C_2. Note that α does not cross C_3.

a quadrilateral crossed by exactly one arc of rFVD through two opposite sides, which we call an *arc-quadrilateral*. The second type is a 3-path-cell. Note that a 3-path-cell is a pseudo-triangle. The third type is a region of P whose boundary consists of one convex chain and one concave chain, which we call a *lune-cell*. Note that a convex polygon is a lune-cell whose concave chain is just a vertex of the polygon.

Let $\{t_k\}$ be the sequence such that $t_1 = n$ and $t_k = \lfloor\sqrt{t_{k-1}}\rfloor + 1$. Initially, P itself is a t_1-path-cell. Assume that the kth iteration is completed. We show how to subdivide each t_k-path-cell with $t_k > 3$ into t_{k+1}-path-cells and base cells in the $(k+1)$th iteration in Section 4.1. Note that a base cell is not subdivided further.

While subdividing the polygon into cells, we compute $\mathsf{rFVD} \cap \partial C$ for each cell C (of any kind) in time linear on $|\partial C|$ and $|\mathsf{rFVD} \cap \partial C|$. In Section 5, we show how to compute $\mathsf{rFVD} \cap T$ for each base cell T in $O(|\mathsf{rFVD} \cap \partial T|)$ time once $\mathsf{rFVD} \cap \partial T$ is computed.

Note that $t_k \leq 3$ with $k = c \log\log n$ for some constant $1 < c$. Moreover, in the kth iteration, P is subdivided into t_k-path-cells and base cells. Thus, in $O(\log\log n)$ iterations, every t-path-cell gets subdivided into base cells. We will show that each iteration takes $O(n)$ time in Section 4.1, which implies that the overall running time for the computation in this section is $O(n \log\log n)$. We will also show that the total complexity of rFVD restricted to the boundaries of all cells in the kth iteration is $O(kn)$ for any $k \in \mathbb{N}$. See Lemma 10.

4.1 Subdividing a t-path-cell into smaller cells

If a t_k-path-cell C is a pseudo-triangle or a lune-cell, C is a base cell and we do not subdivide it further. Otherwise, we subdivide it using the algorithm in this section.

The subdivision consists of three phases. In Phase 1, we subdivide each t_k-path-cell into t_{k+1}-path-cells by a curve connecting at most t_{k+1} vertices of the t_k-path-cell. In Phase 2, we subdivide each t_{k+1}-path-cell further along an arc of rFVD crossing the cell if there is such an arc. In Phase 3, we subdivide cells created in Phase 2 into t_{k+1}-path-cells and lune-cells.

4.1.1 Phase 1. Subdivision by a curve connecting at most t_{k+1} vertices

Let C be a t_k-path-cell computed in the kth iteration. Recall that C consists of at most t_k convex vertices and is simple. Let β be the largest integer satisfying that $\beta\lfloor\sqrt{t_k}\rfloor$ is less than the number of the convex vertices of C. Then we have $\beta \leq \lfloor\sqrt{t_k}\rfloor + 1 = t_{k+1}$.

We choose $\beta+1$ vertices $w_0, w_1, \ldots, w_\beta$ from the convex vertices of C as follows. We choose a convex vertex of C and denote it by w_0. Then we choose the $j\lfloor\sqrt{t_k}\rfloor$th convex vertex of C from w_0 in clockwise order and denote it by w_j for all $j = 1, \ldots, \beta$. We set $w_{\beta+1} = w_0$. Then we construct the closed curve γ_C (or simply γ when C is clear from context) consisting of the geodesic paths $\pi(w_0, w_1), \pi(w_1, w_2), \ldots, (w_\beta, w_0)$. See Figure 1(a). In other words, the closed curve γ_C is the boundary of the geodesic convex hull of $w_0, \ldots, w_\beta$. Note that γ does not cross itself. Moreover, γ is contained in C since C is geodesically convex.

We compute γ in time linear to the number of edges of C as follows. The algorithm computing geodesic paths in [11] takes k source-destination pairs as input, where both sources and destinations are on the boundary of a simple polygon. It returns the geodesic path between the source and the destination for each input pair. For all pairs, computing the geodesic paths takes $O(N+k)$ time in total if k shortest paths do not cross (but possibly overlap) one another, where N is the complexity of the polygon. In our case, the pairs (w_j, w_{j+1}) for $j = 0, \ldots, \beta$ are $\beta+1$ input source-destination pairs. Since the geodesic paths for all input pairs do not cross one another, γ can be computed in $O(\beta + |\partial C|) = O(|\partial C|)$ time. Then we compute $\mathsf{rFVD} \cap \gamma$ in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time using $\mathsf{rFVD} \cap \partial C$ which has already been computed in the kth iteration. We will describe this procedure in Section 4.2.

The curve γ subdivides C into t_{k+1}-path-cells. The set $C \setminus \gamma$ consists of at least $\beta+2$ connected components. Note that the closure of each connected component is a t_{k+1}-path-cell. Moreover, the union of the closures of all connected components is exactly the closure of C since C is simple. These components define the *subdivision of C induced by γ*.

4.1.2 Phase 2. Subdivision along an arc of rFVD

After subdividing C into t_{k+1}-path-cells $C_1, \ldots, C_\delta$ ($\delta \geq \beta+2$) by the curve γ_C, an arc α of rFVD may cross C_j for some $1 \leq j \leq \delta$. In Phase 2, for each arc α crossing C_j, we isolate the subarc $\alpha \cap C_j$ from C_j by creating a new cell which we call an arc-quadrilateral. For an arc-quadrilateral $\square$ created by an arc α, we have $\mathsf{rFVD} \cap \square = \alpha \cap C_j$.

To bound the number of arc-quadrilaterals created in each iteration and the running time for Phase 2, we need the following technical lemma.

▶ **Lemma 7.** *For a geodesic convex polygon C with t convex vertices ($t \in \mathbb{N}$), let γ be a simple closed curve connecting at most t convex vertices of C such that every two consecutive vertices in clockwise order are connected by a geodesic path. Then, for each arc α of* rFVD *with $\alpha \cap C \neq \phi$, α intersects at most three cells in the subdivision of C by γ.*

Since C is geodesically convex, α intersects at most two edges of a cell C in Phase 1, which can be proved in a way similar to the proof of Lemma 7. This implies that $\alpha \cap C_j$ consists of at most two connected components. We say an arc α of rFVD *crosses* a cell C' if exactly two edges of C' intersect α. For example, in Figure 1(c), α crosses C_2 while α does not cross C_3 because there is only one edge of C_3 intersecting α.

First, we find an arc α of rFVD that crosses C_j. Since the points in $\mathsf{rFVD} \cap \partial C_j$ along ∂C_j have already been computed, we can scan them in clockwise order. For all arcs, it can be done in $O(|\mathsf{rFVD} \cap C_j|)$ time by the following lemma.

▶ **Lemma 8.** *All arcs α of* rFVD *crossing C_j can be found in $O(|\mathsf{rFVD} \cap \partial C_j|)$ time. Moreover, for all such α, the pairs $(\triangle_1, \triangle_2)$ of apexed triangles such that $\alpha \cap C_j = \{x \in C_j : g_{\triangle_1}(x) = g_{\triangle_2}(x) > 0\}$ can be found in the same time in total.*

Recall that $\alpha \cap C_j$ consists of at most two connected components. For the case that it consists exactly two connected components, we consider each connected component separately. Thus we show the case that $\alpha \cap C_j$ is connected.

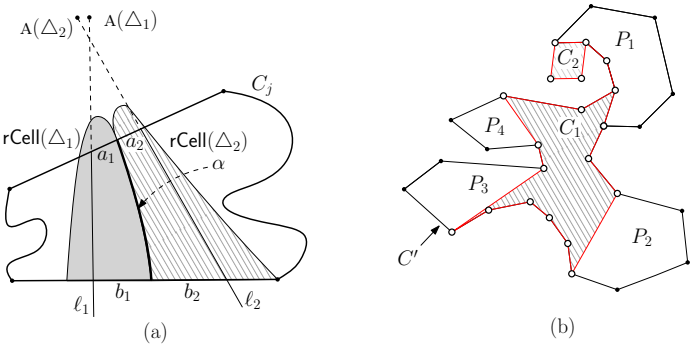


Figure 2 (a) The arc α of rFVD crosses C_j. Thus we isolate α by creating the arc-quadrilateral bounded by ℓ_1, ℓ_2 and ∂C_j. (b) The vertices marked with empty disks are vertices of P while the others are vertices of arc-quadrilaterals lying in $\mathrm{int}(P)$. We subdivide the cell into two t-path-cell C_1, C_2 and four lune-cells $P_1, \ldots, P_4$.

For an arc α crossing C_j, we subdivide C_j further into two cells with t' convex vertices for $t' \leq t_{k+1}$ and one arc-quadrilateral by adding two line segments bounding α such that no arc other than α intersects the arc-quadrilateral. Let $(\triangle_1, \triangle_2)$ be the pair of apexed triangles defining α. Let a_1, b_1 (and a_2, b_2) be the two connected components of $\mathsf{rCell}(\triangle_1) \cap \partial C_j$ (and $\mathsf{rCell}(\triangle_2) \cap \partial C_j$) incident to α such that a_1, a_2 are adjacent to each other and b_1, b_2 are adjacent to each other. See Figure 2(a).

Without loss of generality, we assume that a_1 is closer than b_1 to $\mathrm{A}(\triangle_1)$. Let x be any point on a_1. Then the farthest neighbor of x is the definer of $\triangle_1$. We consider the line ℓ_1 passing through x and the apex of $\triangle_1$. Then the intersection between C_j and ℓ_1 is contained in the closure of $\mathsf{rCell}(\triangle_1)$ by Lemma 4. Similarly, we find the line ℓ_2 passing through the apex of $\triangle_2$ and a point on a_2.

We subdivide C_j into two cells with at most t_{k+1} convex vertices and one arc-quadrilateral by two lines ℓ_1 and ℓ_2. The quadrilateral bounded by the two lines and ∂C_j is an arc-quadrilateral since α is the only arc of rFVD that intersects the quadrilateral. We do this for all arcs crossing some C_j. Note that no arc crosses the resulting cells other than arc-quadrilaterals by the construction. Then the resulting cells with at most t_{k+1} convex vertices and arc-quadrilaterals are the cells in the subdivision of C in the second phase.

4.1.3 Phase 3. Subdivision by a geodesic convex hull

Note that some cell C' with t' convex vertices for $3 < t' \leq t_{k+1}$ created in Phase 2 might be neither a t'-path-cell nor a base cell. Such a cell has some vertices in $\mathrm{int}(P)$ and some vertices of P as its vertices. In Phase 3, we subdivide such cells further into t'-path-cells and base cells.

To subdivide C' into t_{k+1}-path-cells and base cells, we first compute the geodesic convex hull CH of the vertices of C' which are vertices of P in time linear to the number of edges in C' using the algorithm for computing k-pair shortest paths in [11]. Consider the connected components of $C' \setminus \partial\mathsf{CH}$. There are two types of the connected components. A connected component of the first type is enclosed by a simple closed curve which is part of $\partial\mathsf{CH}$. For example, C_1 and C_2 in Figure 2(b) belong to this type. A connected component of the second type is enclosed by a subchain of $\partial\mathsf{CH}$ from u to w in clockwise order and a subchain of $\partial C'$ from w to u in counterclockwise order for some $u, w \in \partial P$. For example, P_i in Figure 2(b) belongs to the second type for $i = 1, \ldots, 4$.

By the construction, a connected component belonging to the first type has all its vertices on ∂P. Moreover, it has at most t' convex vertices since C' has t' convex vertices. Therefore, the closure of a connected component of $C' \setminus \partial\mathsf{CH}$ belonging to the first type is a t'-path-cell.

Every vertex of C' lying in $\mathrm{int}(P)$ is convex with respect to C' by the construction of C'. Thus, for a connected component P' belonging to the second type, the part of $\partial P'$ from $\partial C'$ is a convex chain with respect to P'. Moreover, the part of $\partial P'$ from $\partial\mathsf{CH}$ is the geodesic path between two points, thus it is a concave chain with respect to P'. Therefore, the closure of a connected component belonging to the second type is a lune-cell.

Since C' is a simple polygon, the union of the closures of all connected components of $C' \setminus \mathsf{CH}$ is exactly the closure of C'. The closures of all connected components of the first and the second types are t_{k+1}-path-cells and lune-cells created in the last phase of the $(k+1)$th iteration, respectively. We compute the t_{k+1}-path-cells and the lune-cells subdivided by $\partial\mathsf{CH}$. Then, we compute $\mathsf{rFVD} \cap \partial\mathsf{CH}$ using the procedure in Section 4.2. The resulting t_{k+1}-path-cells and base cells are the final decomposition of C of the $(k+1)$th iteration.

4.1.4 Analysis of the complexity

We first bound the complexity of the refined farthest-point geodesic Voronoi diagram restricted to the boundary of the cells in each iteration. The following technical lemma is used to bound the complexity.

▶ **Lemma 9.** *An arc α of* rFVD *intersects at most nine t_k-path-cells and $O(k)$ base cells at the end of the kth iteration. Moreover, there are at most three t_k-path-cells that α intersects but does not cross at the end of the kth iteration.*

Now we are ready to bound the complexities of the cells and rFVD restricted to the cells in each iteration. Then we finally prove that the running time of the algorithm in this section is $O(n \log\log n)$.

▶ **Lemma 10.** *At the end of the kth iteration, we have $\sum_{C:\text{a } t_k\text{-path-cell}} |\mathsf{rFVD} \cap \partial C| = O(n)$, $\sum_{C:\text{a } t_k\text{-path-cell}} |\partial C| = O(n)$, $\sum_{T:\text{a base cell}} |\mathsf{rFVD} \cap \partial T| = O(kn)$, and $\sum_{T:\text{a base cell}} |\partial T| = O(kn)$.*

Proof. Let α be an arc of rFVD. The first and the third complexity bounds hold by Lemma 9 and the fact that the number of the arcs of rFVD is $O(n)$.

The second complexity bound holds since the set of all edges of the t_k-path-cells is a subset of the chords in some triangulation of P. Note that any triangulation of P has $O(n)$ chords. Moreover, each chord is incident to at most two t_k-path-cells.

For the last complexity bound, the number of edges of base cells whose endpoints are vertices of P is $O(n)$ since they are chords in some triangulation of P. Thus we count the number of edges of base cells which are not incident to vertices of P. In Phase 1, we do not create any such edge. In Phase 2, we create at most $O(1)$ such edges whenever we create one arc-quadrilateral. All edges created in Phase 3 have their endpoints from the vertex set of P. Therefore, the total number of the edges of all base cells is asymptotically bounded by the number of arc-quadrilaterals, which is $O(kn)$. ◀

▶ **Corollary 11.** *In $O(\log\log n)$ iterations, P is subdivided into $O(n \log\log n)$ base cells.*

▶ **Lemma 12.** *The subdivision in each iteration can be done in $O(n)$ time.*

4.2 Computing rFVD restricted to the boundary of a t-path-cell

Recall that the bottom sides of all apexed triangles are interior-disjoint. Moreover, the union of them is ∂P. In this section, we describe a procedure to compute $\mathsf{rFVD} \cap \gamma$ in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time once $\mathsf{rFVD} \cap \partial C$ is computed. Recall that γ is a closed curve connecting consecutive points of every t_{k+1}th convex vertices of C in clockwise order.

If $\mathsf{rCell}(\triangle) \cap \gamma \neq \phi$ for an apexed triangle $\triangle$, then we have $\mathsf{rCell}(\triangle) \cap \partial C \neq \phi$. Thus, we consider only the apexed triangles $\triangle$ with $\mathsf{rCell}(\triangle) \cap \partial C \neq \phi$. Let $\mathcal{L}$ be the list of all such apexed triangles sorted along ∂P with respect to their bottom sides.

Consider a line segment ab contained in P. Without loss of generality, we assume that ab is horizontal and a lies to the left of b. Let $\triangle_a$ and $\triangle_b$ be the apexed triangles which maximize $g_{\triangle_a}(a)$ and $g_{\triangle_b}(b)$, respectively. If there is a tie by more than one apexed triangles, we choose an arbitrary one of them. With the two apexed triangles, we define two sorted lists $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$. Let $\mathcal{L}_{ab}$ be the sorted list of the apexed triangles in $\mathcal{L}$ which intersect ab and whose bottom sides lie from the bottom side of $\triangle_a$ to the bottom side of $\triangle_b$ in clockwise order along ∂P. Similarly, let $\mathcal{L}_{ba}$ be the sorted list of the apexed triangles in $\mathcal{L}$ which intersect ab and whose bottom sides lie from the bottom side of $\triangle_b$ to the bottom side of $\triangle_a$ in clockwise order along ∂P.

The following lemma together with Section 4.2.1 gives a procedure to compute $\mathsf{rFVD} \cap ab$. The procedure is similar to the procedure for computing $\mathsf{rFVD} \cap \partial P$ in Section 3.1.

▶ **Lemma 13.** *Let C be a geodesic convex polygon and a, b be two points with $ab \subset C$. Given the two sorted lists $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$, $\mathsf{rFVD} \cap ab$ can be computed in $O(|\mathcal{L}_{ab}| + |\mathcal{L}_{ba}|)$ time.*

Since an apexed triangle intersects at most two edges of γ, we can compute $\mathsf{rFVD} \cap \gamma$ in $O(|\mathcal{L}|) = O(|\mathsf{rFVD} \cap \partial C|)$ time once we have $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ.

4.2.1 Computing $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ

In this section, we show how to compute $\mathcal{L}_{ab}$ and $\mathcal{L}_{ba}$ for all edges ab of γ in $O(|\mathcal{L}| + |\partial C|)$ time. Recall that all endpoints of the geodesic paths bounding the t-path-cell C lie in ∂P. Let ab be an edge of γ, where b is the clockwise neighbor of a. The edge ab is a chord of P and divides P into two subpolygons such that $\gamma \setminus ab$ is contained in one of the subpolygons. Let $P_1(ab)$ be the subpolygon containing $\gamma \setminus ab$ and $P_2(ab)$ be the other subpolygon. For an apexed triangle in $\mathcal{L}_{ab}$, its bottom side lies in $\partial P_2(ab)$ and its apex lies in $\partial P_1(ab)$. On the other hand, for an apexed triangle in $\mathcal{L}_{ba}$, its bottom side lies in $\partial P_1(ab)$ and its apex lies in $\partial P_2(ab)$. Moreover, if its apex lies in $P_j(ab)$, then so does its definer for $j = 1, 2$. By the construction, $P_2(ab)$ and $P_2(e')$ are disjoint in their interior for any edge $e' \in \gamma \setminus \{ab\}$.

We compute $\mathcal{L}_{ab}$ for all edges ab in γ as follows. Initially, $\mathcal{L}_{ab}$ for all edges ab are set to ϕ. We update the list by scanning the apexed triangles in $\mathcal{L}$ from the first to the end. When we handle an apexed triangle $\triangle \in \mathcal{L}$, we first find the edge ab of γ such that $P_2(ab)$ contains the bottom side of $\triangle$ and check whether $\triangle \cap ab = \phi$. If it is nonempty, we append $\triangle$ to $\mathcal{L}_{ab}$. Otherwise, we do nothing. Then we handle the apexed triangle next to $\triangle$. For $\mathcal{L}_{ba}$, we do analogously, except that we find the edge ab of γ such that $P_2(ab)$ contains the definer of $\triangle$.

Note that any three apexed triangles $\triangle_1, \triangle_2, \triangle_3 \in \mathcal{L}$ appear on $\mathcal{L}$ in the order of their definers (and their bottom sides) appearing on ∂P. Thus to find the edge ab of γ such that $P_2(ab)$ contains the definer (or the bottom side) of $\triangle$, it is sufficient to check at most two edges; the edge e' such that $P_2(e')$ contains the bottom side of the apexed triangle previous to $\triangle$ in $\mathcal{L}$ and the clockwise neighbor of e'. Therefore, this procedure takes in $O(|\mathcal{L}|)$ time.

The following lemmas summarize this section.

▶ **Lemma 14.** *Let C be a t-path-cell and γ be a simple closed curve connecting at most t convex vertices of C lying on ∂P such that two consecutive vertices in clockwise order are connected by a geodesic path. Once $\mathsf{rFVD} \cap \partial C$ is computed, $\mathsf{rFVD} \cap \gamma$ can be computed in $O(|\mathsf{rFVD} \cap \partial C| + |\partial C|)$ time.*

▶ **Lemma 15.** *Each iteration takes $O(n)$ time and the algorithm in this section terminates in $O(\log \log n)$ iterations. Thus the running time of the algorithm in this section is $O(n \log \log n)$.*

5 Computing rFVD in the interior of a base cell

In this section, we consider a base cell T which is a lune-cell or a pseudo-triangle. Assume that $\mathsf{rFVD} \cap \partial T$ has already been computed. We extend $\mathsf{rFVD} \cap \partial T$ into the interior of the cell T in $O(|\mathsf{rFVD} \cap \partial T|)$ time.

To make the description easier, we first make two assumptions: (1) for any apexed triangle $\triangle$, $\mathsf{rCell}(\triangle) \cap \partial T$ is connected and contains the bottom side of $\triangle$, and (2) T is a lune-cell. In the full version of this paper, we show how to avoid the assumptions by subdividing each base cell and trimming each apexed triangle.

5.1 Definition for a new distance function

Without loss of generality, we assume that the bottom side of T is horizontal. We bound the domain by introducing a box B containing T. To apply the algorithm for computing the abstract Voronoi diagram in [5, 9], we need to define a new distance function $f_\triangle : B \to \mathbb{R}$ since $g_\triangle$ is not continuous. Imagine that we partition B into five regions with respect to an apexed triangle $\triangle$. We will define $f_\triangle$ as a function consisting of at most five algebraic functions each of whose domains corresponds to a partitioned region in B.

Consider five line segments ℓ_1, ℓ_2, ℓ_3, ℓ_4 and ℓ_5 such that their common endpoint is $\mathrm{A}(\triangle)$ and the other endpoints lie on ∂B. The line segments ℓ_1 and ℓ_2 contain the left and the right corners of $\triangle$, respectively. The line segments ℓ_3 and ℓ_5 are orthogonal to ℓ_2 and ℓ_1, respectively. The line segment ℓ_4 is contained in the line bisecting the angle of $\triangle$ at $\mathrm{A}(\triangle)$ but it does not intersect $\mathrm{int}(\triangle)$.

Then B is partitioned by these five line segments into five regions. We denote the region bounded by ℓ_1 and ℓ_2 which contains $\triangle$ by $G_{\mathrm{in}}(\triangle)$. Note that $\mathrm{D}(\triangle) \notin G_{\mathrm{in}}(\triangle)$ if $\mathrm{D}(\triangle) \neq \mathrm{A}(\triangle)$. The remaining four regions are denoted by $G_{\mathrm{Lside}}(\triangle)$, $G_{\mathrm{Ltop}}(\triangle)$, $G_{\mathrm{Rtop}}(\triangle)$, and $G_{\mathrm{Rside}}(\triangle)$ in the clockwise order from $G_{\mathrm{in}}(\triangle)$ along ∂B.

For a point $x \in G_{\mathrm{Lside}}(\triangle) \cup G_{\mathrm{Ltop}}(\triangle)$, let $\hat{x}_\triangle$ be the orthogonal projection of x on the line containing ℓ_1. Similarly, for a point $x \in G_{\mathrm{Rside}}(\triangle) \cup G_{\mathrm{Rtop}}(\triangle) \setminus \ell_4$, let $\hat{x}_\triangle$ be the orthogonal projection of x on the line containing ℓ_2. For a point $x \in G_{\mathrm{in}}(\triangle)$, we set $\hat{x}_\triangle = x$.

We define a new distance function $f_\triangle : B \to \mathbb{R}$ for each apexed triangle $\triangle$ with $\mathsf{rCell}(\triangle) \cap \partial T \neq \phi$ as follows.

$$f_\triangle(x) = \begin{cases} d(\mathrm{A}(\triangle), \mathrm{D}(\triangle)) - \|\hat{x}_\triangle - \mathrm{A}(\triangle)\| & \text{if } x \in G_{\mathrm{Ltop}}(\triangle) \cup G_{\mathrm{Rtop}}(\triangle), \\ d(\mathrm{A}(\triangle), \mathrm{D}(\triangle)) + \|\hat{x}_\triangle - \mathrm{A}(\triangle)\| & \text{otherwise.} \end{cases}$$

Note that $f_\triangle$ is continuous on B. Each contour curve, that is a set of points with the same function value, consists of two line segments and at most one circular arc.

Here, we assume that there is no pair $(\triangle_1, \triangle_2)$ of apexed triangles such that two sides, one from $\triangle_1$ and the other from $\triangle_2$, are parallel. If there exists such a pair, the contour curves for two apexed triangles may overlap. In the full version, we show how to avoid the assumption by slightly perturbing the distance function.

5.2 An algorithm for computing rFVD $\cap\, T$

To compute the farthest-point geodesic Voronoi diagram restricted to T, we apply the algorithms in [5, 9] with this new distance function, which computes the abstract Voronoi diagram in a domain where each site has a unique cell touching the boundary of the domain. While the algorithms in [5, 9] compute the abstract nearest-point Voronoi diagram, they can be used to compute the farthest-point Voronoi diagram. These algorithms are generalizations of the linear-time algorithm in [1], which computes the farthest-point and the nearest-point Voronoi diagram of points in convex position.

In the abstract Voronoi diagram, no explicit sites or distance functions are given. Instead, for any pair of sites s and s', the open domains $D(s, s')$ and $D(s', s)$ are given. Let A be the set of all apexed triangles with $\mathsf{rFVD} \cap \partial T$. In our problem, we regard the apexed triangles in A as the sites and B as the domain for the abstract Voronoi diagram. For two apexed triangles $\triangle_1$ and $\triangle_2$ in A, we define the open domain $D(\triangle_1, \triangle_2)$ as the set $\{x \in B : f_{\triangle_1}(x) > f_{\triangle_2}(x)\}$. We denote the abstract Voronoi diagram for the apexed triangles by aFVD and the cell of $\triangle$ on aFVD by $\mathsf{aCell}(\triangle)$.

Here, we need to show that the distance function we define in Section 5.1 satisfies the followings for any subset A' of A. A proof can be found in the full version of this paper.

1. For any two apexed triangles $\triangle_1$ and $\triangle_2$ in A, the set $\{x \in B : f_{\triangle_1}(x) = f_{\triangle_2}(x)\}$ is a curve with endpoints on ∂B. The curve consists of $O(1)$ pieces of algebraic curves.
2. Each apexed triangle $\triangle$ in A' has exactly one connected and nonempty cell in the abstract Voronoi diagram of A'.
3. Each point in B belongs to the closure of an abstract Voronoi cell.
4. The abstract Voronoi diagrams of A and A' form a tree and a forest, respectively.

Thus, we can compute aFVD using the algorithms in [5, 9]. The abstract Voronoi diagram restricted to T is exactly the refined farthest-point geodesic Voronoi diagram restricted to T. Note that we already have the abstract Voronoi diagram restricted to ∂T which coincides with the refined farthest-point geodesic Voronoi diagram restricted to ∂T. After computing aFVD on B, we traverse aFVD and extract aFVD lying inside T.

▶ **Lemma 16.** *Given a base cell T constructed by the subdivision algorithm in Section 4.1, $\mathsf{rFVD} \cap T$ can be computed in $O(|\mathsf{rFVD} \cap \partial T| + |\partial T|)$ time.*

▶ **Theorem 17.** *The farthest-point geodesic Voronoi diagram of the vertices of a simple n-gon can be computed in $O(n \log\log n)$ time.*

6 A set of sites on the boundary

In this section, we show that the results presented above are general enough to work when the set S is an arbitrary set of sites contained in the boundary of P.

Since S is a subset of sites contained in ∂P, we can assume without loss of generality that all sites of S are vertices of P by splitting the edges where they lie on. In this section, we decompose the boundary of P into chains of consecutive vertices that share the same S-farthest neighbor and edges of P whose endpoints have distinct S-farthest neighbors. The following lemma is a counterpart of Lemma 1. Lemma 1 is the only place where it was assumed that S is the set of vertices of P.

▶ **Lemma 18.** *Given a set S of m sites contained in ∂P, we can compute the S-farthest neighbor of each vertex of P in $O(n + m)$ time.*

Proof. Let $w : P \to \mathbb{R}$ be a real valued function on the vertices of P such that for each vertex v of P, $w(v) = D_P$ if $v \in S$, and $w(v) = 0$ otherwise, where D_P is any fixed constant larger than the diameter of P.

For each vertex $p \in P$, we want to identify $\text{N}(p)$. To this end, we define a new distance function $d^* : P \times P \to \mathbb{R}$ such that for any two points u and v of P, $d^*(u, v) = d(u, v) + w(u) + w(v)$. Using a result from Hershberger and Suri [8, Section 6.1 and 6.3], in $O(n + m)$ time we can compute the farthest neighbor of each vertex of P with respect to d^*.

By the definition of the function w, the maximum distance from any vertex of P is achieved at a site of S. Therefore, the farthest neighbor from a vertex v of P with respect to d^* is indeed the S-farthest neighbor, $\text{N}(v)$, of v. ◀

▶ **Theorem 19.** *The farthest-point geodesic Voronoi diagram of m points on the boundary of a simple n-gon can be computed in $O((n + m) \log \log n)$ time.*

References

1. Alok Aggarwal, Leonidas J Guibas, James Saxe, and Peter W Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete & Computational Geometry*, 4(6):591–604, 1989.
2. Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, Matias Korman, and Eunjin Oh. A linear-time algorithm for the geodesic center of a simple polygon. In *Proceedings of the 31st Symposium on Compututaional Geometry, SoCG*, pages 209–223, 2015.
3. Boris Aronov, Steven Fortune, and Gordon Wilfong. The furthest-site geodesic Voronoi diagram. *Discrete & Computational Geometry*, 9(3):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. Cecilia Bohler, Rolf Klein, and Chih-Hung Liu. Forest-like abstract Voronoi diagrams in linear time. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG*, pages 133–141, 2014.
6. B Chazelle. A theorem on polygon cutting with applications. In *Proceedings 23rd Annual Symposium on Foundations of Computer Science, FOCS*, pages 339–349, 1982.
7. 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.
8. John Hershberger and Subhash Suri. Matrix searching with the shortest-path metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997.
9. Rolf Klein and Andrzej Lingas. Hamiltonian abstract Voronoi diagrams in linear time. In *Proceedings of the 5th International Symposium on Algorithms and Computation ISAAC*, pages 11–19, 1994.
10. J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*, pages 633–701. Elsevier, 2000.
11. Evanthia Papadopoulou. k-pairs non-crossing shortest paths in a simple polygon. *International Journal of Computational Geometry and Applications*, 9(6):533–552, 1999.
12. Richard Pollack, Micha Sharir, and Günter Rote. Computing the geodesic center of a simple polygon. *Discrete & Computational Geometry*, 4(6):611–626, 1989.
13. Subhash Suri. Computing geodesic furthest neighbors in simple polygons. *Journal of Computer and System Sciences*, 39(2):220–235, 1989.

Avoiding the Global Sort: A Faster Contour Tree Algorithm*

Benjamin Raichel[1] and C. Seshadhri[2]

1 **Department of Computer Science, University of Texas at Dallas, Richardson, USA**
benjamin.raichel@utdallas.edu

2 **Department of Computer Science, University of California, Santa Cruz, USA**
scomandu@ucsc.edu

Abstract

We revisit the classical problem of computing the *contour tree* of a scalar field $f : \mathbb{M} \to \mathbb{R}$, where $\mathbb{M}$ is a triangulated simplicial mesh in $\mathbb{R}^d$. The contour tree is a fundamental topological structure that tracks the evolution of level sets of f and has numerous applications in data analysis and visualization.

All existing algorithms begin with a global sort of at least all critical values of f, which can require (roughly) $\Omega(n \log n)$ time. Existing lower bounds show that there are pathological instances where this sort is required. We present the first algorithm whose time complexity depends on the contour tree structure, and avoids the global sort for non-pathological inputs. If C denotes the set of critical points in $\mathbb{M}$, the running time is roughly $O(\sum_{v \in C} \log \ell_v)$, where ℓ_v is the depth of v in the contour tree. This matches all existing upper bounds, but is a significant asymptotic improvement when the contour tree is short and fat. Specifically, our approach ensures that any comparison made is between nodes that are either adjacent in $\mathbb{M}$ or in the same descending path in the contour tree, allowing us to argue strong optimality properties of our algorithm.

Our algorithm requires several novel ideas: partitioning $\mathbb{M}$ in well-behaved portions, a local growing procedure to iteratively build contour trees, and the use of heavy path decompositions for the time complexity analysis.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, I.1.2 Algorithms, I.3.5 Computational Geometry and Object Modeling

Keywords and phrases contour trees, computational topology, computational geometry

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.57

1 Introduction

Geometric data is often represented as a function $f : \mathbb{R}^d \to \mathbb{R}$. Typically, a finite representation is given by considering f to be piecewise linear over some triangulated mesh (i.e. simplicial complex) $\mathbb{M}$ in $\mathbb{R}^d$. *Contour trees* are a topological structure used to represent and visualize the function f. It is convenient to think of f as a simplicial complex sitting in $\mathbb{R}^{d+1}$, with the last coordinate (i.e. height) given by f. Imagine sweeping the hyperplane $x_{d+1} = h$ with h going from $+\infty$ to $-\infty$. At every instance, the intersection of this plane with f gives a set of connected components, the *contours* at height h. As the sweeping proceeds various events occur: new contours are created or destroyed, contours merge into

* The full updated version of this paper is available on the arXiv [20].


© Benjamin Raichel and C. Seshadhri;
licensed under Creative Commons License CC-BY

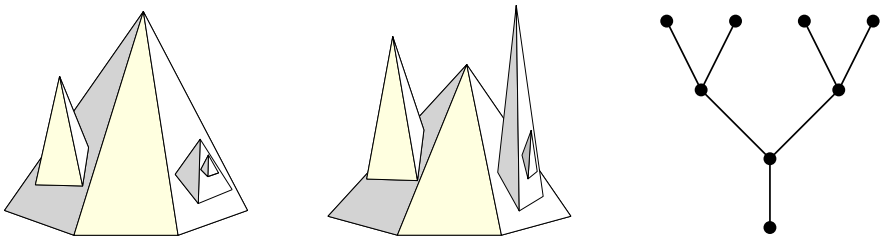


Figure 1 Two surfaces with different orderings of the maxima, but the same contour tree.

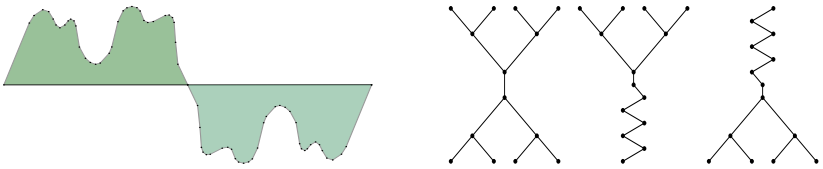


Figure 2 On left, a surface with a balanced contour tree, but whose join and split trees have long tails. On right (from left to right), the contour, join and split trees.

each other or split into new components, contours acquire or lose handles. The contour tree is a concise representation of all these events. Throughout we follow the definition of contour trees from [24] which includes all changes in topology. For $d > 2$, some subsequent works, such as [7], only include changes in the number of components.

If f is smooth, all points where the gradient of f is zero are *critical points*. These points are the "events" where the contour topology changes and form the vertices of the contour tree. An edge of the contour tree connects two critical points if one event immediately "follows" the other as the sweep plane makes its pass. (We provide formal definitions later.) Figure 1 and Figure 2 show examples of simplicial complexes, with heights and their contour trees. Think of the contour tree edges as pointing downwards. Leaves are either maxima or minima, and internal nodes are either "joins" or "splits".

Consider $f : \mathbb{M} \to \mathbb{R}$, where $\mathbb{M}$ is a triangulated mesh with n vertices, N faces in total, and $t \leq n$ critical points. (We assume that $f : \mathbb{M} \to \mathbb{R}$ a linear interpolant over distinct valued vertices, where the contour tree T has maximum degree 3. The degree assumption simplifies the presentation, and is commonly made [24].) A fundamental result in this area is the algorithm of Carr, Snoeyink, and Axen to compute contour trees, which runs in $O(n \log n + N\alpha(N))$ time [7] (where $\alpha(\cdot)$ denotes the inverse Ackermann function). In practical applications, N is typically $\Theta(n)$ (certainly true for $d = 2$). The most expensive operation is an initial sort of all the vertex heights. Chiang *et al.* build on this approach to get a faster algorithm that only sorts the critical vertices, yielding a running time of $O(t \log t + N)$ [8]. Common applications for contour trees involve turbulent combustion or noisy data, where the number of critical points is likely to be $\Omega(n)$. There is a worst-case lower bound of $\Omega(t \log t)$ by Chiang *et al.* [8], based on a construction of Bajaj *et al.* [1].

All previous algorithms begin by sorting (at least) the critical points. Can we beat this sorting bound for certain instances, and can we characterize which inputs are hard? Intuitively, points that are incomparable in the contour tree do not need to be compared. Look at Figure 1 to see such an example. All previous algorithms waste time sorting all the maxima. Also consider the surface of Figure 2. The final contour tree is basically two binary trees joined at their roots, and we do not need the entire sorted order of critical points to construct the contour tree.

Our main result gives an affirmative answer. Remember that we can consider the contour tree as directed from top to bottom. For any node v in the tree, let ℓ_v denote the length of the longest directed path passing through v.

▶ **Theorem 1.** *Consider a simplicial complex $f : \mathbb{M} \to \mathbb{R}$, described as above, and denote the contour tree by T with vertex set (the critical points) $C(T)$. There exists an algorithm to compute T in $O(\sum_{v \in C(T)} \log \ell_v + t\alpha(t) + N)$ time. Moreover, this algorithm only compares function values at pairs of vertices that are ancestor-descendant in T or adjacent in $\mathbb{M}$.*

Essentially, the "run time per critical point" is the height/depth of the point in the contour tree. This bound immediately yields a run time of $O(t \log D + t\alpha(t) + N)$, where D is the diameter of the contour tree. This is a significant asymptotic improvement for short and fat contour trees. For example, if the tree is balanced, then we get a bound of $O(t \log \log t)$. Even if T contains a long path of length $O(t/\log t)$, but is otherwise short, we get the improved bound of $O(t \log \log t)$.

1.1 A refined bound with optimality properties

Theorem 1 is a direct corollary of a stronger but more cumbersome theorem.

▶ **Definition 2.** For a contour tree T, a *leaf path* is any path in T containing a leaf, which is also monotonic in the height values of its vertices. Then a *path decomposition*, $P(T)$, is a partition of the vertices of T into a set of vertex disjoint leaf paths.

▶ **Theorem 3.** *There is a deterministic algorithm to compute the contour tree, T, whose running time is $O(\sum_{p \in P(T)} |p| \log |p| + t\alpha(t) + N)$, where $P(T)$ is a specific path decomposition (constructed implicitly by the algorithm). The number of comparisons made is $O(\sum_{p \in P(T)} |p| \log |p| + N)$. In particular, comparisons are only made between pairs of vertices that are ancestor-descendant in T or adjacent in $\mathbb{M}$.*

Note that Theorem 1 is a direct corollary of this statement. For any v, ℓ_v is at least the length of the path in $P(T)$ that contains v. This bound is strictly stronger, since for any balanced contour tree, the run time bound of Theorem 3 is $O(t\alpha(t) + N)$, and $O(N)$ comparisons are made. (Since for a balanced tree, one can show $\sum_{p \in P(T)} |p| \log |p| = O(t)$.)

The bound of Theorem 3 may seem artificial, since it actually depends on the $P(T)$ that is implicitly constructed by the algorithm. Nonetheless, we prove that the algorithm of Theorem 3 has strong optimality properties. For convenience, fix some value $t = \Omega(N)$, and consider the set of terrains ($d = 2$) with t critical points. The bound of Theorem 3 takes values ranging from t to $t \log t$. Consider some $\gamma \in [t, t \log t]$, and consider the set of terrains where the algorithm makes γ comparisons. Then *any algorithm* must make roughly γ comparisons in the worst-case over this set. (Further details are in the full version [20].)

▶ **Theorem 4.** *There exists some absolute constant α such that the following holds. For sufficiently large t and any $\gamma \in [t, t \log t]$, consider the set $\boldsymbol{F}_\gamma$ of terrains with t critical points such that the number of comparisons made by the algorithm of Theorem 3 on these terrains is in $[\gamma, \alpha\gamma]$. Any algebraic decision tree that correctly computes the contour tree on all of $\boldsymbol{F}_\gamma$ has a worst case running time of $\Omega(\gamma)$.*

1.2 Previous Work

Contour trees were first used to study terrain maps by Boyell and Ruston, and Freeman and Morse [3, 13]. Contour trees have been applied in analysis of fluid mixing, combustion

simulations, and studying chemical systems [15, 4, 2, 5, 16]. Carr's thesis [6] gives various applications of contour trees for data visualization and is an excellent reference for contour tree definitions and algorithms.

The first formal result was an $O(N \log N)$ time algorithm for functions over 2D meshes and an $O(N^2)$ algorithm for higher dimensions, by van Kreveld *et al.* [24]. Tarasov and Vyalya [22] improved the running time to $O(N \log N)$ for the 3D case. The influential paper of Carr *et al.* [7] improved the running time for all dimensions to $O(n \log n + N\alpha(N))$. Pascucci and Cole-McLaughlin [18] provided an $O(n + t \log n)$ time algorithm for 3-dimensional structured meshes. Chiang *et al.* [8] provide an unconditional $O(N + t \log t)$ algorithm.

Contour trees are a special case of Reeb graphs, a general topological representation for real-valued functions on any manifold. Algorithms for computing Reeb graphs is an active topic of research [21, 9, 19, 10, 14, 17], where two results explicitly reduce to computing contour trees [23, 11].

2 Contour tree basics

We detail the basic definitions about contour trees, following the terminology of Chapter 6 of Carr's thesis [6]. All our assumptions and definitions are standard for results in this area, though there is some variability in notation. The input is a continuous piecewise-linear function $f : \mathbb{M} \to \mathbb{R}$, where $\mathbb{M}$ is a simply connected and fully triangulated simplicial complex in $\mathbb{R}^d$, except for specially designated *boundary facets*. So f is explicitly defined only on the vertices of $\mathbb{M}$, and all other values are obtained by linear interpolation.

We assume that the boundary values satisfy a special property. This is mainly for convenience in presentation.

▶ **Definition 5.** The function f is *boundary critical* if the following holds. Consider a boundary facet F. All vertices of F have the same function value. Furthermore, all neighbors of vertices in F, which are not also in F itself, either have all function values strictly greater than or all function values strictly less than the function value at F.

This is convenient, as we can now assume that f is defined on $\mathbb{R}^d$. Any point inside a boundary facet has a well-defined height, including the infinite facet, which is required to be a boundary facet. However, we allow for other boundary facets, to capture the resulting surface pieces after our algorithm makes a horizontal cut.

We think of the dimension d, as constant, and assume that $\mathbb{M}$ is represented in a data structure that allows constant-time access to neighboring simplices in $\mathbb{M}$ (e.g. incidence graphs [12]). (This is analogous to a doubly connected edge list, but for higher dimensions.) Observe that $f : \mathbb{M} \to \mathbb{R}$ can be thought of as a d-dimensional simplicial complex living in $\mathbb{R}^{d+1}$, where $f(x)$ is the "height" of a point $x \in \mathbb{M}$, which is encoded in the representation of $\mathbb{M}$. Specifically, rather than writing our input as $(\mathbb{M}, f)$, we abuse notation and typically just write $\mathbb{M}$ to denote the lifted complex.

▶ **Definition 6.** The *level set* at value h is the set $\{x | f(x) = h\}$. A *contour* is a connected component of a level set. An *h-contour* is a contour where f-values are h.

Note that a contour that does not contain a boundary is itself a simplicial complex of one dimension lower, and is represented (in our algorithms) as such. We let δ and ε denote infinitesimals. Let $B_\varepsilon(x)$ denote a ball of radius ε around x, and let $f|B_\varepsilon(x)$ be the restriction of f to $B_\varepsilon(x)$.

▶ **Definition 7.** The *Morse up-degree* of x is the number of $(f(x)+\delta)$-contours of $f|B_\varepsilon(x)$ as $\delta, \varepsilon \to 0^+$. The *Morse down-degree* is the number of $(f(x)-\delta)$-contours of $f|B_\varepsilon(x)$ as $\delta, \varepsilon \to 0^+$.

A *regular* point has both Morse up-degree and down-degree 1. A *maximum* has Morse up-degree 0, while a *minimum* has Morse down-degree 0. A *Morse Join* has Morse up-degree strictly greater than 1, while a *Morse Split* has Morse down-degree strictly greater than 1. Non-regular points are called *critical*.

The set of critical points is denoted by $\mathcal{V}(f)$. Because f is piecewise-linear, all critical points are vertices in $\mathbb{M}$. A value h is called *critical*, if $f(v) = h$, for some $v \in \mathcal{V}(f)$. A contour is called *critical*, if it contains a critical point, and it is called *regular* otherwise.

The critical points are exactly where the topology of level sets change. By assuming that our simplicial complex is boundary critical, the vertices on a given boundary are either collectively all maxima or all minima. We abuse notation and refer to this entire set of vertices as a maximum or minimum.

▶ **Definition 8.** Two regular contours ψ and ψ' are *equivalent* if there exists an f-monotone path p connecting a point in ψ to ψ', such that no $x \in p$ belongs to a critical contour.

This equivalence relation gives a set of *contour classes*. Every such class maps to intervals of the form $(f(x_i), f(x_j))$, where x_i, x_j are critical points. Such a class is said to be created at x_i and destroyed at x_j.

▶ **Definition 9.** The *contour tree* is the graph on vertex set $\mathcal{V} = \mathcal{V}(f)$, where edges are formed as follows. For every contour class that is created at v_i and destroyed v_j, there is an edge (v_i, v_j). (Conventionally, edges are directed from higher to lower function value.)

We denote the contour tree of $\mathbb{M}$ by $\mathcal{C}(\mathbb{M})$. The corresponding node and edge sets are denoted as $\mathcal{V}(\cdot)$ and $\mathcal{E}(\cdot)$. It is not immediately obvious that this graph is a tree, but alternate definitions of the contour tree in [7] imply this is a tree. Since this tree has height values associated with the vertices, we can talk about up-degrees and down-degrees in $\mathcal{C}(\mathbb{M})$. Similar to [24] (among others), multi-saddles are treated as a set of ordinary saddles, which can be realized via vertex unfolding (which can increase surface complexity if multi-saddle degrees are allowed to be super-constant). Therefore, to simplify the presentation, for the remainder of the paper up and down-degrees are at most 2, and total degree is at most 3.

See the full version [20] for further technical remarks on the above definitions.

3 Divide and conquer through contour surgery

The cutting operation: We define a "cut" operation on $f : \mathbb{M} \to \mathbb{R}$ that cuts along a regular contour to create a new simplicial complex with an added boundary. Given a contour ϕ, roughly speaking, this constructs the simplicial complex $\mathbb{M} \setminus \phi$. We will always enforce the condition that ϕ never passes through a vertex of $\mathbb{M}$. Again, we use ε for an infinitesimally small value. We denote ϕ^+ (resp. ϕ^-) to be the contour at value $f(\phi) + \varepsilon$ (resp. $f(\phi) - \varepsilon$), which is at distance ε from ϕ.

An h-contour is achieved by intersecting $\mathbb{M}$ with the hyperplane $x_{d+1} = h$ and taking a connected component. (Think of the $d+1$-dimension as height.) Given some point x on an h-contour ϕ, we can walk along $\mathbb{M}$ from x to determine ϕ. We can "cut" along ϕ to get a new (possibly) disconnected simplicial complex $\mathbb{M}'$. This is achieved by splitting every face F that ϕ intersects into an "upper" face and "lower" face. Algorithmically, we cut F with ϕ^+ and take everything above ϕ^+ in F to make the upper face. Analogously, we cut with ϕ^-

to get the lower face. The faces are then triangulated to ensure that they are all simplices. This creates the two new boundaries ϕ^+ and ϕ^-, and we maintain the property of constant f-value at a boundary.

Note that by assumption ϕ cannot cut a boundary face, and moreover all non-boundary faces have constant size. Therefore, this process takes time linear in $|\phi|$, the number of faces ϕ intersects. This new simplicial complex is denoted by $\texttt{cut}(\phi, \mathbb{M})$. We now describe a high-level approach to construct $\mathcal{C}(\mathbb{M})$ using this cutting procedure.

$\texttt{surgery}(\mathbb{M}, \phi)$
1. Let $\mathbb{M}' = \texttt{cut}(\mathbb{M}, \phi)$.
2. Construct $\mathcal{C}(\mathbb{M}')$ and let A, B be the nodes corresponding to the new boundaries created in $\mathbb{M}'$. (One is a minimum and the other is maximum.)
3. Since A, B are leaves, they each have unique neighbors A' and B', respectively. Insert edge (A', B') and delete A, B to obtain $\mathcal{C}(\mathbb{M})$.

The following theorems are intuitively obvious, and are proven in the full version [20].

▶ **Theorem 10.** *For any regular contour ϕ, the output of* $\texttt{surgery}(\mathbb{M}, \phi)$ *is* $\mathcal{C}(\mathbb{M})$.

▶ **Theorem 11.** $\texttt{cut}(\mathbb{M}, \phi)$ *consists of two disconnected simplicial complexes.*

4 Raining to partition $\mathbb{M}$

In this section, we describe a linear time procedure that partitions $\mathbb{M}$ into special *extremum dominant* simplicial complexes.

▶ **Definition 12.** A simplicial complex is *minimum dominant* if there exists a minimum x such that every non-minimal *vertex* in the complex has a non-ascending path to x. Analogously define *maximum dominant*.

The first aspect of the partitioning is "raining". Start at some point $x \in \mathbb{M}$ and imagine rain at x. The water will flow downwards along non-ascending paths and "wet" all the points encountered. Note that this procedure considers all points of the complex, not just vertices.

▶ **Definition 13.** Fix $x \in \mathbb{M}$. The set of points $y \in \mathbb{M}$ such that there is a non-ascending path from x to y is denoted by $\texttt{wet}(x, \mathbb{M})$ (which in turn is represented as a simplicial complex). A point z is at the *interface* of $\texttt{wet}(x, \mathbb{M})$ if every neighborhood of z has non-trivial intersection with $\texttt{wet}(x, \mathbb{M})$ (i.e. the intersection is neither empty nor the entire neighborhood).

The following claim gives a description of the interface.

▶ **Claim 14.** *For any x, each component of the interface of* $\texttt{wet}(x, \mathbb{M})$ *contains a join vertex.*

Proof. If $p \in \texttt{wet}(x, \mathbb{M})$, all the points in any contour containing p are also in $\texttt{wet}(x, \mathbb{M})$. (Follow the non-ascending path from x to p and then walk along the contour.) The converse is also true, so $\texttt{wet}(x, \mathbb{M})$ contains entire contours.

Let ε, δ be sufficiently small as usual. Fix some y at the interface. Note that $y \in \texttt{wet}(x, \mathbb{M})$. (Otherwise, $B_\varepsilon(y) \cap \mathbb{M}$ is dry.) The points in $B_\varepsilon(y) \cap \mathbb{M}$ that lie below y have a descending path from y and hence must be wet. There must also be a dry point in $B_\varepsilon(y) \cap \mathbb{M}$ that is above y, and hence, there exists a dry, regular $(f(y) + \delta)$-contour ϕ intersecting $B_\varepsilon(y)$.

Let Γ_y be the contour containing y. Suppose for contradiction that $\forall p \in \Gamma_y$, p has up-degree 1 (see Definition 7). Consider the non-ascending path from x to y and let z be the first point of Γ_y encountered. There exists a wet, regular $(f(y) + \delta)$-contour ψ intersecting

$B_\varepsilon(z)$. Now, walk from z to y along Γ_y. If all points w in this walk have up-degree 1, then ψ is the unique $(f(y)+\delta)$-contour intersecting $B_\varepsilon(w)$. This would imply that $\phi = \psi$, contradicting the fact that ψ is wet and ϕ is dry. ◀

Note that $\texttt{wet}(x, \mathbb{M})$ (and its interface) can be computed in time linear in the size of the wet simplicial complex. We perform a non-ascending search from x. Any face F of $\mathbb{M}$ encountered is partially (if not entirely) in $\texttt{wet}(x, \mathbb{M})$. The wet portion is determined by cutting F along the interface. Since each component of the interface is a contour, this is equivalent to locally cutting F by a hyperplane. All these operations can be performed to output $\texttt{wet}(x, \mathbb{M})$ in time linear in $|\texttt{wet}(x, \mathbb{M})|$.

We define a simple `lift` operation on the interface components. Consider such a component ϕ containing a join vertex y. Take any dry increasing edge incident to y, and pick the point z on this edge at height $f(y)+\delta$ (where δ is an infinitesimal, but larger than the value ε used in the definition of `cut`). Let $\texttt{lift}(\phi)$ be the unique contour through the regular point z. Note that $\texttt{lift}(\phi)$ is dry. The following claim follows directly from Theorem 11.

▶ **Claim 15.** *Let ϕ be a connected component of the interface. Then $\texttt{cut}(\mathbb{M}, \texttt{lift}(\phi))$ results in two disjoint simplicial complexes, one consisting entirely of dry points.*

We describe the main partitioning procedure that cuts a simplicial complex $\mathbb{N}$ into extremum dominant complexes. It takes an additional input of a maximum x. To initialize, we begin with $\mathbb{N}$ set to $\mathbb{M}$ and x as an arbitrary maximum. When we start, rain flows downwards. In each recursive call, the direction of rain is *switched* to the opposite direction. This is crucial to ensure a linear running time. The switching is easily implemented by inverting a complex $\mathbb{N}'$, achieved by negating the height values. We can now let rain flow downwards, as it usually does in our world.

$\texttt{rain}(x, \mathbb{N})$

1. Determine interface of $\texttt{wet}(x, \mathbb{N})$.
2. If the interface is empty, simply output $\mathbb{N}$. Otherwise, denote the connected components by $\phi_1, \phi_2, \ldots, \phi_k$ and set $\phi'_i = \texttt{lift}(\phi_i)$.
3. Initialize $\mathbb{N}_1 = \mathbb{N}$.
4. For i from 1 to k:
   a. Construct $\texttt{cut}(\mathbb{N}_i, \phi'_i)$, consisting of dry complex $\mathbb{L}_i$ and remainder $\mathbb{N}_{i+1}$.
   b. Let the newly created boundary of $\mathbb{L}_i$ be B_i. Invert $\mathbb{L}_i$ so that B_i is a maximum. Recursively call $\texttt{rain}(B_i, \mathbb{L}_i)$.
5. Output $\mathbb{N}_{k+1}$ together with any complexes output by recursive calls.

For convenience, denote the total output of $\texttt{rain}(x, \mathbb{M})$ by $\mathbb{M}_1, \mathbb{M}_2, \ldots, \mathbb{M}_r$.

▶ **Lemma 16.** *Each output $\mathbb{M}_i$ is extremum dominant.*

Proof. Consider a call to $\texttt{rain}(x, \mathbb{N})$. If the interface is empty, then all of $\mathbb{N}$ is in $\texttt{wet}(x, \mathbb{N})$, so $\mathbb{N}$ is trivially extremum dominant. So suppose the interface is non-empty and consists of $\phi_1, \ldots, \phi_k$ (as denoted in the procedure). By repeated applications of Claim 15, $\mathbb{N}_{k+1}$ contains $\texttt{wet}(x, \mathbb{M})$. Consider $\texttt{wet}(x, \mathbb{N}_{k+1})$. The interface is exactly $\phi_1, \ldots, \phi_k$. So the only dry vertices are those in the boundaries $B_1, \ldots, B_k$. But these boundaries are maxima. ◀

As $\texttt{rain}(x, \mathbb{M})$ proceeds, new faces/simplices are created because of repeated cutting. The key to the running time of $\texttt{rain}(x, \mathbb{M})$ is bounding the number of newly created faces, for which we have the following lemma.

▶ **Lemma 17.** *A face $F \in \mathbb{M}$ is cut*[1] *at most once during* `rain`$(x, \mathbb{M})$.

Proof. Notation here follows the pseudocode of `rain`. First, by Theorem 11, all the pieces on which `rain` is invoked are disjoint. Second, all recursive calls are made on dry complexes.

Consider the first time that F is cut, say, during the call to `rain`$(x, \mathbb{N})$. Specifically, say this happens when `cut`$(\mathbb{N}_i, \phi_i')$ is constructed. `cut`$(\mathbb{N}_i, \phi_i')$ will cut F with two horizontal cutting planes, one ε above ϕ_i' and one ε below ϕ_i'. This breaks F into lower and upper portions which are then triangulated (there is also a discarded middle portion). The lower portion, which is adjacent to ϕ_i, gets included in $\mathbb{N}_{k+1}$, the complex containing the wet points, and hence does not participate in any later recursive calls. The upper portion (call it U) is in $\mathbb{L}_i$. Note that the lower boundary of U is in the boundary B_i. Since a recursive call is made to `rain`$(B_i, \mathbb{L}_i)$ (and $\mathbb{L}_i$ is inverted), U becomes wet. Hence U, and correspondingly F, will not be subsequently cut. ◀

The following are direct consequences of Lemma 17 and the `surgery` procedure.

▶ **Theorem 18.** *The total running time of* `rain`$(x, \mathbb{M})$ *is* $O(|\mathbb{M}|)$.

▶ **Claim 19.** *Given* $\mathcal{C}(\mathbb{M}_1), \mathcal{C}(\mathbb{M}_2), \ldots, \mathcal{C}(\mathbb{M}_r)$, $\mathcal{C}(\mathbb{M})$ *can be constructed in* $O(|\mathbb{M}|)$ *time.*

5 Contour trees of extremum dominant complexes

The previous section allows us to restrict attention to extremum dominant complexes. We will orient so that the extremum in question is always a *minimum*. We will fix such a simplicial complex $\mathbb{M}$, with the dominant minimum m^*. For vertex v, we use $\mathbb{M}_v^+$ to denote the simplicial complex obtained by only keeping vertices u such that $f(u) > f(v)$. Analogously, define $\mathbb{M}_v^-$. Note that $\mathbb{M}_v^+$ may contain numerous connected components.

The main theorem of this section asserts that contour trees of minimum dominant complexes have a simple description. Intuitively, the cutting procedure of Section 4 introduced "stumps" where we made cuts, which is why we allowed for non-dominant minima and maxima in the definition of extremum dominant complexes. The punchline is that, ignoring these stumps, the contour tree of an extremum dominant complex is equivalent to its *join* (or *split*) tree, defined in [7]. Hence the critical join tree below is just the join tree without these stumps. Additional definitions and proofs are given in the full version [20].

▶ **Definition 20.** The *critical join tree* $\mathcal{J}_C(\mathbb{M})$ is built on the set V' of all critical points other than the non-dominant minima. The directed edge (u, v) is present when u is the smallest valued vertex in V' in a connected component of $\mathbb{M}_v^+$ and v is adjacent (in $\mathbb{M}$) to a vertex in this component. The *join tree*, $\mathcal{J}(\mathbb{M})$, is defined analogously, but instead on $\mathcal{V}(\mathbb{M})$.

Each non-dominant minimum, m_i, connects to the contour tree at a corresponding split s_i. We have the following (see the full version [20] for more details).

▶ **Theorem 21.** *Let $\mathbb{M}$ have a dominant minimum. The contour tree $\mathcal{C}(\mathbb{M})$ consists of all edges $\{(s_i, m_i)\}$ and $\mathcal{J}_C(\mathbb{M})$.*

▶ **Remark 22.** The above theorem, combined with the previous sections, implies that in order to get an efficient contour tree algorithm, it suffices to have an efficient algorithm for computing $\mathcal{J}_C(\mathbb{M})$. Due to minor technicalities, it is easier to phrase the following section

[1] Technically what we are calling a single cut is done with two hyperplanes.

instead in terms of computing $\mathcal{J}(\mathbb{M})$ efficiently. Note however that for minimum dominant complexes output by `rain`, converting between $\mathcal{J}_C$ and $\mathcal{J}$ is trivial, as $\mathcal{J}$ is just $\mathcal{J}_C$ with each non-dominant minimum m_i augmented along the edge leaving s_i.

6 Painting to compute contour trees

The main algorithmic contribution is a new algorithm for computing join trees of any triangulated simplicial complex $\mathbb{M}$.

Painting: The central tool is a notion of *painting* $\mathbb{M}$. Initially associate a color with each maximum. Imagine there being a large can of paint of a distinct color at each maximum x. We will spill different paint from each maximum and watch it flow down. This is analogous to the raining of Section 4, but paint is a much more viscous liquid. *So paint only flows down edges, and it does not color the interior of higher dimensional faces.* Furthermore, paints do not mix, so every edge of $\mathbb{M}$ gets a unique color. This process (and indeed the entire algorithm) works purely on the 1-skeleton of $\mathbb{M}$, which is just a graph.

▶ **Definition 23.** Let the 1-skeleton of $\mathbb{M}$ have edge set E and maxima X. A *painting* of $\mathbb{M}$ is a map $\chi : X \cup E \to [|X|]$, where $\chi(z)$ is referred to as the *color* of z, with the following property. Consider an edge e. There exists a descending path from some maximum x to e consisting of edges in E, such that all edges along this path have the same color as x.

An *initial* painting also requires that the restriction $\chi : X \to [|X|]$ is a bijection.

▶ **Definition 24.** Fix a painting χ and vertex v.

- An *up-star* of v is the set of edges that all connected to a fixed component of $\mathbb{M}_v^+$.
- A vertex v is *touched by color* c if v is incident to a c-colored edge with v at the lower endpoint. For v, $col(v)$ is the set of colors that touch v.
- A color $c \in col(v)$ *fully touches* v if all edges in an up-star are colored c.
- For any maximum $x \in X$, we say that x is both touched and fully touched by $\chi(x)$.

6.1 The data structures

The binomial heaps $T(c)$: For each color c, $T(c)$ is a subset of vertices touched by c, This is stored as a *binomial max-heap* keyed by vertex heights. Abusing notation, $T(c)$ refers both to the set and the data structure used to store it.

The union-find data structure on colors: We will repeatedly perform unions of classes of colors, and this will be maintained as a standard union-find data structure. For any color c, $rep(c)$ denotes the representative of its class.

Color list $col(v)$: For each point v, we maintain $col(v)$ as a simple list. In addition, we will maintain another (sub)list of colors L such that $\forall c \in L$, v is *not* the highest vertex in $T(rep(c))$. Note that given $c \in col(v)$ and $rep(c)$, this property can be checked in constant time. If c is ever removed from this sublist, it can never enter it again.

For notational convenience, we will not explicitly maintain this sublist. We simply assume that, in constant time, one can determine (if it exists) an arbitrary color $c \in col(v)$ such that v is not the highest vertex in $T(rep(c))$.

The stack K: This consists of non-extremal critical points, with monotonically increasing heights as we go from the base to the head.

Attachment vertex $att(c)$: For each color c, we maintain a critical point $att(c)$ of this color. We will maintain the guarantee that the portion of the join tree above (and including) $att(c)$ has already been constructed.

6.2 The algorithm

We formally describe the algorithm below. We require a technical definition of *ripe* vertices.

▶ **Definition 25.** A vertex v is *ripe* if: for all $c \in col(v)$, v is present in $T(rep(c))$ and is also the highest vertex in this heap.

`init`($\mathbb{M}$)

1. Construct an initial painting of $\mathbb{M}$ using a descending BFS from maxima that does not explore previously colored edges.
2. Determine all critical points in $\mathbb{M}$. For each v, look at $(f(v) \pm \delta)$-contours in $f|_{B_\varepsilon(v)}$ to determine the up and down degrees.
3. Mark each critical v as unprocessed.
4. For each critical v and each up-star, pick an arbitrary color c touching v. Insert v into $T(c)$.
5. Initialize $rep(c) = c$ and set $att(c)$ to be the unique maximum colored c.
6. Initialize K to be an empty stack.

`build`($\mathbb{M}$)

1. Run `init`($\mathbb{M}$).
2. While there are unprocessed critical points:
   a. Run `update`(K). Pop K to get h.
   b. Let $cur(h) = \{rep(c) | c \in col(h)\}$.
   c. For all $c' \in cur(h)$:
      i. Add edge $(att(c'), h)$ to $\mathcal{J}(\mathbb{M})$.
      ii. Delete h from $T(c')$.
   d. Merge heaps $\{T(c') | c' \in cur(h)\}$.
   e. Take union of $cur(h)$ and denote resulting color by $\widehat{c}$.
   f. Set $att(\widehat{c}) = h$ and mark h as processed.

`update`(K)

1. If K is empty, push arbitrary unprocessed critical point v.
2. Let h be the head of K.
3. While h is not ripe:
   a. Find $c \in col(h)$ such that h is not the highest in $T(rep(c))$.
   b. Push the highest of $T(rep(c))$ onto K, and update head h.

A few simple facts:

- At all times, the colors form a valid painting.
- Each vertex is present in at most 2 heaps. After processing, it is removed from all heaps.
- After v is processed, all edges incident to v have the same (representative) color.
- Vertices on the stack are in increasing height order.

▶ **Observation 26.** Each unprocessed vertex is always in exactly one queue of the colors in each of its up-stars. Specifically, for a given up-star of a vertex v, `init`($\mathbb{M}$) puts v into the

queue of exactly one of the colors of the up-star, say c. As time goes on this queue may merge with other queues, but while v remains unprocessed, it is only ever (and always) in the queue of $rep(c)$, since v is never added to a new queue and is not removed until it is processed. In particular, finding the queues of a vertex in $\texttt{update}(K)$ requires at most two union find operations (assuming each vertex records its two colors from $\texttt{init}(\mathbb{M})$).

6.3 Proving correctness

Our main workhorse is the following technical lemma. In the following, the current color of an edge, e, is the value of $rep(\chi(e))$, where $\chi(e)$ is the color of e from the initial painting.

▶ **Lemma 27.** *Suppose vertex v is connected to a component $\mathbb{P}$ of $\mathbb{M}_v^+$ by an edge e which is currently colored c. At all times: either all edges in $\mathbb{P}$ are currently colored c, or there exists a critical vertex $w \in \mathbb{P}$ fully touched by c and touched by another color.*

Proof. Since e has color c, there must exist vertices in $\mathbb{P}$ touched by c. Consider the highest vertex w in $\mathbb{P}$ that is touched by c and some other color. If no such vertex exists, this means all edges incident to a vertex touched by c are colored c. By walking through $\mathbb{P}$, we deduce that all edges are colored c.

So assume w exists. Take the $(f(w)+\delta)$-contour ϕ that intersects $B_\varepsilon(w)$ and intersects some c-colored edge incident to w. Note that all edges intersecting ϕ are also colored c, since w is the highest vertex to be touched by c and some other color. (Take the path of c-colored edges from the maximum to w. For any point on this path, the contour passing through this point must be colored c.) Hence, c fully touches w. But w is touched by another color, and the corresponding edge cannot intersect ϕ. So w must have up-degree 2 and is critical. ◀

▶ **Corollary 28.** *Each time $\texttt{update}(K)$ is called, it terminates with a ripe vertex on top of the stack.*

Proof. $\texttt{update}(K)$ is only called if there are unprocessed vertices remaining, and so by the time we reach step 3 in $\texttt{update}(K)$, the stack has some unprocessed vertex h on it. If h is ripe, then we are done, so suppose otherwise.

Let $\mathbb{P}$ be one of the components of $\mathbb{M}_h^+$. By construction, h was put in the heap of some initial adjacent color c. Therefore, h must be in the current heap of $rep(c)$ (see Observation 26). Now by Lemma 27, either all edges in $\mathbb{P}$ are colored $rep(c)$ or there is some vertex w fully touched by $rep(c)$ and some other color. The former case implies that if there are any unprocessed vertices in $\mathbb{P}$ then they are all in $T(rep(c))$, implying that h is not the highest vertex and a new higher up unprocessed vertex will be put on the stack for the next iteration of the while loop. Otherwise, all the vertices in $\mathbb{P}$ have been processed. However, it cannot be the case that all vertices in all components of $\mathbb{M}_h^+$ have already been processed, since this would imply that h was ripe, and so one can apply the same argument to the other non-fully processed component.

Now consider the latter case, where we have a non-monochromatic vertex w. In this case w cannot have been processed (since after being processed it is touched only by one color), and so it must be in $T(rep(c))$ since it must be in some heap of a color in each up-star (and one up-star is entirely colored $rep(c)$). As w lies above h in $\mathbb{M}$, this implies h is not on the top of this heap. ◀

▶ **Claim 29.** *Consider a ripe vertex v and take the up-star connecting to some component of $\mathbb{M}_v^+$. All edges in this component and the up-star have the same color.*

Proof. Let c be the color of some edge in this up-star. By ripeness, v is the highest in $T(rep(c))$. Denote the component of $\mathbb{M}_v^+$ by $\mathbb{P}$. By Lemma 27, either all edges in $\mathbb{P}$ are colored $rep(c)$ or there exists critical vertex $w \in \mathbb{P}$ fully touched by $rep(c)$ and another color. In the latter case, w has not been processed, so $w \in T(rep(c))$ (contradiction to ripeness). Therefore, all edges in $\mathbb{P}$ are colored $rep(c)$. ◀

▶ **Claim 30.** *The partial output on the processed vertices is exactly the restriction of $\mathcal{J}(\mathbb{M})$ to these vertices.*

Proof. More generally, we prove the following: all outputs on processed vertices are edges of $\mathcal{J}(\mathbb{M})$ and for any current color c, $att(c)$ is the lowest processed vertex of that color. We prove this by induction on the processing order. The base case is trivially true, as initially the processed vertices and attachments of the color classes are the set of maxima. For the induction step, consider the situation when v is being processed.

Since v is being processed, we know by Corollary 28 that it is ripe. Take any up-star of v, and the corresponding component $\mathbb{P}$ of $\mathbb{M}_v^+$ that it connects to. By Claim 29, all edges in $\mathbb{P}$ and the up-star have the same color (say c). If some critical vertex in $\mathbb{P}$ is not processed, it must be in $T(c)$, which violates the ripeness of v. Thus, all critical vertices in $\mathbb{P}$ have been processed, and so by the induction hypothesis, the restriction of $\mathcal{J}(\mathbb{M})$ to $\mathbb{P}$ has been correctly computed. Additionally, since all critical vertices in $\mathbb{P}$ have processed, they all have the same color c of the lowest critical vertex in $\mathbb{P}$. Thus by the strengthened induction hypothesis, this lowest critical vertex is $att(c)$.

If there is another component of $\mathbb{M}_v^+$, the same argument implies the lowest critical vertex in this component is $att(c')$ (where c' is the color of edges in the respective component). Now by the definition of $\mathcal{J}(\mathbb{M})$, the critical vertex v connects to the lowest critical vertex in each component of $\mathbb{M}_v^+$, and so by the above v should connect to $att(c)$ and $att(c')$, which is precisely what v is connected to by `build`$(\mathbb{M})$. Moreover, `build` merges the colors c and c' and correctly sets v to be the attachment, as v is the lowest processed vertex of this merged color (as by induction $att(c)$ and $att(c')$ were the lowest vertices before merging colors). ◀

▶ **Theorem 31.** *Given an input complex $\mathbb{M}$,* `build`$(\mathbb{M})$ *terminates and outputs $\mathcal{J}(\mathbb{M})$.*

Proof. First observe that each vertex can be processed at most once by `build`$(\mathbb{M})$. By Corollary 28, we know that as long as there is an unprocessed vertex, `update`(K) will be called and will terminate with a ripe vertex which is ready to be processed. Therefore, eventually all vertices will be processed, and so by Claim 30 the algorithm will terminate having computed $\mathcal{J}(\mathbb{M})$. ◀

6.4 Upper Bounds for Running Time

The algorithm `build`$(\mathbb{M})$ processes vertices in $\mathcal{J}(\mathbb{M})$ one at a time. The main processing cost comes from priority queue operations. The cost of these operations is a function of the size of the queue which is in turn a function of the choices made by the subroutine `init`$(\mathbb{M})$.

▶ **Definition 32.** A *leaf assignment* χ of a binary tree T assigns *two* distinct leaves to each internal vertex v, one from the left and one from the right subtree of v (or only one leaf if v has only one child). The subroutine `init`$(\mathbb{M})$ naturally defines a leaf assignment to $\mathcal{J}(\mathbb{M})$ (which is a rooted binary tree) according to the priority queue for each up-star we put a given vertex in. Call this the *initial coloring* of the vertices in $\mathcal{J}(\mathbb{M})$, and denote it by χ.

For a vertex v in $\mathcal{J}(\mathbb{M})$, let $L(v)$ denote the set of leaves of the subtree rooted at v, and let $A(v)$ denote the set of ancestors of v, i.e. the vertices on the v to root path. For a

vertex $v \in \mathcal{J}(\mathbb{M})$, and an initial coloring χ, we use H_v to denote the *heap* at v. Formally, $H_v = \{u | u \in A(v), \chi(u) \cap L(v) \neq \emptyset\}$, i.e. the set of ancestors colored by some leaf in $L(v)$.

Given the above technical definition, the proof of the following lemma is straightforward, though long and so has been moved to the full version [20].

▶ **Lemma 33.** *Let $\mathbb{M}$ be a simplicial complex with t critical points. The running time of* `build`$(\mathbb{M})$ *is $O(N + t\alpha(t) + \sum_{v \in \mathcal{J}(\mathbb{M})} \log |H_v|)$, where H_v is defined by an initial coloring.*

Our main result, Theorem 1, is an easy corollary of the above lemma.

Proof of Theorem 1. Consider a critical point v of the initial input complex. By Theorem 11 this vertex appears in exactly one of the pieces output by `rain`. As in the Theorem 1 statement, let ℓ_v denote the length of the longest directed path passing through v in the contour tree of the input complex, and let ℓ'_v denote the longest directed path passing through v in the join tree of the piece containing v. By Theorem 10, ignoring non-dominant extrema introduced from cutting (whose cost can be charged to a corresponding saddle), the join tree on each piece output by `rain` is isomorphic to some connected subgraph of the contour tree of the input complex, and hence $\ell'_v \leq \ell_v$. Moreover, $|H_v|$ only counts vertices in a v to root path and so trivially $|H_v| \leq \ell'_v$, implying Theorem 1. ◀

Note that there is fair amount of slack in this argument as $|H_v|$ may be significantly smaller than ℓ'_v. This slack allows for the more refined upper and lower bounds mentioned in Section 1.1. Quantifying this slack however is quite challenging, and requires a significantly more sophisticated analysis involving path decompositions, detailed in the full version [20].

Acknowledgements. We thank Hsien-Chih Chang, Jeff Erickson, and Yusu Wang for numerous useful discussions. This work is supported by the Laboratory Directed Research and Development (LDRD) program of Sandia National Laboratories. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1 C. Bajaj, M. van Kreveld, R. W. van Oostrum, V. Pascucci, and D. R. Schikore. Contour trees and small seed sets for isosurface traversal. Technical Report UU-CS-1998-25, Department of Information and Computing Sciences, Utrecht University, 1998.

2 K. Beketayev, G. Weber, M. Haranczyk, P.-T. Bremer, M. Hlawitschka, and B. Hamann. Visualization of topology of transformation pathways in complex chemical systems. In *Computer Graphics Forum (EuroVis 2011)*, pages 663–672, 2011.

3 R. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proceedings of Fall Joint Computer Conference*, pages 445–458, 1963.

4 P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

5 P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1307–1325, 2011.

6 H. Carr. *Topological Manipulation of Isosurfaces.* PhD thesis, University of British Columbia, 2004.

7 H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24(2):75–94, 2003.

8 Y. Chiang, T. Lenz, X. Lu, and G. Rote. Simple and optimal output-sensitive construction of contour trees using monotone paths. *Computational Geometry: Theory and Applications*, 30(2):165–195, 2005. doi:10.1016/j.comgeo.2004.05.002.

9 K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Loops in reeb graphs of 2-manifolds. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 344–350, 2003. doi:10.1145/777792.777844.

10 H. Doraiswamy and V. Natarajan. Efficient algorithms for computing reeb graphs. *Computational Geometry: Theory and Applications*, 42:606–616, 2009.

11 H. Doraiswamy and V. Natarajan. Computing reeb graphs as a union of contour trees. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):249–262, 2013.

12 H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.

13 H. Freeman and S. Morse. On searching a contour map for a given terrain elevation profile. *Journal of the Franklin Institute*, 284(1):1–25, 1967.

14 W. Harvey, Y. Wang, and R. Wenger. A randomized o(m log m) time algorithm for computing reeb graph of arbitrary simplicial complexes. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 267–276, 2010.

15 D. Laney, P.-T. Bremer, A. Macarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1053–1060, 2006.

16 A. Mascarenhas, R. Grout, P.-T. Bremer, V. Pascucci, E. Hawkes, and J. Chen. Topological feature extraction for comparison of length scales in terascale combustion simulation data. In *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, pages 229–240, 2011.

17 S. Parsa. A deterministic o(m log m) time algorithm for the reeb graph. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 269–276, 2012.

18 V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level set. In *IEEE Visualization*, pages 187–194, 2002. doi:10.1109/VISUAL.2002.1183774.

19 V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. *ACM Transactions on Graphics*, 26(58), 2007.

20 B. Raichel and C. Seshadhri. Avoiding the global sort: A faster contour tree algorithm. *CoRR*, abs/1411.2689, 2014.

21 Y. Shinagawa and T. Kunii. Constructing a reeb graph automatically from cross sections. *IEEE Comput. Graphics Appl.*, 11(6):44–51, 1991.

22 S. Tarasov and M. Vyalyi. Construction of contour trees in 3d in $O(n \log n)$ steps. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 68–75, 1998. doi:10.1145/276884.276892.

23 J. Tierny, A. Gyulassy, E. Simon, and V. Pascucci. Loop surgery for volumetric meshes: Reeb graphs reduced to contour trees. *IEEE Trans. on Visualization and Computer Graphics*, 15(6):1177–1184, 2009.

24 M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the Symposium on Computational Geometry (SoCG)*, pages 212–220, 1997. doi:10.1145/262839.269238.

Configurations of Lines in 3-Space and Rigidity of Planar Structures*

Orit E. Raz

Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel
oritraz@post.tau.ac.il

Abstract

Let L be a sequence $(\ell_1, \ell_2, \ldots, \ell_n)$ of n lines in $\mathbb{C}^3$. We define the *intersection graph* $G_L = ([n], E)$ of L, where $[n] := \{1, \ldots, n\}$, and with $\{i, j\} \in E$ if and only if $i \neq j$ and the corresponding lines ℓ_i and ℓ_j intersect, or are parallel (or coincide). For a graph $G = ([n], E)$, we say that a sequence L is a *realization* of G if $G \subset G_L$. One of the main results of this paper is to provide a combinatorial characterization of graphs $G = ([n], E)$ that have the following property: For every *generic* (see Definition 4.1) realization L of G, that consists of n pairwise distinct lines, we have $G_L = K_n$, in which case the lines of L are either all concurrent or all coplanar.

The general statements that we obtain about lines, apart from their independent interest, turns out to be closely related to the notion of graph rigidity. The connection is established due to the so-called Elekes–Sharir framework, which allows us to transform the problem into an incidence problem involving lines in three dimensions. By exploiting the geometry of contacts between lines in 3D, we can obtain alternative, simpler, and more precise characterizations of the rigidity of graphs.

1998 ACM Subject Classification G.2. Discrete Mathematics

Keywords and phrases Line configurations, Rigidity, Global Rigidity, Laman graphs

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.58

1 Introduction

Let L be a sequence $(\ell_1, \ell_2, \ldots, \ell_n)$ of n lines in $\mathbb{C}^3$. We define the *intersection graph* $G_L = ([n], E)$ of L, where $[n] := \{1, \ldots, n\}$, and with $\{i, j\} \in E$ if and only if $i \neq j$ and the corresponding lines ℓ_i and ℓ_j intersect, or are parallel (or coincide). For a graph $G = ([n], E)$, we say that a sequence L is a *realization* of G if[1] $G \subset G_L$.

We consider the following general question: What can be said about realizations L of a certain graph G? In particular, we are interested in conditions on graphs G that guarantee that, for every realization L of G, the lines of L must be either all concurrent or all coplanar. In other words, we want conditions on G that guarantee that, for every realization L of G, we have $G_L = K_n$, the complete graph on n vertices. (It can be easily verified that $G_L = K_n$ implies that the lines of L must be either all concurrent or all coplanar.) Unfortunately, already by removing one edge of K_n, this property seems to fail: One can easily find configurations L with lines that are neither all concurrent nor all coplanar, and such that $G_L = K_n \setminus \{1, 2\}$, say. Indeed, consider $n - 2$ lines $\ell_3, \ldots, \ell_n$ that are all concurrent *and* all coplanar, let ℓ_1 be any line that lies on the common plane supporting those lines (but does

* Work on this paper was supported by Grant 892/13 from the Israel Science Foundation, by the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11), and by a Shulamit Aloni Fellowship from the Israeli Ministry of Science.

[1] For graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we say that $G_1 \subset G_2$ if $V_1 = V_2$ and $E_1 \subseteq E_2$.

© Orit Esther Raz;
licensed under Creative Commons License CC-BY

not go through their common intersection point), and let ℓ_2 be any line that goes through the common intersection point of $\ell_3, \ldots, \ell_n$ (but does not lie on the plane supporting those lines). Then for $L = (\ell_1, \ell_2, \ldots, \ell_n)$, we get $G_L = K_n \setminus \{1, 2\}$, and the lines of L are neither all concurrent nor all coplanar. Note however that in this example we had to use $n - 2$ lines which are both concurrent and coplanar, which we would like to think of as a degenerate configuration of lines.

In this paper we characterize graphs $G = ([n], E)$ with the property that, for every *generic* (see Definition 4.1) realization L of G, that consists of n pairwise distinct lines, we have $G_L = K_n$ (that is, the lines of L are either all concurrent or all coplanar).

In the background of our results lies a connection (that we establish here) between line configurations and the classical notion of rigidity of planar realizations of graphs (see Section 5 for the definitions). The connection is established (in Section 6) due to the so-called Elekes–Sharir framework, which allows us to transform the problem into an incidence problem involving lines in three dimensions. By exploiting the geometry of contacts between lines in 3D, we can obtain alternative, simpler, and more precise characterizations of the rigidity of graphs.

2 Intersection graph and the variety X_G

For simplicity, in this paper all the lines are assumed to be non-horizontal. Let L be a sequence $(\ell_1, \ell_2, \ldots, \ell_n)$ of n (not necessarily distinct) complex lines in $\mathbb{C}^3$. A line ℓ can be parametrized as

$$\ell(t) = (a, b, 0) + t(c, d, 1), \quad t \in \mathbb{C},$$

for certain unique $a, b, c, d \in \mathbb{C}$, and we may represent ℓ as a point $u = (a, b, c, d) \in \mathbb{C}^4$, in this sense. By identifying $\mathbb{C}^{4n}$ with $(\mathbb{C}^4)^n$, we may regard a sequence $L = (\ell_1, \ell_2, \ldots, \ell_n)$ of n lines in $\mathbb{C}^3$ as a point $(u_1, \ldots, u_n) \in \mathbb{C}^{4n}$, where each $u_i = (a_i, b_i, c_i, d_i) \in \mathbb{C}^4$ is the point representing the line ℓ_i of L. Similarly, every point $x \in \mathbb{C}^{4n}$ can be interpreted as a sequence $L = L(x)$ of n (not necessarily distinct) lines in $\mathbb{C}^3$.

Given a graph $G = ([n], E)$, we define the variety $X_G := \mathrm{Cl}(\hat{X}_G)$, where $\mathrm{Cl}(S)$ is the *Zariski closure* of a set $S \subset \mathbb{C}^{4n}$, and

$$\hat{X}_G := \left\{ x \in \mathbb{C}^{4n} \mid G \subset G_{L(x)} \text{ and the lines of } L(x) \text{ are pairwise distinct} \right\}.$$

Note that for every point $x \in X_G$, we have $G \subset G_{L(x)}$ (where here it is possible for some of the lines of $L(x)$ to coincide). Indeed, we have $\hat{X}_G \subset Y_G$, where $Y_G := \left\{ x \in \mathbb{C}^{4n} \mid G \subset G_{L(x)} \right\}$. Since Y_G is an algebraic variety, it follows that $X_G \subset Y_G$. To see that Y_G is a variety, let g be the polynomial in the eight coordinates a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, which vanishes if and only if the pair of lines associated with the coordinates (a_1, b_1, c_1, d_1) and (a_2, b_2, c_2, d_2) intersect or are parallel. Namely,

$$g(a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2) = (a_1 - a_2)(d_1 - d_2) - (b_1 - b_2)(c_1 - c_2). \tag{1}$$

By definition, Y_G is given by the system

$$g(a_i, b_i, c_i, d_i, a_j, b_j, c_j, d_j) = 0 \quad \forall\, (i, j) \in E,$$

and is therefore a variety, as claimed.

We have the following simple observation.

▶ **Lemma 2.1.** *The variety X_{K_n} is $(2n + 3)$-dimensional.*

Proof. First note that a point in $\hat{X}_{K_n}$ corresponds to a sequence L of n mutually intersecting lines, and hence its lines are either all concurrent (or mutually parallel), or all coplanar.

A line, given by the parameterization

$$\ell(t) = (a, b, 0) + t(c, d, 1), \quad t \in \mathbb{C}$$

passes through a point $(x, y, z) \in \mathbb{C}^3$ if and only if $a = x - cz$ and $b = y - dz$. Thus, sequences L of n concurrent lines in $\mathbb{C}^3$ form a Zariski-dense subset of the following variety

$$\{(x - c_1 z, y - d_1 z, c_1, d_1, \ldots, x - c_n z, y - d_n z, c_n, d_n) \mid x, y, z, c_1, d_1, \ldots, c_n, d_n \in \mathbb{C}\},$$

which is $(2n + 3)$-dimensional, as it is the image of $\mathbb{C}^{2n+3}$ under a polynomial function.

Sequences L of n mutually parallel lines in $\mathbb{C}^3$ span the $(2n + 2)$-dimensional subvariety of X_{K_n} given by

$$\{(a_1, b_1, c_0, d_0, \ldots, a_n, b_n, c_0, d_0) \in \mathbb{C}^{4n} \mid c_0, d_0, a_1, b_1, \ldots, a_n, b_n \in \mathbb{C}\}.$$

Similarly, a line $\ell(t) = (a, b, 0) + t(c, d, 1)$ lies in a plane given by $z = \lambda x + \mu y + \nu$, if and only if $1 = \lambda c + \mu d$ and $-\nu = \lambda a + \mu b$. Thus, sequences L of n coplanar lines form a Zariski-dense subset of the following variety

$$\{(a_1, b_1, c_1, d_1, \ldots, a_n, b_n, c_n, d_n) \mid 1 = \lambda c_i + \mu d_i, -\nu = \lambda a_i + \nu b_i, \lambda, \mu, \nu \in \mathbb{C}, i \in [n]\}.$$

which is again $(2n + 3)$-dimensional. So X_{K_n} is the union of these three varieties, and hence it is $(2n + 3)$-dimensional. ◀

The following simple lemma asserts that X_G is always at least $(2n + 3)$-dimensional, where n is the number of vertices of G. Later on (in Corollary 3.3), we introduce a sufficient and necessary condition on G for X_G to be exactly $(2n + 3)$-dimensional.

▶ **Lemma 2.2.** *Let G be a graph on the vertex set $[n]$. Then X_G is of dimension at least $2n + 3$.*

Proof. Since G (and every graph on $[n]$) is contained in K_n, we have, by definition, that $\hat{X}_{K_n} \subset \hat{X}_G$ and hence also $X_{K_n} \subset X_G$. By Lemma 2.1, X_{K_n} is $(2n + 3)$-dimensional, and thus X_G is of dimension at least $2n + 3$, as asserted. ◀

3 Laman graphs

We recall the definition of *Laman graphs*. These graphs play a fundamental role in the theory of *combinatorial rigidity* of graphs in the plane, and were discovered by Laman [10]. In our definition we refer only to the combinatorial properties of those graphs. Later, in Section 6, the "reason" why these graphs pop up also in our context will become more apparent.

▶ **Definition 3.1.** A graph $G = (V, E)$ is called a *Laman graph* if

(i) $|E| = 2|V| - 3$, and

(ii) every subgraph $G' = (V', E')$ of G, with $|V'| \geq 2$, satisfies $|E'| \leq 2|V'| - 3$.

The main result of this section is the following theorem.

▶ **Theorem 3.2.** *If G is a Laman graph on n vertices, then X_G is $(2n + 3)$-dimensional.*

It follows from Theorem 3.2 that, for every graph G on n vertices that contains a subgraph $G' \subset G$ which is Laman, we have $\dim X_G = 2n+3$. Indeed, we have $G' \subset G \subset K_n$, which implies that $X_{K_n} \subset X_G \subset X_{G'}$, and, in particular, $\dim X_{K_n} \leq \dim X_G \leq \dim X_{G'}$. Combining Theorem 3.2 and Lemma 2.1, the claim follows. In Section 7 we show (in Theorem 7.1) that it is also necessary for G to contain a subgraph which is Laman, in order for X_G to be $(2n+3)$-dimensional. We thus obtain the following characterization.

▶ **Corollary 3.3.** *Let G be a graph on n vertices. Then X_G is $(2n+3)$-dimensional if and only if there exists a subgraph $G' \subset G$ which is Laman.*

For the proof of Theorem 3.2 we use the following constructive characterization of Laman graphs (called the Henneberg or Henneberg–Laman construction; see Jackson and Jordán [8, Corollary 2.12] and references therein). In the statement, a 0-*extension* means adding a new vertex of degree 2 to G, and a 1-*extension* means subdividing an edge uv of G by a new vertex z, and adding a new edge zw for some $w \neq u, v$. That is, the edge set of the new graph is $(E \setminus \{uv\}) \cup \{uz, vz, wz\}$.

▶ **Theorem 3.4** (Henneberg–Laman)**.** *A graph $G = (V, E)$ is Laman if and only if G can be obtained from K_2 by a sequence of* 0- *and* 1-*extensions.*

We observe the following simple property.

▶ **Lemma 3.5.** *Let ℓ_1, ℓ_2, ℓ_3 be three distinct lines in $\mathbb{C}^3$. Consider the variety W of lines $\ell \subset \mathbb{C}^3$ that intersect each of ℓ_i, for $i = 1, 2, 3$ (similar to above, W can be interpreted as a variety in $\mathbb{C}^4$). If ℓ_1, ℓ_2, ℓ_3 are either concurrent or coplanar, W is two-dimensional. Otherwise, W is one-dimensional.*

Proof. If ℓ_1, ℓ_2, ℓ_3 are both concurrent and coplanar, then each $\ell \in W$ must either pass through their common point or lie on their common plane. If ℓ_1, ℓ_2, ℓ_3 are concurrent and not coplanar, then each $\ell \in W$ must pass through their common point. If the they are coplanar but not concurrent, each $\ell \in W$ must lie on their common plane. In each of these scenarios, ℓ has two degrees of freedom.

If the three lines are pairwise skew, $\ell \in W$ must be a generator line of the regulus that they span, and then has only one degree of freedom.

It is left to handle the case where ℓ_1, ℓ_2, ℓ_3 are neither concurrent nor coplanar, and a pair of them, say, ℓ_1, ℓ_2, is concurrent. Then ℓ_3 intersects their common plane in a single point q, which is not $\ell_1 \cap \ell_2$. Then each $\ell \in W$ must either pass through $\ell_1 \cap \ell_2$ and an arbitrary point of ℓ_3, or pass through q and lie on the common plan spanned by ℓ_1, ℓ_2. In either case ℓ has one degree of freedom. ◀

3.1 Proof of Theorem 3.2

By Lemma 2.2, X_G is at least $(2n+3)$-dimensional. So it suffices to show that, in case G is Laman, X_G is at most $(2n+3)$-dimensional.

We use induction on the number $n \geq 2$ of vertices of G. For the base case $n = 2$, the only graph to consider is $G = K_2$, for which X_G is 7-dimensional, by Lemma 2.1. Assume the correctness of the statement for each $2 \leq n' < n$ and let G be a Laman graph on the vertex set $[n]$. By Theorem 3.4, G can be obtained from K_2 by a sequence of $N = n - 2$ extensions (note that each extension adds one new vertex to the graph). Fix such a sequence, and let G' be the graph obtained after applying the first $N - 1$ extensions in the sequence. So G can be obtained from G' by applying a 0- or 1-extension to G', and G' is Laman. Up to renaming

the vertices, we may assume that the vertex set of G' is $[n-1]$. By the induction hypothesis $\dim X_{G'} = 2(n-1)+3 = 2n+1$.

Suppose first that G is obtained from G' by a 0-extension, that is, by adding a new vertex, n, to G' and two edges connecting n to some pair of vertices, say, $1, 2$, of G'.

Let $\pi : \mathbb{C}^{4n} \to \mathbb{C}^{4n-4}$ denote that projection of $\mathbb{C}^{4n}$ onto its first $4n-4$ coordinates. We claim that $\pi(\hat{X}_G) = \hat{X}_{G'}$. Indeed, for every sequence of lines $L = (\ell_1, \ldots, \ell_{n-1}, \ell_n)$ with $G \subset G_L$, removing the line ℓ_n results in a sequence $L' := (\ell_1, \ldots, \ell_{n-1})$ with $G' \subset G_{L'}$; so $\pi(\hat{X}_G) \subseteq \hat{X}_{G'}$. Conversely, for every sequence $L' = (\ell_1, \ldots, \ell_{n-1})$ with $G' \subset G_{L'}$, there exists a line ℓ_n (in fact, a two-dimensional family of lines — see below) that intersects both ℓ_1, ℓ_2, and thus, for $L := (\ell_1, \ldots, \ell_{n-1}, \ell_n)$, we have $G \subset G_L$; so also $\hat{X}_{G'} \subseteq \pi(\hat{X}_G)$. This implies $\pi(\hat{X}_G) = \hat{X}_{G'}$, as claimed.

An easy property in algebraic geometry, given in Lemma 1.3 in the Appendix, implies that

$$X_{G'} = \mathrm{Cl}(\hat{X}_{G'}) = \mathrm{Cl}(\pi(\hat{X}_G)) = \mathrm{Cl}(\pi(X_G)).$$

Note also that for every $y \in \hat{X}_{G'}$, we have that $\pi^{-1}(y) \cap X_G$ is two-dimensional. Indeed, as was already noted, writing $L(y) = (\ell_1, \ldots, \ell_{n-1})$, we have $x \in \pi^{-1}(y) \cap X_G$ for every $L(x)$ of the form $L(x) = (\ell_1, \ldots, \ell_{n-1}, \ell_n)$, where the first $n-1$ lines are fixed and the line ℓ_n is any line that intersects both ℓ_1, ℓ_2. The set of such lines is two-dimensional. Indeed, each such line can be parameterized by the pair of points of its intersection with ℓ_1 and ℓ_2, except for lines that pass through the intersection point $\ell_1 \cap \ell_2$, if such a point exists. However, the space of lines of this latter kind is also two-dimensional, and the claim follows.

Let X be any irreducible component of $\hat{X}_G$. Since $X_G = \mathrm{Cl}(\hat{X}_G)$, we must have $X \cap \hat{X}_G \neq \emptyset$, for every such component X (otherwise, the union of the irreducible components of X_G, excluding X, already contains $\hat{X}_G$). Let $y \in \pi(X \cap \hat{X}_G) \subset \hat{X}_{G'}$. By another basic property in algebraic geometry, given in Lemma 1.2 in the Appendix, we have, for each $y \in \hat{X}_{G'}$,

$$\dim X \leq \dim \pi(X) + \dim(\pi^{-1}(y) \cap X) \leq \dim X_{G'} + \dim(\pi^{-1}(y) \cap X_G) \leq 2n+3.$$

Since this holds for every irreducible component X of X_G, we get $\dim X_G \leq 2n+3$, which completes the proof of the theorem for this case.

Assume next that G is obtained from G' by applying a 1-extension, that is, by subdividing an edge, say $\{i, j\}$ of G' by a new vertex n, and adding a new edge $\{n, k\}$ for some $k \neq i, j$. That is, G is obtained from G' by replacing the edge $\{i, j\}$ by the three edges $\{i, n\}$, $\{j, n\}$, and $\{k, n\}$. Without loss of generality assume $i = 1$, $j = 2$, $k = 3$. Consider the graph G'_{12} resulting by removing the edge $\{1, 2\}$ from G', and the corresponding variety $X_{G'_{12}}$.

Arguing similar to above, we have $X_{G'_{12}} = \mathrm{Cl}(\pi(X_G))$. Indeed, for every sequence of lines $L = (\ell_1, \ldots, \ell_{n-1}, \ell_n)$ with $G \subset G_L$, removing the line ℓ_n results in a sequence $L' := (\ell_1, \ldots, \ell_{n-1})$ with $G'_{12} \subset G_{L'}$; so $\pi(\hat{X}_G) \subseteq \hat{X}_{G'}$. Conversely, for every sequence $L' = (\ell_1, \ldots, \ell_{n-1})$ with $G'_{12} \subset G_{L'}$, there exists a line ℓ_n (in fact, a family of lines which is at least one-dimensional, by Lemma 3.5) that intersects each of ℓ_1, ℓ_2, ℓ_3, and thus, for $L := (\ell_1, \ldots, \ell_{n-1}, \ell_n)$, we have $G \subset G_L$; so also $\hat{X}_{G'} \subseteq \pi(\hat{X}_G)$. This implies $\pi(\hat{X}_G) = \hat{X}_{G'}$. Applying Lemma 1.3 in the Appendix, we get $X_{G'_{12}} = \mathrm{Cl}(\pi(X_G))$, as claimed.

Let g be the polynomial in (1) in the first eight coordinates of $\mathbb{C}^{4n-4}$, which vanishes if and only if the pair of lines ℓ_1, ℓ_2 associated with those 8 coordinates intersect. Let $Z(g)$ denote the zero-set of g in $\mathbb{C}^{4n-4}$. Clearly, $X_{G'} = X_{G'_{12}} \cap Z(g) \subset X_{G'_{12}}$.

Assume, without loss of generality, that $X_{G'_{12}}$ is irreducible (otherwise apply the same argument to each of its irreducible components). By Lemma 1.1 in the Appendix, we have

either $X_{G'} = X_{G'_{12}}$ and thus $\dim X_{G'_{12}} = \dim X_{G'} \leq 2n+1$, or, otherwise, $\dim X_{G'} = \dim X_{G'_{12}} - 1$.

Suppose that the former case occurs, that is, $\dim X_{G'_{12}} \leq 2n+1$. It follows from Lemma 3.5 that $\dim(\pi^{-1}(y) \cap X_G) \leq 2$, for every $y \in \hat{X}_{G'_{12}}$. Lemma 1.2 then implies that

$$\dim X_G \leq \dim X_{G'_{12}} + \dim(\pi^{-1}(y) \cap X_G) \leq 2n+1+2 = 2n+3,$$

as needed.

Suppose next that the latter case occurs, that is,

$$\dim X_{G'} = \dim(X_{G'_{12}} \cap Z(g)) = \dim X_{G'_{12}} - 1$$

(and, in particular, $\dim X_{G'_{12}} \leq 2n+2$).

Choose a point $y_0 \in \hat{X}_{G'_{12}} \setminus Z(g)$, and write $L(y_0) := (\ell_1, \ldots, \ell_{n-1})$. Since $y_0 \notin Z(g)$, we have in particular that the lines ℓ_1, ℓ_2, ℓ_3 of $L(y_0)$ are neither all concurrent nor all coplanar. By Lemma 3.5, we have $\dim(\pi^{-1}(y_0) \cap X_G) = 1$. Applying Lemma 1.2 once again, we get

$$\dim X_G \leq \dim X_{G'_{12}} + \dim(\pi^{-1}(y_0) \cap X_G) \leq 2n+2+1 = 2n+3.$$

This completes the proof. ◀

4 Hendrickson graphs

In this section we introduce a characterization for graphs G, such that $G \subset G_L$ guarantees that the lines of L are either all concurrent or all coplanar. As already discussed in the introduction, for this we need to restrict ourselves only to *generic* configurations L, defined as follows.

▶ **Definition 4.1.** Let G be a graph and put $k := \dim X_G$. A point $x \in X_G$ is called *generic* if it is a regular point of a k-dimensional irreducible component of X_G.

The following theorem is the main result of this section, preceded a definition needed for its statement.

▶ **Definition 4.2.** A graph $G = (V, E)$ is called *redundant* if, for every edge $e \in E$, there exists a subgraph $G' \subset G \setminus \{e\}$ which is Laman. A graph G is called a *Hendrickson graph* if it is redundant and 3-(vertex-)connected.[2]

▶ **Theorem 4.3.** *Let G be a Hendrickson graph on n vertices, and let X be any $(2n+3)$-dimensional irreducible component of X_G. Then, for every $x \in X$, the lines of $L(x)$ are either all concurrent or all coplanar.*

For the proof of Theorem 4.3 we use the following constructive characterization of Hendrickson graphs obtained by Jackson and Jordán [8].

▶ **Theorem 4.4** (Jackson and Jordán [8])**.** *Every Hendrickson graph can be built up from K_4 by a sequence of edge additions and 1-extensions.*

▶ Remark. Note that applying a 1-extension or an edge addition preserves the property of being Hendrickson.

[2] Recall that a graph G is 3-(vertex-)connected if a removal of any pair of vertices of G results in a connected graph.

4.1 Proof of Theorem 4.3

Let G be a Hendrickson graph on the vertex set $[n]$. By Theorem 4.4, there exists a sequence $G_0, \ldots, G_N$ of Hendrckson (i.e., 3-connected and redundant) graphs, such that $G_0 = K_4$, $G_N = G$, and G_i is obtained from G_{i-1} by an edge addition or a 1-extension, for each $i = 1, \ldots, N$.

We use induction on N. The base case $N = 0$ and $G_0 = K_4$ is trivial, because a simple case analysis shows that every four distinct lines in $\mathbb{C}^3$ which pairwise intersect are either all concurrent or all coplanar. Assume the correctness of the statement for $0 \leq N' < N$, and assume that G is obtained from $G' := G_{N-1}$ by either adding a new edge or by applying a 1-extension to G'; by the remark following Theorem 4.4, G' is 3-connected and redundant.

Consider first the case where G is obtained from G' by adding a new edge, say $\{1, 2\}$. By assumption, each of G, G' contains a subgraph which is Laman, and thus each of X_G and $X_{G'}$ is $(2n + 3)$-dimensional, by Theorem 3.2. Clearly, $G' \subset G$ and so $X_G \subset X_{G'}$. In particular, every $(2n+3)$-dimensional irreducible component X of X_G is also a component of $X_{G'}$. By the induction hypothesis, for every regular point $x \in X$, the sequence $L(x)$ consists of either n concurrent or n coplanar lines. This completes the proof for this case.

Consider next the case where G is obtained from G' by a 1-extension. Up to renaming the vertices, we may assume that G' is a graph on the vertex set $[n-1]$ and G is obtained from G' by adding a new vertex n, and by replacing an edge $\{1, 2\}$ by three edges $\{1, n\}$, $\{2, n\}$, $\{3, n\}$. Let G'_{12} be the graph obtained from G' by removing its edge $\{1, 2\}$. Let $\pi : \mathbb{C}^{4n} \to \mathbb{C}^{4n-4}$ stand for the projection of $\mathbb{C}^{4n}$ onto its first $4n - 4$ coordinates. As argued in the proof of Theorem 3.2, we have $X_{G'_{12}} = \mathrm{Cl}(\pi(X_G))$.

Our assumption that G' is redundant implies that G'_{12} contains a subgraph which is Laman. Hence, each of $X_{G'_{12}}$, $X_{G'}$ is of dimension $2(n - 1) + 3 = 2n + 1$, by Theorem 3.2. Note that G is obtained from G'_{12} by adding a new vertex, n, of degree 3.

Let X be a $(2n + 3)$-dimensional irreducible component of X_G and let x be a regular point of $X \cap \hat{X}_G$ (note that this intersection is nonempty since $X \subset \mathrm{Cl}(\hat{X}_G)$). Let X' be some irreducible component of $\mathrm{Cl}(\pi(X))$ that contains $\pi(x)$ (if $\pi(x)$ lies on more than one such irreducible component, choose X' to be one with maximal dimension).

We claim that X' is $(2n + 1)$-dimensional. Indeed, for contradiction, assume that X' is of dimension at most $2n$ (X' cannot have higher dimension, because $X_{G'_{12}}$ is $(2n + 1)$-dimensional). Note that, for every $y \in X'$, $\pi^{-1}(y) \cap X$ is at most two-dimensional, by Lemma 3.5. Combined with Lemma 1.2, we get

$$\dim X \leq \dim X' + \dim(\pi^{-1}(y) \cap X) \leq 2n + 2,$$

which contradicts our assumption that X is $(2n + 3)$-dimensional.

We next claim that X' is also an irreducible component of $X_{G'}$ (and not only of $X_{G'_{12}}$); that is, we claim that $X' \subset X_{G'}$. Indeed, assume for contradiction that $X' \cap X_{G'} \neq X'$ (and hence, by Lemma 1.1, the intersection is of dimension at most $\dim X' - 1 = 2n$). By the definition of $X_{G'_{12}}$, for every point $y \in X' \setminus X_{G'}$, the first eight coordinates (which represent the first two lines in the sequence $L(y)$) correspond to a pair of non-intersecting lines. In particular, $\pi^{-1}(y) \cap X$ is one-dimensional, by Lemma 3.5. Hence, using Lemma 1.2, we have

$$\dim X \leq \dim X' + \dim(\pi^{-1}(y) \cap X) \leq 2n + 1 + 1 = 2n + 2,$$

which contradicts our assumption that X is $(2n + 3)$-dimensional. Hence $X' \subset X_{G'}$, as claimed.

So X' is a $(2n+1)$-dimensional irreducible component of $X_{G'}$. By the induction hypothesis, every point of X' corresponds to a sequence of $n-1$ lines which are either all concurrent or all

coplanar. In particular, for the point $y := \pi(x)$, we have that the $n-1$ lines of the sequence $L(y) = (\ell_1, \ldots, \ell_{n-1})$ are either all concurrent or all coplanar. Since $x \in \pi^{-1}(y) \cap X$, we have that $L(x) = (\ell_1, \ldots, \ell_{n-1}, \ell_n)$, where the line ℓ_n intersects ℓ_1, ℓ_2, ℓ_3.

Assume first that $(\ell_1, \ldots, \ell_{n-1})$ are all concurrent. If ℓ_n passes through the common intersection point of ℓ_1, ℓ_2, ℓ_3, which is the same intersection point of $\ell_1, \ldots, \ell_{n-1}$, then the n lines of $L(x)$ are all concurrent. Otherwise, ℓ_1, ℓ_2, ℓ_3 must lie on a common plane with ℓ_n, and so ℓ_1, ℓ_2, ℓ_3 are both concurrent and coplanar. Moreover, by continuity, this is the case for every point ξ in a sufficiently small open neighborhood of $x \in X$. Hence the *local dimension* of $x \in X$ is at most $2n+2$. Indeed, the configurations involve $n-1$ concurrent lines where three of them are coplanar, and an nth line, coplanar with the first three lines. To specify such a configuration, we need three parameters to specify the point o of concurrency, $2(n-2)$ parameters to specify $n-2$ of the lines through o, except for ℓ_3, one to specify ℓ_3, and two to specify ℓ_n, for a total of $3 + 2(n-2) + 1 + 2 = 2n+2$. Since x is a regular point of X, its local dimension is (well defined and) equals to $\dim X = 2n+3 > 2n+2$. This contradiction implies that the n lines of $L(x)$ are all concurrent in this case.

Similarly, assume that $(\ell_1, \ldots, \ell_{n-1})$ are all coplanar. If ℓ_n lies on the plane h that supports ℓ_1, ℓ_2, ℓ_3, which is the same plane that supports $\ell_1, \ldots, \ell_{n-1}$, then the n lines of $L(x)$ are all coplanar. Otherwise, $\ell_1, \ell_2, \ell_3, \ell_n$ must be concurrent, where their common point is the unique intersection point of ℓ_n with h. So ℓ_1, ℓ_2, ℓ_3 are both concurrent and coplanar. Moreover, by continuity, this is the case for every point ξ in a sufficiently small open neighborhood of $x \in X$. But then the local dimension of $x \in X$ is at most $2n+2$ (here we have $n-1$ coplanar lines where three of them are also concurrent, and an nth line concurrent with the first three lines, and the analysis is symmetric to the one given above). Since x is a regular point of X, this yields a contradiction, as above. Thus the n lines of $L(x)$ are all coplanar in this case.

To recap, we have shown so far that in case $x \in X$ is a regular point, $L(x)$ is a sequence of n lines that are either all concurrent or all coplanar. By continuity, this is the case for every point of X. This establishes the induction step, and thus completes the proof. ◀

5 Rigidity of planar embeddings of graphs

5.1 Definitions and basic properties

In this section we introduce some basic definitions of the classical notion of combinatorial rigidity of graphs, focusing on planar embeddings. For more details about the notions being reviewed here, see [1, 2] and references therein.

Let $G = (V, E)$ be a graph with n vertices and m edges, and write $(v_1, \ldots, v_n)$ and $(e_1, \ldots, e_m)$ for the vertices and edges of G, respectively. Let $\mathbf{p} : V \mapsto \mathbb{R}^2$ be an injection, referred to as a (planar) *embedding* of G. We often identify an embedding $\mathbf{p}$ with a point in $\mathbb{R}^{2n}$, in the obvious way. With this identifications, we define $f_G : \mathbb{R}^{2n} \to \mathbb{R}^m$ by

$$f_G(\mathbf{p}) = (\ldots, \|\mathbf{p}(v_i) - \mathbf{p}(v_j)\|^2, \ldots) \in \mathbb{R}^m,$$

for each point $\mathbf{p} \in \mathbb{R}^{2n}$, where the entries correspond to the edges $(v_i, v_j) \in E$ in their prescribed order, and where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^2$. Note that f_G is well defined even for points $\mathbf{p} \in \mathbb{R}^{2n}$ that correspond to embeddings which are not injective. We refer to f_G as the *edge function* of the graph G.

For a graph G, with n vertices and m edges, let $r(G) := \max\{\mathrm{rank} J_{f_G}(\mathbf{p}) \mid \mathbf{p} \in \mathbb{R}^{2n}\}$, where J_{f_G} stands for the $m \times (2n-3)$ Jacobian matrix of f_G. Then G is called *rigid* if $r(G) = 2n-3$, and *flexible*, otherwise (see Asimow and Roth [1] for equivalent definitions

and for more details). Equivalently, G is rigid if and only if the image of f_G over $\mathbb{R}^{2n}$ forms a $(2n-3)$-dimensional algebraic variety in $\mathbb{R}^m$. Note that a graph with n vertices and $m < 2n-3$ edges is never rigid. A graph G is called *minimally rigid* if it is rigid and has exactly $m = 2n-3$ edges.

We have the following result from rigidity theory.

▶ **Lemma 5.1.** *If G is rigid, then there exists a subgraph $G' \subset G$ which is Laman.*

▶ Remark. The other direction of Lemma 5.1 is true too ([3]), but we prove it independently (see Theorem 7.1).

We say that a point $\mathbf{p} \in \mathbb{R}^{2n}$ is a *regular* embedding of G if $\mathrm{rank} J_{f_G}(\mathbf{p}) = r(G)$, and *singular*, otherwise. We say that a point $\mathbf{p} \in \mathbb{R}^{2n}$ is a *generic* embedding of G if $\mathbf{p}$ is regular and $\mathbf{y} := f_G(\mathbf{p})$ is a regular point of the variety $I := f_G(\mathbb{R}^{2n})$. A graph G is called *globally rigid* if, for every pair $\mathbf{p}, \mathbf{p}' \in \mathbb{R}^{2n}$ of generic embeddings of G such that $f_G(\mathbf{p}) = f_G(\mathbf{p}')$, the sets $\mathbf{p}(V)$ and $\mathbf{p}'(V)$ are congruent.

▶ Remark. We note that the standard definition (see, e.g., Connelly [2]) of global rigidity refers only to embeddings $\mathbf{p}$ of a graph G which are "generic" in the sense that their coordinates are algebraically independent over $\mathbb{Q}$. In our definition we consider a larger set of embeddings of G and consider them as generic. As we will see, both definitions yield the same family of globally rigid graphs. So our definition is better, in the sense that it applies to a larger set of embeddings of G.

We have the following result from rigidity theory.[3]

▶ **Lemma 5.2** (Hendrickson [7])**.** *If G is globally rigid, then G is Hendrickson.*

The other direction, namely, that every Hendrickson graph G is globally rigid, follows by combining the two results of Connelly [2] and of Jackson and Jordan [8]. The analysis in this paper reproves this fact (extending it slightly, by showing it applies to a larger set of "generic" embeddings), using only [8], and bypassing, or finding alternative proof, for the result from [2].

5.2 Pairs of embeddings that induce the same edge distances

We consider the following (real) variety

$$V_G := \{(\mathbf{p}, \mathbf{p}') \in (\mathbb{R}^{2n})^2 \mid f_G(\mathbf{p}) = f_G(\mathbf{p}')\}.$$

Note that $\dim_{\mathbb{R}} V_G \geq 2n+3$, for every graph G on n vertices, simply since it contains the subvariety

$$\{(\mathbf{p}, \mathbf{p}') \in (\mathbb{R}^{2n})^2 \mid \mathbf{p}' \text{ is congruent to } \mathbf{p}\},$$

and the latter has (real) dimension $2n+3$, as can be easily verified.

We have the following property.

▶ **Lemma 5.3.** *Let G be a graph on n vertices. If $\dim_{\mathbb{R}} V_G = 2n+3$, then G is rigid.*

▶ Remark. Later (Theorem 7.1) we show that in fact $\dim_{\mathbb{R}} V_G = 2n+3$ if and only if G is rigid.

[3] Note that Lemma 5.2 applies also to our notion of global rigidity, since (a priori) our notion is more restrictive.

Proof. Assume that G is flexible. Let $\pi : \mathbb{R}^{4n} \to \mathbb{R}^{2n}$ denote the projection of $\mathbb{R}^{4n}$ onto its first $2n$ coordinates. Clearly, $\pi(V_G) = \mathbb{R}^{2n}$. We show that, for every $\mathbf{p} \in \mathbb{R}^{2n}$, the preimage $\pi^{-1}(\mathbf{p}) \cap V_G$ is at least four-dimensional. As above, for an embedding $\mathbf{p} \in \mathbb{R}^{2n}$ of G, define the variety

$$T(\mathbf{p}) := \{\mathbf{p}' \in \mathbb{R}^{2n} \mid \mathbf{p}' \text{ is congruent to } \mathbf{p}\}.$$

As was already noted, for $\mathbf{p}$ fixed, $T(\mathbf{p})$ is 3-dimensional.

Let $\mathbf{p}_0$ be a regular embedding of G. By our assumption that G is flexible, there exists a continuous path $\mathbf{q}(t)$, $t \in [0,1)$, such that $\mathbf{q}(0) = \mathbf{p}_0$, $\mathbf{q}(t) \in f_G^{-1}(f_G(\mathbf{p}_0))$, and $\mathbf{q}(t)$ is not congruent to $\mathbf{p}_0$, for every $t \in (0,1)$ (see Asimow and Roth [1, Proposition 1]). Then the set

$$\{\mathbf{q} \mid t \in (0,1), \mathbf{q} \text{ is congruent to } \mathbf{q}(t))\} = \bigcup_{t \in (0,1)} T(\mathbf{q}(t))$$

is contained in $\pi^{-1}(\mathbf{p}) \cap V_G$ and forms a four-dimensional real manifold, as is not hard to verify (note that the sets in the union are pairwise disjoint). Thus, $\dim_{\mathbb{R}} \pi^{-1}(\mathbf{p}) \cap V_G \geq 4$ for every regular embedding $\mathbf{p}$ of G, which implies that $\dim_{\mathbb{R}} V_G \geq 2n+4$. This completes the proof of the lemma. ◀

▶ **Lemma 5.4.** *Let G be a graph on n vertices. Suppose that every irreducible component of V_G which has maximal dimension is contained in*

$$\{(\mathbf{p}, \mathbf{p}') \in (\mathbb{R}^{2n})^2 \mid \mathbf{p}' \text{ is congruent to } \mathbf{p}\}.$$

Then G is globally rigid.

Proof. By assumption $\dim V_G = 2n+3$, and so, by Lemma 5.3, G is rigid.

Let $\mathbf{p}_0, \mathbf{p}_0'$ be generic embeddings of G such that $f_G(\mathbf{p}_0) = f_G(\mathbf{p}_0')$. We claim that the pair $(\mathbf{p}_0, \mathbf{p}_0')$ lies on an irreducible component of V_G of maximal dimension (that is, of dimension $2n+3$). Put $\mathbf{y}_0 := f_G(\mathbf{p}_0) = f_G(\mathbf{p}_0')$ and $I := f_G(\mathbb{R}^{2n})$. Let $N, N' \subset \mathbb{R}^{2n}$ be some open neighborhood of $\mathbf{p}_0, \mathbf{p}_0'$, respectively. Since each of $\mathbf{p}_0$ (resp., $\mathbf{p}_0'$) is generic, taking N (resp., N') to be sufficiently small, we may assume that the image of $f_{G|_N}$ (resp., $f_{G|_{N'}}$), the restriction of f_G to N (resp., to N'), is $(2n-3)$-dimensional. Moreover, we may assume that $f_G(N) = f_G(N')$. Indeed, each of $f_G(N), f_G(N')$ is a (relatively) open neighborhood of $\mathbf{y}_0$ in I, so their intersection (since it is nonempty) must be open.

Put $M := f_G(N) = f_G(N')$. By what have just been argued, M is a $(2n-3)$-dimensional neighborhood of $\mathbf{y}_0$ in I, and, for every $\mathbf{y} \in M$, we have $(\mathbf{p}, \mathbf{p}') \in V_G \cap (N \times N')$, where $\mathbf{p} \in f_{G|_N}^{-1}(\mathbf{y})$, $\mathbf{p}' \in f_{G|_{N'}}^{-1}(\mathbf{y})$, and each of $f_{G|_N}^{-1}(\mathbf{y}) \cap N$ and $f_{G|_{N'}}^{-1}(\mathbf{y}) \cap N'$ is 3-dimensional. In other words, $V_G \cap (N \times N')$ is a neighborhood of $(\mathbf{p}_0, \mathbf{p}_0')$ in V_G which is of dimension $2n-3+6 = 2n+3$. So $(\mathbf{p}_0, \mathbf{p}_0')$ necessarily lies on an irreducible component of V_G of maximal dimension. This establishes the claim.

Our assumption then implies that $\mathbf{p}_0$ is congruent to $\mathbf{p}_0'$. Since this is true for every pair $(\mathbf{p}_0, \mathbf{p}_0')$ of generic embeddings of G, the lemma follows. ◀

6 Reduction to line incidences in three dimensions

We apply the Elekes–Sharir framework (see [4, 5]) to connect the notion of graph rigidity of planar structures, discussed in Section 5, with line configurations in $\mathbb{R}^3$ (or, rather, in the

real projective 3-space[4]). Specifically, we represent each orientation-preserving rigid motion of the plane (called a *rotation* in [4, 5]) as a point $(c, \cot(\theta/2))$ in $\mathbb{R}^3$, where c is the center of rotation, and θ is the (counterclockwise) angle of rotation. (Note that pure translations are mapped in this manner to points at infinity.) Given a pair of distinct points $a, b \in \mathbb{R}^2$, the locus of all rotations that map a to b is a line $\ell_{a,b}$ in the above parametric 3-space, given by the parametric equation

$$\ell_{a,b} = \{(u_{a,b} + tv_{a,b},\ t) \mid t \in \mathbb{R}\}, \tag{2}$$

where $u_{a,b} = \frac{1}{2}(a+b)$ is the midpoint of ab, and $v_{a,b} = \frac{1}{2}(a-b)^{\perp}$ is a vector orthogonal to $\vec{ab}$ of length $\frac{1}{2}\|a-b\|$, with $\vec{ab}$, $v_{a,b}$ positively oriented (i.e., $v_{a,b}$ is obtained by turning $\vec{ab}$ counterclockwise by $\pi/2$).

It is instructive to note (and easy to verify) that every non-horizontal line ℓ in $\mathbb{R}^3$ can be written as $\ell_{a,b}$, for a unique (ordered) pair $a, b \in \mathbb{R}^2$. More precisely, if ℓ is also non-vertical, the resulting a and b are distinct. If ℓ is vertical, then a and b coincide, at the intersection of ℓ with the xy-plane, and ℓ represents all rotations of the plane about this point.

A simple yet crucial property of this transformation is that, for any pair of pairs (a, b) and (c, d) of points in the plane, $\|a-c\| = \|b-d\|$ if and only if $\ell_{a,b}$ and $\ell_{c,d}$ intersect, at (the point representing) the unique rotation τ that maps a to b and c to d. This also includes the special case where $\ell_{a,b}$ and $\ell_{c,d}$ are parallel, corresponding to the situation where the transformation that maps a to b and c to d is a pure translation (this is the case when $\vec{ac}$ and $\vec{bd}$ are parallel (and of equal length)).

Note that no pair of lines $\ell_{a,b}$, $\ell_{a,c}$ with $b \neq c$ can intersect (or be parallel), because such an intersection would represent a rotation that maps a both to b and to c, which is impossible.

▶ **Lemma 6.1.** *Let $L = \{\ell_{a_i,b_i} \mid a_i, b_i \in \mathbb{R}^2,\ i = 1, \ldots, r\}$ be a collection of $r \geq 3$ (non-horizontal) lines in $\mathbb{R}^3$.*

- **(a)** *If all the lines of L are concurrent, at some common point τ, then the sequences $A = (a_1, \ldots, a_r)$ and $B = (b_1, \ldots, b_r)$ are congruent, with equal orientations, and τ (corresponds to a rotation that) maps a_i to b_i, for each $i = 1, \ldots, r$.*
- **(b)** *If all the lines of L are coplanar, within some common plane h, then the sequences $A = (a_1, \ldots, a_r)$ and $B = (b_1, \ldots, b_r)$ are congruent, with opposite orientations, and h defines, in a unique manner, an orientation-reversing rigid motion h^* that maps a_i to b_i, for each $i = 1, \ldots, r$.*
- **(c)** *If all the lines of L are both concurrent and coplanar, then the points of A are collinear, the points of B are collinear, and A and B are congruent.*

Proof.

- **(a)** This is an immediate consequence of the properties of the Elekes–Sharir transformation, as noted above.
- **(b)** For each pair of indices $i \neq j$, the lines ℓ_{a_i,b_i} and ℓ_{a_j,b_j} intersect or are parallel. Hence $\|a_i - a_j\| = \|b_i - b_j\|$. This implies that the sequences A and B are congruent. Assume that the lines ℓ_{a_i,b_i} are not all concurrent (this is the case that will be addressed in part (c)). That is, there is no orientation-preserving rigid motion that maps A to B, so A and B must have opposite orientations, and the unique rigid motion h^* that maps A to B is orientation-reversing.

[4] To simplify the presentation, we continue to work in the affine $\mathbb{R}^3$, but the extension of the analysis to the projective setup is straightforward. Issues related to this extension will be noted throughout the analysis.

(c) As in the previous cases, $\|a_i - a_j\| = \|b_i - b_j\|$ for every i, j, and hence the sequences A and B are necessarily congruent. As in part (a), since the lines of L are concurrent, there exists an orientation-preserving rigid motion τ that maps A to B. In our case the lines of L also lie on a common plane h, and thus any line λ in h intersects each of the lines of L. Choose λ in h so that the lines of $L \cup \{\lambda\}$ are not all concurrent. Note that we have $\lambda = \ell_{a',b'}$, for some $a', b' \in \mathbb{R}^2$, as every line in $\mathbb{R}^3$ can be interpreted in this way. By part (b), applied to the set $L \cup \{\lambda\}$, there exists an orientation-reversing rigid motion h^* that (in particular) maps the A to B. So B is the image of A (as sequences) under an orientation-preserving rigid motion, and also under an orientation-reversing rigid motion, which can happen only if both sets are collinear. ◀

7 Necessity of our conditions

In this section we show that the conditions in Theorem 3.2 and Theorem 4.3 are not only sufficient, but also necessary. That is, we have the following statements.

▶ **Theorem 7.1.** *Let G be a graph on n vertices. Then X_G is $(2n+3)$-dimensional if and only if there exists a subgraph $G' \subset G$ which is Laman.*

▶ **Theorem 7.2.** *Let G be a graph on n vertices. Then G is Hendrickson if and only if every irreducible component of maximal dimension $X \subset X_G$ is contained in X_{K_n}.*

The following observation says that $X_G \cap \mathbb{R}^{4n}$ contains an isomorphic copy of V_G.

▶ **Lemma 7.3.** *Let $G = (V, E)$ be a graph on n vertices and m edges. Then there exists a polynomial mapping $\varphi : (\mathbb{R}^{2n})^2 \to \mathbb{R}^{4n}$ such that $\varphi(V_G) \subset X_G \cap \mathbb{R}^{4n}$.*

Proof. Define $\varphi : (\mathbb{R}^{2n})^2 \to \mathbb{R}^{4n}$ by

$$((p_1, \ldots, p_n), (p'_1, \ldots, p'_n)) \mapsto (\ell_{p_1,p'_1}, \ldots, \ell_{p_n,p'_n}).$$

We claim that $\varphi(V_G) \subset X_G \cap \mathbb{R}^{4n}$. Indeed, by the definition of V_G, $\|p_i - p_j\| = \|p'_i - p'_j\|$, for every $(i, j) \in E$. By the properties reviewed in Section 6, the lines ℓ_{p_i,p'_i} and ℓ_{p_j,p'_j} then must intersect, for every $(i, j) \in E$, and thus $(\ell_{p_1,p'_1}, \ldots, \ell_{p_n,p'_n}) \in X_G$. Since the points p_i, p'_i, for $i = 1, \ldots, n$, have real coordinates, the representation of each of the lines ℓ_{p_i,p'_i}, for $i = 1, \ldots, n$, as points in $\mathbb{C}^4$, requires only real coefficients. Thus $(\ell_{p_1,p'_1}, \ldots, \ell_{p_n,p'_n}) \in X_G \cap \mathbb{R}^{4n}$, as claimed. ◀

We are now ready to prove Theorem 7.1 and Theorem 7.2.

Proof of Theorem 7.1. Let G be a graph on n vertices. If G contains a subgraph G' which is Laman, then X_G is $(2n+3)$-dimensional, by Theorem 3.2 (and the remark following it).

We now prove the opposite direction. For contradiction, assume that X_G is $(2n+3)$-dimensional, and that G does not contain a subgraph which is Laman. By Lemma 5.1, G is flexible. Using Lemma 5.3 (and the fact that $\dim V_G \geq 2n+3$, for every graph G), we get that $\dim V_G \geq 2n+4$. Lemma 7.3 then implies that also $\dim_{\mathbb{C}} X_G \geq \dim_{\mathbb{R}} X_G \geq 2n+4$. This contradicts our assumption about the dimension of X_G, and by this completes the proof. ◀

Proof of Theorem 7.2. Let G be a graph on n vertices. If G is Hendrickson, then every irreducible component of maximal dimension $X \subset X_G$ is contained in X_{K_n}, by Theorem 4.3.

For the opposite direction, assume that every irreducible component of maximal dimension $X \subset X_G$ is contained in X_{K_n}. Since

$$X_{K_n} \cap \mathbb{R}^{4n} \subset \varphi(V_G) \subset X_G \cap \mathbb{R}^{4n},$$

it follows that irreducible components of V_G which have maximal dimension are exactly $X_{K_n} \cap \mathbb{R}^{4n}$. Thus G is globally rigid, by Lemma 5.4. Using Lemma 5.2, we get that G is Hendrickson, as asserted. ◀

8 Applications

8.1 Standard rigidity

We reprove the following theorem of Connelly [2] (see also [9] for a simplification of the proof in [2]).

▶ **Theorem 8.1.** *If G is obtained from K_4 by a sequence of edge additions and 1-extensions, then G is globally rigid in $\mathbb{R}^2$.*

Proof. This follows by combining Theorem 4.3, Lemma 5.4 and Lemma 7.3. ◀

▶ **Lemma 8.2.** *Let $G = (V, E)$ be a graph with $|E| = \Omega(n^{3/2})$, and suppose that the largest independent set in G is of size $o(n^{1/2}/\log n)$. Then, for every pair of embeddings $\mathbf{p}, \mathbf{q}$ of G in $\mathbb{R}^2$, with $f_G(\mathbf{p}) = f_G(\mathbf{q})$, there exist a subsequence $\mathbf{p}' \subset \mathbf{p}$ of size $\Omega(n^{1/2}/\log n)$, and a matching subsequence $\mathbf{q}' \subset \mathbf{q}$, such that $\mathbf{p}'$ and $\mathbf{q}'$ are congruent.*

Proof. This follows from the reduction in Section 6 and the incidence bound between lines and points in $\mathbb{R}^3$ obtained by Guth and Katz [5]. ◀

8.2 Rigidity and global rigidity on a two-dimensional sphere

Consider the rigidity problem, where this time the vertices of a graph G are embedded to the unit sphere in $\mathbb{R}^3$. Note that isometries of the sphere can be represented as points in $\mathbb{R}^3$ (see [12]). Thus, in view of our general results about lines, we obtain the following corollaries.

▶ **Corollary 8.3.** *Consider embeddings of graphs in the unit sphere $\S^2 \subset \mathbb{R}^3$. Then a graph G is rigid if and only if there exists a subgraph $G' \subset G$ which is Laman.*

▶ **Corollary 8.4.** *Consider embeddings of graphs in the unit sphere $\S^2 \subset \mathbb{R}^3$. Then a graph G is globally rigid if and only if there exists a subgraph G is Hendrickson.*

8.3 Other reductions to line configurations

In the full version of this paper we include some variants of the rigidity problem, that can be reduced to the problems about line configurations studied here. One such variant is when edge-lengths are replaced by edge-slopes. This problem was previously studied by Martin [11]. For this case we again obtain the same characterization for rigidity and global rigidity, as in the standard rigidity notion, because both problems correspond to the same question about line configurations.

References

1 L. Asimow and B. Roth. The rigidity of graphs. *Trans. Amer. Math. Soc.*, 245:279–289, 1978.
2 R. Connelly. Generic global rigidity. *Discrete Comput. Geom.*, 33:549–563, 2005.
3 H. Crapo. Structural rigidity. *Structural Topology*, 1:26–45, 1979.
4 Gy. Elekes and M. Sharir. Incidences in three dimensions and distinct distances in the plane. *Combinat. Probab. Comput.*, 20:571–608, 2011.
5 L. Guth and N. H. Katz. On the Erdős distinct distances problem in the plane. *Annals Math.*, 18:155–190, 2015.
6 R. Hartshorne. *Algebraic Geometry*. Springer-Verlag, New York, 1977.
7 B. Hendrickson. *Conditions for unique graph embeddings*. Technical Report 88-950, Department of Computer Science, Cornell University, 1988.
8 B. Jackson and T. Jordán. Connected rigidity matroids and unique realizations of graphs. *J. Combinat. Theory, Ser. B*, 94:1–29, 2005.
9 B. Jackson, T. Jordán, and Z. Szabadka. Globally linked pairs of vertices in equivalent realizations of graphs. *Discrete Comput. Geom.*, 35:493–512, 2006.
10 G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:333–338, 1970.
11 J.L. Martin. Geometry of graph varieties. *Trans. Amer. Math. Soc.*, 355:4151–4169, 2003.
12 T. Tao. Lines in the Euclidean group $SE(2)$. *Blog post, available at* `https://terrytao.wordpress.com/2011/03/05/lines-in-the-euclidean-group-se2/`.

A Tools from algebraic geometry

We use the following basic facts from algebraic geometry theory.

▶ **Lemma 1.1** ([6, Proposition I.7.1]). *Let $X \subset \mathbb{C}^D$ be an irreducible variety of dimension k and $f \in \mathbb{C}[z_1, \ldots, z_D]$. Then $X \cap Z(f)$ is either X, the empty set, or has dimension $k-1$.*

▶ **Lemma 1.2.** *Let π denote the projection of $\mathbb{C}^D$ onto its first k coordinates and let X be an irreducible subvariety of $\mathbb{C}^D$. Let Y denote the Zariski closure of $\pi(X)$. Then*

$$\dim X \leq \dim Y + \dim(\pi^{-1}(y) \cap X)$$

for every $y \in \pi(X)$.

▶ **Lemma 1.3.** *Let π denote the projection of $\mathbb{C}^D$ onto its first k coordinates and let $S \subset \mathbb{C}^d$ be any subset. Then $\mathrm{Cl}(\pi(\mathrm{Cl}(S))) = \mathrm{Cl}(\pi(S))$.*

Weak $\frac{1}{r}$-Nets for Moving Points*

Alexandre Rok[1] and Shakhar Smorodinsky †[2]

1 **Department of Mathematics, Ben-Gurion University of the NEGEV, Be'er Sheva 84105, Israel**
rok@math.bgu.ac.il

2 **Department of Mathematics, Ben-Gurion University of the NEGEV, Be'er Sheva 84105, Israel**
shakhar@math.bgu.ac.il

Abstract

In this paper, we extend the weak $\frac{1}{r}$-net theorem to a kinetic setting where the underlying set of points is moving polynomially with bounded description complexity. We establish that one can find a kinetic analog N of a weak $\frac{1}{r}$-net of cardinality $O(r^{\frac{d(d+1)}{2}} \log^d r)$ whose points are moving with coordinates that are rational functions with bounded description complexity. Moreover, each member of N has one polynomial coordinate.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics, G.2.2 Graph Theory

Keywords and phrases Hypergraphs, Weak ϵ-nets.

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.59

1 Introduction and Preliminaries

This paper deals with weak $\frac{1}{r}$-nets for convex sets. It is a central notion in discrete geometry. We initiate the study of kinetic weak $\frac{1}{r}$-nets, and extend the classical weak $\frac{1}{r}$-net theorem to a kinetic setting. Our main motivation is the recent result of De Carufel et al. [4] on kinetic hypergraphs.

Before presenting our results, we need a few definitions and well known facts: A pair $(X, \mathcal{S})$, where $\mathcal{S} \subset P(X)$, is called a *set system* or a *hypergraph*. A subset $A \subset X$ is called *shattered* if $\mathcal{S}|_A = 2^A$. The largest size of a shattered subset from X with respect to $\mathcal{S}$ is called the VC-*dimension* of $(X, \mathcal{S})$. The concept of VC-dimension has its roots in statistics. It first appeared in the paper of Vapnik and Chervonenkis in [10]. Nowadays, this notion plays a key role in learning theory and discrete geometry. Given a set system $(X, \mathcal{S})$, we say that $Y \subset X$ is a *strong* $\frac{1}{r}$*-net* if for each $S \in \mathcal{S}$ with $|S| > |X|/r$ we have $S \cap Y \neq \emptyset$. Based on the concept of VC-dimension, Haussler and Welzl provided a link to strong nets by proving that any set system with VC-dimension d has a strong $\frac{1}{r}$-net of size $O(dr \log r)$ [7].

The intersection of all convex sets contaning $X \subset \mathbb{R}^d$, denoted by $\mathrm{conv}(X)$, is called *the convex hull* of X. The *affine hull* of a finite set X, denoted by $\mathrm{aff}(X)$, is the intersection of all affine subspaces containing X. It is well known that $\mathrm{aff}(X) = \{\sum_{i=1}^n \alpha_i x_i : \sum_{i=1}^n \alpha_i = 1 \text{ and } x_i \in X\}$. A set of points $X = \{x_1, \ldots, x_n\}$ is said to be affinely independent if for each $1 \leq i \leq n$ we have $x_i \notin \mathrm{aff}(X \setminus \{x_i\})$. We refer to the convex hull of an affinely independent

* Work was partially supported by Grant 1136/12 from the Israel Science Foundation
† Work by this author was partially supported by Swiss National Science Foundation Grants 200020144531 and 200021-137574.

© Alexandre Rok and Shakhar Smorodinsky;
licensed under Creative Commons License CC-BY

set S as $(|S|-1)$-*dimensional simplex spanned by* S. A simplex S is *spanned by* P if it arises from some subset of P.

1.1 Weak $\frac{1}{r}$-nets

We now study the notion of weak $\frac{1}{r}$-net in a kinetic setting. Let us first recall the concept of weak $\frac{1}{r}$-net in the static case.

▶ **Definition 1** (Weak $\frac{1}{r}$-net)**.** Let $P \subset \mathbb{R}^d$ be a finite set of points and $r \geq 1$. A set $N \subset \mathbb{R}^d$ is said to be a weak $\frac{1}{r}$-net for P if every convex set containing $> \frac{1}{r}|P|$ points of P also contains a point of N.

The following theorem is one of the major milestones in modern discrete geometry:

▶ **Theorem 2** (Weak $\frac{1}{r}$-net Theorem [1, 5, 8])**.** *Let $r \geq 1$ and $d \geq 1$ an integer. Then there exists a least integer $f(r,d)$ such that for every finite set $P \subset \mathbb{R}^d$ there is a weak $\frac{1}{r}$-net of size at most $f(r,d)$.*

The existence of $f(r,d)$ was first proved by Alon et al. [1] with the bounds $f(r,2) = O(r^2)$ and $f(r,d) = O(r^{(d+1)(1-\frac{1}{s_d})})$ for $d \geq 3$, where s_d tends to 0 exponentially fast. Later, better bounds on $f(r,d)$ for $d \geq 3$ were obtained by Chazelle at al. in [5], who showed that $f(r,d) = O(r^d \log^{b_d} r)$, where b_d is roughly $2^{d-1}(d-1)!$. The current best known upper bound for $d \geq 3$ due to Matoušek and Wagner [8] is $f(r,d) = O(r^d \log^{c(d)} r)$, where $c(d) = O(d^2 \log d)$, and $f(r,2) = O(r^2)$ [1]. The best known lower bound was provided by Bukh, Matoušek, and Nivasch [3], who showed that $f(r,d) = \Omega(r \log^{d-1} r)$ for $d \geq 2$.

Recently, some interesting connections were found between strong and weak nets. In particular, Mustafa and Ray [9] showed how one can construct weak $\frac{1}{r}$-nets from strong $\frac{1}{r}$-nets. They obtained a bound of $O(r^3 \log^3 r)$ in $\mathbb{R}^2$, $O(r^5 \log^5 r)$ in $\mathbb{R}^3$, and $O(r^{d^2} \log^{d^2} r)$ for $d \geq 4$ on the size of weak $\frac{1}{r}$-nets.

A kinetic framework: The problem of finding strong $\frac{1}{r}$-nets has been recently considered in a kinetic setting by De Carufel et al. [4]. Their work and extensive research in the static case motivates us to consider the problem of weak $\frac{1}{r}$-net in a kinetic setting.

Let us define this setting: The dimension $d \geq 1$ is assumed to be fixed. A *moving point* is a function from $\mathbb{R}_+$ to $\mathbb{R}^d \cup \{\emptyset\}$ for some $d \geq 1$. A *point p moving in* $\mathbb{R}^d$ is simply a moving point whose codomain is $\mathbb{R}^d \cup \{\emptyset\}$ and such that $p(t) \in \mathbb{R}^d$ for some $t \geq 0$. In this paper, we are interested in the case where this function is polynomial or rational, i.e., each coordinate is a polynomial or a rational function. If one of the coordinates is not defined for some t, then the moving point is not defined at t. For simplicity, we often use the term *point* for a moving point if there is no confusion. In what follows, the dimension d is assumed to be fixed. For a set P of moving points and a "time" $t \in \mathbb{R}_+$, we denote by $P(t)$ the set $\{p(t) | p \in P\}$. We say that a set P of moving points in $\mathbb{R}^d$ has *bounded description complexity* β if for each point $p(t) = (p_1(t), \ldots, p_d(t))$, each $p_i(t)$ is a rational function with both numerator and denominator having degree at most β.

We say that the function h with domain $\mathbb{R}_+$ is a *moving affine subspace* if for some integer k and any $t \geq 0$, $h(t)$ is an affine subspace of dimension k or the emptyset. In the case $h(t)$ is not always equal to the emptyset, we also say that such a h has *dimension* k. If the dimension is 1 or $d-1$ we refer to the corresponding moving affine subspaces as *moving line* and *moving hyperplane*, respectively. For simplicity, we often write *moving subspace* instead of moving affine subspace. We now introduce some notations to define affine subspaces.

We say that $\tilde{h}$ is given by $x_1 = p_1, \ldots, x_k = p_k$ if $\tilde{h} = \{x \in \mathbb{R}^d : \text{for } 1 \leq i \leq k,\ x_i = p_i\}$. Analogously, we say that a moving affine subspace h is given by $x_1 = p_1, \ldots, x_k = p_k$, where each p_i is a point moving in $\mathbb{R}$, if $h(t)$ is given by $x_1 = p_1(t), \ldots, x_k = p_k(t)$. Similarly to moving points, if a moving subspace h is given by $x_1 = p_1, \ldots, x_k = p_k$, where each p_i is a point moving in $\mathbb{R}$, and $p_i(t)$ is not defined for some $t \geq 0$, then $h(t)$ is not defined.

Finally, for a set $P = \{p_1, \ldots, p_n\}$ of points moving in $\mathbb{R}^d$ and a vector space $V \subset \mathbb{R}^d$, we say that $P' = \{p'_1, \ldots, p'_n\}$ is a *projection of P onto V* if $p'_i(t) = \text{proj}_V(p_i(t))$ for all $t \geq 0$.

▶ **Definition 3** (Kinetic Weak $\frac{1}{r}$-net)**.** Given a set P of n points moving in $\mathbb{R}^d$, we say that a set of moving points N is a kinetic weak $\frac{1}{r}$-net for P if for any $t \in \mathbb{R}_+$ and any convex set C with $C \cap P(t) > n/r$ we have $C \cap N(t) \neq \emptyset$.

We sometimes abuse the notation and write *net* or *weak net* instead of kinetic weak net. In order to establish our result regarding kinetic weak nets, we need the following natural general position assumption on the set P of moving points: We assume that for any $t \geq 0$ the affine hull of any d-tuple of points in $P(t)$ is a hyperplane, but no $d+2$ points of $P(t)$ are contained in a hyperplane. The latter can easily be relaxed to no $c(d) \geq d+2$ points in a hyperplane.

Under these assumptions, we prove the following theorem that could be viewed as a generalization of Theorem 2:

▶ **Theorem 4** (Kinetic Weak $\frac{1}{r}$-net Theorem)**.** *For every pair of integers $d \geq 1$, β and every $r \geq 1$, there exist a least integer $c(r, d, \beta)$ and $g(d, \beta)$ such that for every finite set P of points moving in $\mathbb{R}^d$ with description complexity β there is a kinetic weak $\frac{1}{r}$-net of cardinality at most $c(r, d, \beta)$ and description complexity $g(d, \beta)$. Moreover, for fixed d and β and $r \geq 2$, we have $c(r, d, \beta) = O(r^{\frac{d(d+1)}{2}} \log^d r)$.*

Furthermore, in the case where the points of P move polynomially, the moving points of the kinetic weak $\frac{1}{r}$-net have one polynomial coordinate. This is an important advantage of our construction as many naturally defined moving points, obtained by intersecting moving affine spaces, have no polynomial coordinates.

2 Weak $\frac{1}{r}$-net in a Kinetic Setting

2.1 Points moving in $\mathbb{R}$

In a kinetic setting, one needs to capture the combinatorial changes occurring with time. The concept of *kinetic hypergraph* defined below was introduced in [4] by De Carufel et al.

▶ **Definition 5** (Kinetic Hypergraph)**.** Let P be a set of points moving in $\mathbb{R}^d$ with bounded description complexity and let $\mathcal{R}$ be a set of ranges. We denote by $(P, \mathcal{S})$ the kinetic hypergraph of P with respect to $\mathcal{R}$. Namely, $S \in \mathcal{S}$ if and only if there exists an $R \in \mathcal{R}$ and a "time" $t \in \mathbb{R}_+$ such that $S(t) = R \cap P(t)$. We sometimes abuse the notation, and denote by $(P, \mathcal{R})$ the kinetic hypergraph $(P, \mathcal{S})$.

Figure 1 illustrates the concept of a kinetic hypergraph for $d = 1$ and $\mathcal{R}$ being the family of intervals. De Carufel et al. [4] also established the following important lemma to investigate strong $\frac{1}{r}$-nets in a kinetic setting.

▶ **Lemma 6** (De Carufel et al. [4])**.** *Let $\mathcal{R}$ be a collection of semi-algebraic sets in $\mathbb{R}^d$, each of which can be expressed as a Boolean combination of a constant number of polynomial equations and inequalities of maximum degree c, where c is some constant. Let P be a family*

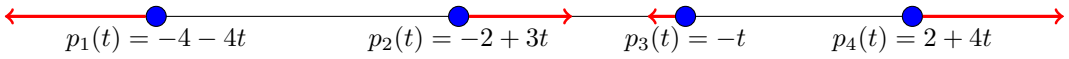


Figure 1 A family $P = \{p_1, p_2, p_3, p_4\}$ of points moving linearly along the the real line. One can easily see that the kinetic hypergraph of P with respect to intervals is $(P, 2^P \setminus \{\{p_1, p_2, p_4\}, \{p_1, p_3, p_4\}, \{p_1, p_4\}\})$.

of points moving polynomially in $\mathbb{R}^d$ with bounded description complexity. Then the kinetic hypergraph of P with respect to $\mathcal{R}$ has bounded VC*-dimension.*

Unfortunately, this is not enough for our purposes, since we need to assume that the moving points can be described with coordinates which are rational functions. However, by following a similar scheme it is not hard to prove the lemma below:

▶ **Lemma 7.** *Let P be a finite set of points moving in $\mathbb{R}$ with bounded description complexity, and let $\mathcal{K} = (P, \mathcal{S})$ be the kinetic hypergraph of P with respect to intervals. Then the* VC*-dimension of $\mathcal{K}$ is $O(1)$.*

We start by defining the concept of *primal shatter function*.

▶ **Definition 8.** Let P be a finite set. For a set system $X = (P, \mathcal{S})$ the primal shatter function $\pi_X : \{1, \ldots, |P|\} \to \mathbb{N}$ is defined by

$$\pi_X(m) = \max_{A \subset P:\ |A| = m} |\{A \cap S : S \in \mathcal{S}\}|.$$

First, we establish a link between the primal shatter function and the VC-dimension of a set system. The lemma below is folklore:

▶ **Lemma 9.** *Let P be a finite set, and $X = (P, \mathcal{S})$ be a set system such that $\pi_X(m) \leq cm^k$ (for $k \geq 2$ say), where c is some constant. Furthermore, let d be the* VC*-dimension of X. Then $d = O(k \log k)$.*

Proof. If $d = 0$, then there is nothing to show. Otherwise, $\pi_X(d)$ is defined, and we easily see that $c \geq 2$ since $\pi_X(1) = 2$. Hence, by the definition of the primal shatter function and the lower bound on c, the following inequalities are satisfied $2^d \leq cd^k \leq (cd)^k$. This implies that $d \leq k \log cd$. Obviously, there is a $c' > 0$ (depending only on c) such that

$$c' d^{\frac{1}{2}} \leq \frac{d}{\log cd} \leq k.$$

Hence,

$$d \leq k \log \frac{c}{c'^2} k^2 = k \log \frac{c}{c'^2} + 2k \log k = O(k \log k).$$ ◀

By some pretty elementary arguments one can establish the lemma below.

▶ **Lemma 10.** *Let P be a set of $n \geq 1$ points moving in $\mathbb{R}$ with bounded description complexity β. Then the number of hyperedges in the kinetic hypergraph $\mathcal{K} = (P, \mathcal{S})$ with respect to intervals is at most $c_\beta n^4$ for some $c_\beta > 0$.*

It is easy to see that the bound on the number of hyperedges above is also valid for induced hypergraphs having at least one vertex. Consider an induced hypergraph $(X, \mathcal{S}|_X)$ of $(P, \mathcal{S})$, and let $A = S \cap X$ be a hyperdge of $(X, \mathcal{S}|_X)$ arising from some $S \in \mathcal{S}$. By definition,

there is an interval $[a, b]$ and a $t \geq 0$ such that $P(t) \cap [a, b] = S(t)$. We now show that $[a, b] \cap X(t) = A(t)$. Clearly, $A(t) \subset [a, b]$ otherwise for some $a \in A$ we have $a(t) \notin S(t)$ implying $a \notin S$, hence $A(t) \subset [a, b] \cap X(t)$. Let us prove that $[a, b] \cap X(t) \subset A(t)$. Take an $x(t) \in X(t) \cap [a, b]$, then clearly $x \in S$ implying $x \in S \cap X = A$, so $x(t) \in A(t)$.

This proves that the induced hypergraph $(X, \mathcal{S}|_X)$ is contained in the kinetic hypergraph of X with respect to intervals, hence the bound of Lemma 10 holds for induced hypergraphs that have at least one vertex.

Proof of Lemma 7. The lemma is an immediate corollary of Lemma 9 combined wirh Lemma 10 and the reasoning above. ◀

Together with the well known strong $\frac{1}{r}$-net theorem mentioned in Section 1, Lemma 7 implies:

▶ **Lemma 11.** *Let P be a finite set of points moving in $\mathbb{R}$ with bounded description complexity. Then the kinetic hypergraph of P with respect to intervals has a strong $\frac{1}{r}$-net (for $r \geq 2$ say) of size $O(r \log r)$.*

For technical reasons, we need the two lemmas above without any general position assumption. Hence, for any $t \geq 0$ more than two moving points from P can coincide at t. Later on, we shall use Lemma 11 in order to find weak $\frac{1}{r}$-nets in a kinetic setting.

2.2 Points moving in $\mathbb{R}^d$

The proof of Theorem 12 below is inspired by a construction from Chazelle et al. [6].

The arguments we use are also valid when the set P consists of points with bounded description complexity. However, as explained in the first section, when the motion is polynomial the construction we present has an important feature: One coordinate is a polynomial. In particular, when $d = 2$ the construction below gives a kinetic weak $\frac{1}{r}$-net N of size only $O(r^3 \log^2 r)$ and the first coordinate of each point in N is a polynomial. Note that in the static setting, the best known upper bound on the function $f(r, 2)$, defined in Section 1, is $O(r^2)$, so our bound is only an $O(r \log^2 r)$ factor of it.

We recall the general position assumption made in Section 1: Given a set of moving points P in $\mathbb{R}^d$, for any $t \geq 0$ the affine hull of any d-tuple of points in $P(t)$ is a hyperplane, and no $d + 2$ points of $P(t)$ are contained in a hyperplane.

▶ **Theorem 12** (Weak $\frac{1}{r}$-net in a Kinetic Setting)**.** *Let P be a set of n points moving polynomially in $\mathbb{R}^d$ with bounded description complexity β. Then there exists a kinetic weak $\frac{1}{r}$-net (for $r \geq 2$ say) N of size $O(r^{\frac{d(d+1)}{2}} \log^d r)$ and bounded description complexity. Moreover, the first coordinate of each point of N is a polynomial.*

Proof. The case $d = 1$ is implied by Lemma 11, so we can assume that $d \geq 2$. The method below works for $n \geq cr$, where c is a sufficiently large constant whose existence is proved later. If $n < cr$, then the theorem holds trivially, since one defines the kinetic weak $\frac{1}{r}$-net to be P.

We start by defining N and other structures we need throughout the proof. Later, we show that N is indeed a kinetic weak $\frac{1}{r}$-net for P. The claims regarding the size and the description complexity of N will follow easily from its definition. First, we need to introduce the concept of *moving subspace of step j* for $1 \leq j \leq d$. It will be some specific moving subspace of dimension $d - j$. Moreover, a moving subspace of step $i + 1$ arises from some moving subspace of step i, hence these structures will be defined iteratively. In what follows, we use parameters $\lambda_1, \ldots, \lambda_d$ with $0 < \lambda_i \leq 1$, whose values are specified later.

Call the projection of P onto x_1-axis P_1. Note that P_1 has description complexity β. Choose a strong $\frac{\lambda_1}{r}$-net N_1 for the kinetic hypergraph of P_1 with respect to intervals. Lemma 11 guarantees that one can select N_1 with $|N_1| \leq b_1 r/\lambda_1 \log r/\lambda_1$, where b_1 depends on β. For each point p of N_1, we consider the moving hyperplane such that at any $t \geq 0$ it is orthogonal to x_1-axis and passes through $p(t)$. The moving affine subspaces of step 1 are exactly these moving hyperplanes arising from N_1.

The construction of moving subspaces of step at least 2 is more involved. Assume that we have constructed the moving affine subspaces up to step j satisfying $1 \leq j \leq d-1$. For each moving subspace h of step j, we define F_h to be the set consisting of moving points $p^{h,X}$ for all $(j+1)$-tuples X of P. The position of $p^{h,X}$ at $t \geq 0$ is given by $p^{h,X}(t) = \mathrm{aff}(X(t)) \cap h(t)$ if this intersection contains a single point. A moving point $p^{h,X}$ is not necessarily uniquely defined, but this not a problem for our purposes. One can define it with description complexity $f(j+1)$ for some increasing function $f : \{1, \ldots, d\} \to \mathbb{N}$ such that $f(1) = \beta$. The technical proof of this fact is provided later in Lemma 17.

Next, for each moving subspace h of step j call the projection of F_h onto x_{j+1}-axis P_h. Note that P_h also has description complexity $f(j+1)$. Choose a strong $\frac{\lambda_{j+1}}{r^{j+1}}$-net N_h for the kinetic hypergraph of P_h with respect to intervals. Again, Lemma 11 ensures that one can select N_h with $|N_h| \leq b_{j+1} r^{j+1}/\lambda_{j+1} \log r^{j+1}/\lambda_{j+1}$, where b_{j+1} depends on $f(j+1)$.

If N_h consists of $q_1, \ldots, q_s$, then the moving affine subspaces of step $j+1$ *induced* by h are $\tilde{h}_i$ given by $x_1 = x_{h,1}, \ldots,\ x_j = x_{h,j},\ x_{j+1} = q_i$ for $1 \leq i \leq s$, where $x_{h,k}$ is the moving point giving the k-th coordinate of h. The set of moving subspaces of step $j+1$ is the union of moving subspaces induced by h among all moving subspaces h of step j.

We define the kinetic weak $\frac{1}{r}$-net N to be the union of the moving subspaces of step d. This makes sense, since the moving subspaces of step d have each coordinate specified by some function, so those are moving points. The size of N is at most

$$b_1 \frac{r}{\lambda_1} \log \frac{r}{\lambda_1} b_2 \frac{r^2}{\lambda_2} \log \frac{r^2}{\lambda_2} \ldots b_d \frac{r^d}{\lambda_d} \log \frac{r^d}{\lambda_d} = O(r^{\frac{d(d+1)}{2}} \log^d r).$$

Moreover, for each $v = (v_1, \ldots, v_d)$ of N, the moving point v_i has description complexity $f(i)$. Since f is an increasing function, the moving point v has description complexity $f(d)$.

We start by briefly outlining main ideas of the proof for $d \geq 3$. The case $d = 2$ is much easier, and does not require the inductive step presented below.

Let $t \geq 0$ and let C be a convex set containing $> n/r$ points of $P(t)$. We start by showing that if one chooses an appropriate value for λ_1, then for some moving subspace h of step 1 the set $h(t)$ intersects "a lot" of segments spanned by $C \cap P(t)$.

Next, the inductive step comes. We assume that λ_i were defined up to some $1 \leq j \leq d-2$, and some moving subspace h of step j (of dimension $d-j$) is such that $h(t)$ intersects a "large" number of j-simplices spanned by $C \cap P(t)$. We start finding a static affine subspace s contained in $h(t)$ of dimension $d-j-1$ such that s intersects a "large" number of $(j+1)$-simplices spanned by $C \cap P(t)$. These $(j+1)$-simplices are obtained from the j-simplices intersecting $h(t)$. Then we show that with an appropriate choice of λ_{j+1}, there are two moving subspaces h_1, h_2 of step $j+1$ induced by h such that $h_1(t)$ and $h_2(t)$ are "close" to s, and therefore at least one of them also intersects a "large" number of $(j+1)$-simplices spanned by $C \cap P(t)$, which completes the inductive step.

This way, we establish that one can define λ_i for $1 \leq i \leq d-1$, so that for some moving line l of step $d-1$ "a lot" of $(d-1)$-simplices spanned by $C \cap P(t)$ are intersected by $l(t)$. In particular, from the definition of F_l the segment $C \cap l(t)$ is such that for "many" moving points $p \in F_l$ the point $p(t)$ belongs to it. Hence, the projection of $C \cap l(t)$ (call it I) onto x_d-axis leads to a "heavy" hyperedge in the kinetic hypergraph of P_l with respect to intervals

(because P_l is the projection of F_l). For an appropriate choice of λ_d, there is a point q of the net N_l such that $q(t)$ must be in I. Finally, by construction of N the moving point whose first $d-1$ coordinates are given by l and the last one by q is in N, so $q(t)$ is in C and we are done.

We now proceed with a detailed proof. Let us show that the set N we defined is indeed a kinetic weak $\frac{1}{r}$-net for P for an appropriate choice of λ_i.

Let $t \geq 0$ and let C be any convex set containing at least n/r points from $P(t)$. It is sufficient to assume that C contains exactly n/r points of $P(t)$ (we choose any n/r points of $C \cap P(t)$, and disregard the remaining ones). We will define the parameters λ_i so that C must contain a point of $N(t)$. It is important to notice that these parameters do not depend on C or t.

For technical reasons, for $1 \leq j \leq d-1$ we also prove the existence of $\gamma_j n^{j+1}/r^{j+1}$ j-simplices spanned by $C \cap P(t)$ and intersecting $h(t)$ for some moving subspace h of step j exactly once in their relative interior, where $\gamma_j > 0$ are iteratively defined later. Clearly, this implies for each simplex above that the affine hull of the $j+1$ points of $P(t)$ spanning it intersects $h(t)$ exactly once as well. In particular, if $h(t)$ intersects $\gamma_j n^{j+1}/r^{j+1}$ j-simplices spanned by $C \cap P(t)$ once in their relative interior, then for at least $\gamma_j n^{j+1}/r^{j+1}$ points $p \in F_h$ we have $p(t) \in C \cap h(t)$. This implication is crucial for our purposes, and will be used in order to prove that N is a kinetic net once the parameters λ_i are specified.

We prove the existence of γ_j and define λ_j for $1 \leq j \leq d-1$ by induction. Then, we define λ_d and show that the values λ_j imply that N is a kinetic weak $\frac{1}{r}$-net.

▶ **Lemma 13.** *If $\lambda_1 = 1/4$ and $n > 4r(2d+2)$, then there exists a moving hyperplane h of step 1 such that $h(t)$ intersects at least $n^2/16r^2$ segments spanned by $C \cap P(t)$ once and in their relative interior.*

Proof. Among moving subspaces $\tilde{h}$ of step 1 with the property that $> n/4r$ points of $C \cap P(t)$ have a strictly smaller x_1-coordinate than the intersection of $\tilde{h}(t)$ with x_1-axis, choose a moving space with the smallest intersection point with x_1-axis at t and denote it by h. The moving subspace h exists, since the above defined set of moving subspaces is easily seen to be nonempty. Indeed, let z be the largest real such that at most $n/4r$ points of $C \cap P(t)$ have their x_1-coordinate in $]-\infty, z[$. Then, since from the general position assumption at most $d+1$ points of $C \cap P(t)$ can share the same x_1-coordinate, we deduce that there are at least $n/r - n/4r - d - 1 > n/4r$ points of $C \cap P(t)$ whose x_1-coordinate is in $]z, \infty[$. Since N_1 is a strong $\frac{1}{4r}$-net for the kinetic hypergraph of P_1 with respect to intervals, there should be a point $w \in N_1$ such that $w(t) \in]z, \infty[$ implying the existence of a moving subspace of step 1 whose intersection with x_1-axis at t is $w(t)$. Hence, the above set of moving subspaces is indeed nonempty, so h exists.

Let x denote the intersection point of $h(t)$ with x_1-axis. We now show that we also have at least $n/4r$ points of $C \cap P(t)$ having a strictly bigger x_1-coordinate than x. Indeed, using one more time the hypothesis that no $d+2$ points are contained in a hyperplane, we deduce that the number of points of $C \cap P(t)$ having their x_1-coordinate smaller or equal to x is at most $2n/4r + 2(d+1) < 3n/4r$. To see this, let $\tilde{h}(t)$ be a predecessor of $h(t)$, i.e., a moving hyperplane of step 1 at t whose intersection point with x_1-axis is the biggest one among those having an intersection point with x_1-axis strictly smaller than $h(t)$. The existence of such a hyperplane is again implied by the the definition of N_1. Indeed, we have $> n/4r$ points $p \in P_1$ such that $p(t) \in]-\infty, x[$, so there is a point $w \in N_1$ such that $w(t) \in]-\infty, x[$. Thus, there is a moving hyperplane of step 1 with its x_1-coordinate equal to $w(t)$, which implies the existence of $\tilde{h}(t)$. Similarly, by our choice of $\tilde{h}(t)$, it is easily seen that at most $n/4r$ points

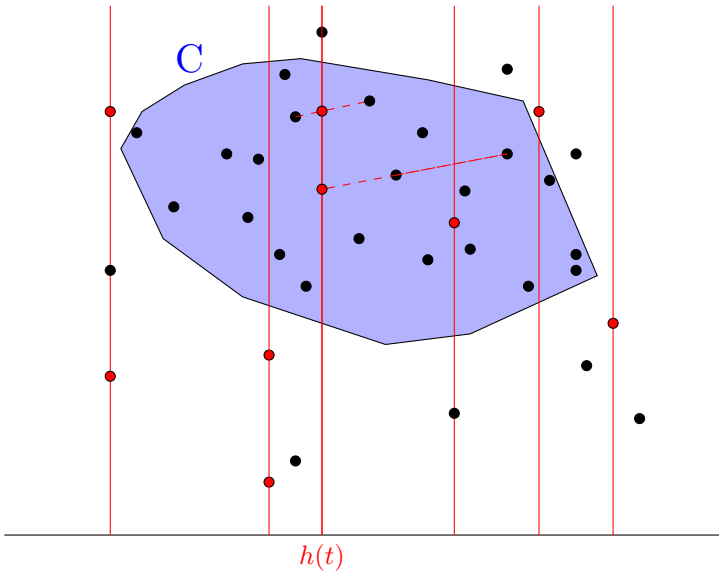


Figure 2 The elements from $P(t)$ and elements of $N(t)$ are black and red dots, respectively. The red lines are moving affine subspaces of step 1 at t. The line $h(t)$ splits $C \cap P(t)$ into two parts of cardinality $> n/4r$. At least one point from the net N induced by h must be in C at t.

of $C \cap P(t)$ are strictly between $h(t)$ and $\tilde{h}(t)$. In summary, by the choice of h, at most $n/4r$ points of $C \cap P(t)$ have their x_1-coordinate strictly smaller than the intersection of $\tilde{h}(t)$ with x_1-axis, at most $n/4r$ are strictly between $h(t)$ and $\tilde{h}(t)$, and at most $d+1$ points lie on each of $h(t)$, $\tilde{h}(t)$.

Hence, both open halfspaces delimited by $h(t)$ contain $> n/4r$ points of $C \cap P(t)$. This means that at least $n^2/16r^2$ segments spanned by $C \cap P(t)$ intersect $h(t)$ once and in their relative interior, so the lemma follows. ◀

The lemma above implies that if we define $\lambda_1 = 1/4$, then we can set $\gamma_1 = 1/16$. If $d = 2$, then set $\lambda_2 = 1/8$. Indeed, let h be the moving subspace guaranteed by Lemma 13. By definition of F_h, there exist at least $n^2/16r^2 > \binom{n}{2}/8r^2 = |F_h|/8r^2$ points p of F_h such that $p(t) \in C \cap h(t)$. Since P_h is the projection of F_h onto x_2-axis, there exist $> |P_h|/8r^2$ points p' of P_h such that $p'(t)$ belongs to the projection of the segment $C \cap h(t)$ onto x_2-axis. Hence, since N_h is a strong $\frac{1}{8r^2}$-net for the kinetic hypergraph of P_h with respect to intervals, the projection of $C \cap h(t)$ onto x_2-axis must contain a point $v(t)$ of $N_h(t)$. By definition of N, the moving point $q = (x_{h,1}, v)$ is in N, so $C \cap N(t) \neq \emptyset$ and the case $d = 2$ follows, see Figure 2 for an illustration. Hence, one can assume that $d \geq 3$.

In higher dimensions the analysis requires more effort. We need the following lemma implicitly established by Chazelle et al. in [6]. For the sake of completeness, the technical proof is postponed to the end of this section.

▶ **Lemma 14** (Chazelle et al. [6]). *Let $d \geq 3$ and $P \subset \mathbb{R}^d$ be a set of n/r points such that any d points of P are affinely independent. Assume that we have an affine subspace h given by $x_1 = a_1, \ldots, x_j = a_j$ with $1 \leq j \leq d-2$, and a set $\mathcal{S}$ of at least $\alpha_j n^{j+1}/r^{j+1}$ $(j+1)$-tuples of P with $\alpha_j > 0$ such that the corresponding simplices intersect h exactly once. Then given $n \geq 4(j+1)r/\alpha_j$, there is an $\alpha_{j+1} > 0$ and an affine subspace $x_1 = a_1, \ldots, x_j = a_j, x_{j+1} = a_{j+1}$ intersecting at least $\alpha_{j+1} n^{j+2}/r^{j+2}$ $(j+1)$-simplices spanned by $(j+2)$-tuples*

from P. Moreover, each such $(j+2)$-tuple has the form $\{p_1,\dots,p_{j+1}\}\cup\{p_1,\dots p_j,q_1\}$ for $\{p_1,\dots p_{j+1}\}$, $\{p_1,\dots p_j,q_1\}\in\mathcal{S}$. Finally, $a_{j+1}\in[\{p_1,\dots p_{j+1}\},\{p_1,\dots p_j,q_1\}]$, where by abuse of notation $\{p_1,\dots p_{j+1}\}$ is the projection of the intersection point of the corresponding j-simplex with h onto x_{j+1}-axis.

Assume that we have defined λ_i, γ_i for $i\le j$, where $1\le j\le d-2$. Let h be a moving subspace of step j such that at least $\gamma_j n^{j+1}/r^{j+1}$ j-simplices spanned by $C\cap P(t)$ intersect $h(t)$ once in their relative interior. Let us assume that $n\ge 4(j+1)r/\gamma_j$. In what follows, we use the same notation as in the statement of Lemma 14. By this lemma (used with $\alpha_j=\gamma_j$, the affine subspace $h(t)$, and the set of points $C\cap P(t)$), we get a point a_{j+1} contained in at least $\alpha_{j+1}n^{j+2}/r^{j+2}$ intervals $[\{p_1(t),\dots p_{j+1}(t)\},\{p_1(t),\dots p_j(t),q_1(t)\}]$ as in the statement of Lemma 14. This is true, because we distinguish two intervals that do not arise from the same pair of $(j+1)$-tuples. We sometimes refer to the projection $\{p_1(t),\dots,p_{j+1}(t)\}$ as a vertex.

Set $J=\{x_{\tilde{h},j+1}(t):\tilde{h}$ is a moving subspace induced by $h\}$. We recall that $x_{\tilde{h},j+1}(t)$ is the $j+1$-th coordinate of $\tilde{h}(t)$. Let y_1 be the biggest $a\in J$ smaller or equal to a_{j+1} (if no such a exists, take $-\infty$). Similarly, let y_2 be the smallest $a\in J$ bigger or equal to a_{j+1}(if no such a exists, take ∞). The following lemma shows that by an appropriate choice of λ_{j+1}, not many intervals as above can lie strictly between y_1 and y_2.

▶ **Lemma 15.** *If $\lambda_{j+1}=2\alpha_{j+1}/3(j+1)$, then at most $\alpha_{j+1}n^{j+2}/3r^{j+2}$ intervals as above are contained in $]y_1,y_2[$ on x_{j+1}-axis.*

Proof. By contradiction, assume that $\ge\alpha_{j+1}n^{j+2}/3r^{j+2}$ intervals are contained in $]y_1,y_2[$. In what follows, we distinguish two vertices arising from different $(j+1)$-tuples. Counted with multiplicities, there are at least $2\alpha_{j+1}n^{j+2}/3r^{j+2}$ vertices $\{p_1(t),\dots,p_{j+1}(t)\}$ in $]y_1,y_2[$. Each vertex $\{p_1(t),\dots,p_{j+1}(t)\}$ is counted at most $(j+1)n/r$ times, since there are at most $j+1$ choices of $\{p_{i_1}(t),\dots,p_{i_j}(t)\}\subset\{p_1(t),\dots,p_{j+1}(t)\}$ and at most n/r choices for $q(t)$ so that $[\{p_1(t),\dots,p_{j+1}(t)\},\{p_{i_1}(t),\dots,p_{i_j}(t),q(t)\}]$ is an interval as above. Hence, there are at least $\ge 2\alpha_{j+1}n^{j+1}/3(j+1)r^{j+1}$ distinct vertices in $]y_1,y_2[$, a contradiction with the value of λ_{j+1}. To see this, we recall that each vertex $\{p_1(t),\dots,p_{j+1}(t)\}$ is the projection of $p^{h,\{p_1,\dots,p_{j+1}\}}(t)$ onto x_{j+1}-axis for $p^{h,\{p_1,\dots,p_{j+1}\}}\in F_h$. Since the number of vertices $\{p_1(t),\dots,p_{j+1}(t)\}$ in $]y_1,y_2[$ is at least $\ge 2\alpha_{j+1}n^{j+1}/3(j+1)r^{j+1}$, the number of $p^{h,\{p_1,\dots,p_{j+1}\}}\in F_h$ such that the projection of $p^{h,\{p_1,\dots,p_{j+1}\}}(t)$ onto x_{j+1}-axis is in $]y_1,y_2[$ is obviously also $\ge 2\alpha_{j+1}n^{j+1}/3(j+1)r^{j+1}$. Hence, by definition of P_h the number of $p\in P_h$ such that $p(t)\in]y_1,y_2[$ is at least

$$\frac{2\alpha_{j+1}n^{j+1}}{3(j+1)r^{j+1}}=\frac{\lambda_{j+1}n^{j+1}}{r^{j+1}}>\frac{\lambda_{j+1}\binom{n}{j+1}}{r^{j+1}}=\frac{\lambda_{j+1}|P_h|}{r^{j+1}}.$$

Thus, since N_h is a strong $\frac{\lambda_{j+1}}{r^{j+1}}$-net for the kinetic hypergraph of P_h with respect to intervals, there should be a point $w\in N_h$ such that $w(t)$ is in $]y_1,y_2[$. This means that there is a moving affine subspace induced by h whose x_{j+1}-coordinate at t $w(t)$ is strictly between y_1 and y_2, which contradicts the definition of y_1 or y_2. ◀

Let us set $\lambda_{j+1}=2\alpha_{j+1}/3(j+1)$. By the pigeonhole principle and the lemma above, y_1 or y_2 belongs to at least $\alpha_{j+1}n^{j+2}/3r^{j+2}$ intervals as above (say w.l.o.g. y_1). Let us denote by h_1 a moving subspace induced by h such that the x_{j+1}-coordinate of $h_1(t)$ is y_1. Thus, at least $\alpha_{j+1}n^{j+2}/3r^{j+2}$ $(j+1)$-simplices spanned by $C\cap P(t)$ intersect $h_1(t)$. One needs to be careful, since some of these simplices may intersect $h_1(t)$ more than once or not in their relative interior. However, assuming that $n\ge c_{\alpha_j/3}r$, where $c_{\alpha_j/3}$ is as in Lemma 16, one

can apply this lemma to conclude that at least $\alpha_{j+1}n^{j+2}/6r^{j+2}$ of them intersect $h_1(t)$ only once and in their relative interior. Hence, setting $\gamma_{j+1} = \alpha_{j+1}/6$ completes the induction.

Note that we still need to define λ_d. Let us set $\lambda_d = \gamma_{d-1}$. It remains us to see that the resulting N is a kinetic weak $\frac{1}{r}$-net for P. From the definition of $\gamma_{d-1} = \lambda_d$, we know that some affine subspace $h(t)$ where h is a moving space of step $d-1$, i.e., a moving line of step $d-1$, must intersect at least $\lambda_d n^d/r^d > \lambda_d\binom{n}{d}/r^d = \lambda_d|F_h|/r^d$ $(d-1)$-simplices spanned by $C \cap P(t)$ once in their relative interior. By definition of F_h, this implies that there exist $> \lambda_d|F_h|/r^d$ points p of F_h such that $p(t)$ belongs to the segment $C \cap h(t)$. Since P_h is the projection of F_h onto x_d-axis, there exist $> \lambda_d|P_h|/r^d$ points p' of P_h such that $p'(t)$ belongs to the projection of the segment $C \cap h(t)$ onto x_d-axis. Hence, since N_h is a strong $\frac{\lambda_d}{r^d}$-net for the kinetic hypergraph of P_h with respect to intervals, the projection of $C \cap h(t)$ onto x_d-axis must contain a point $v(t)$ of $N_h(t)$. By definition of N, the moving point $q = (x_{h,1}, \ldots, x_{h,d-1}, v)$ is in N and obviously belongs to C. Thus, N is a kinetic weak $\frac{1}{r}$-net for P, and the theorem follows. ◀

We now establish the remaining technical lemmas.

▶ **Lemma 16.** *Let $1 \leq j \leq d-1$ and $P \subset \mathbb{R}^d$ be a set of n/r points such that no $d+2$ of them lie in a hyperplane. Assume that we have a set $\mathcal{S}$ of $\alpha n^{j+1}/r^{j+1}$ $(j+1)$-tuples from P such that the convex hull of each of them intersects a given affine subspace V of dimension $d-j$. Then there exists c_α such that if $n \geq c_\alpha r$, then there are at least $\alpha n^{j+1}/2r^{j+1}$ $(j+1)$-tuples from $\mathcal{S}$ such that their convex hulls intersect V exactly once and in their relative interior.*

Proof. We can assume that $\alpha > 0$, otherwise there is nothing to show. Assume that the convex hulls of at least $\alpha n^{j+1}/2r^{j+1}$ $(j+1)$-tuples from $\mathcal{S}$ intersect the affine subspace V more than once or on their relative boundary. We will show that for $n \geq c_\alpha r$, where c_α is large enough, we obtain a contradiction. When the convex hull of a $(j+1)$-tuple A intersects V more than once, one can take two intersection points x_1 and x_2 with the affine subspace V and follow the line passing through x_1, x_2 until the relative boundary of $\mathrm{conv}(A)$ is intersected. Hence, since the line through x_1,x_2 is in V, in both cases the relative boundary of $\mathrm{conv}(A)$ must be intersected. Clearly, this means that there is a subset of j points from A whose convex hull intersects V. Each such j-tuple can be counted at most n/r times. Hence, there are at least $\alpha n^j/2r^j$ distinct j-tuples arising from elements of $\mathcal{S}$ as above.

We define $\mathcal{S}_j$ to be the set of j-tuples above, i.e., those whose convex hulls intersect V. Set $\gamma_j = \alpha/2$. If $j \geq 2$, then in order to obtain a contradiction we consider the following iterative procedure. Assume that $\mathcal{S}_i$ was defined for some $2 \leq i \leq j$ and contains at least $\gamma_i n^i/r^i$ i-tuples whose convex hulls intersect V. We say that $\mathcal{S}_i$ is *good* if it has a subset of at least $\gamma_i n^i/2r^i$ i-tuples, denoted by $\mathcal{G}_i$, such that the convex hull of no $(i-1)$-tuples which are $(i-1)$-subsets of the i-tuples from $\mathcal{G}_i$ intersects the affine space V. Otherwise, we say that the set $\mathcal{S}_i$ is *bad*, and define $\mathcal{S}_{i-1}$ to be the set of $(i-1)$-tuples whose convex hulls intersect V and each of them is contained in some i-tuple from $\mathcal{S}_i$. Clearly, the size of S_{i-1} is at least $\gamma_i n^{i-1}/2r^{i-1}$, since an $(i-1)$-tuple can appear in at most n/r i-tuples of S_i. Finally, we set $\gamma_{i-1} = \gamma_i/2$. For some i the procedure must stop with a good $\mathcal{S}_i$. Indeed, if we had to compute $\mathcal{S}_1$, then this means that we have a set of points from P of cardinality at least $\gamma_1 n/r$ such that each point belongs to V. This means that for n large enough ($n \geq (d+2)r/\gamma_1$), we get a set of at least $d+2$ points contained in V. That is, an affine subspace of dimension at most $d-1$, a contradiction.

Hence, we can assume that S_i is good for some $i \geq 2$. Let $\mathcal{G}_i$ be as above. Define a graph G whose vertices are the different $(i-1)$-tuples each contained in some i-tuple from $\mathcal{G}_i$. For

each i-tuple from $\mathcal{G}_i$ choose two different $(i-1)$ subsets and connect them by an edge. The number of edges is at least $\gamma_i n^i/2r^i$, since an edge determines the i-tuple it arises from.

Clearly, there is a vertex of degree at least $\gamma_i n^i/2r^i\binom{n/r}{i-1} \geq \gamma_i n/2r$. Take one such $(i-1)$-tuple $\{p_1,\ldots,p_{i-1}\}$. This means that the affine space given by $\text{aff}(V,p_1,\ldots,p_{i-1})$ of dimension at most $d-1$ contains at least $i-1+\gamma_i n/2r$ points, i.e., $p_1,\ldots,p_{i-1}$ and the points of the union of all neighbours of $\{p_1,\ldots,p_{i-1}\}$ in G. Indeed, let p be the intersection point of $\text{conv}(\{p_1,\ldots,p_i\}$ with V, where p_i belongs to some neighbour of $\{p_1,\ldots,p_{i-1}\}$ in G. We show that $\text{aff}(\{p_1,\ldots,p_{i-1},p\})$=$\text{aff}(\{p_1,\ldots,p_{i-1},p_i\})$. If p_i is in $\text{aff}(\{p_1,\ldots,p_{i-1}\})$, then the equality is clear. If not, then $\text{aff}(\{p_1,\ldots,p_{i-1},p\})$ has dimension strictly bigger than $\text{aff}(\{p_1,\ldots,p_{i-1}\})$ while being contained in $\text{aff}(\{p_1,\ldots,p_{i-1},p_i\})$, so the equality holds. Hence, for n large enough $(n \geq (d+1)2r/\gamma_i)$ we get a contradiction, since strictly more than $d+2$ points are in the affine subspace $\text{aff}(\{V,p_1,\ldots,p_{i-1}\})$ whose dimension is at most $d-1$, in particular, the points are contained in a hyperplane. ◀

▶ **Lemma 17.** *Let P be a set of points moving polynomially in $\mathbb{R}^d$ with bounded description complexity β. Let $\{p_1,\ldots,p_{j+1}\}$ be a $(j+1)$-tuple from P and h some moving affine subspace of step j, as defined in the proof of Theorem 12. Then one can define a moving point p such that for each $t \geq 0$ when the intersection of $\text{aff}(\{p_1(t),\ldots,p_{j+1}(t)\})$ and $h(t)$ is a single point, it is equal to $p(t)$. Moreover, p has description complexity $f(j+1)$, where $f:\{1,\ldots,d\}\to\mathbb{N}$ is some increasing function with $f(1)=\beta$.*

Proof. The case where for each $t \geq 0$ the intersection of $\text{aff}(\{p_1(t),\ldots,p_{j+1}(t)\})$ and $h(t)$ is empty or contains more than one point is trivial, since one can define p to be static.

Hence, one can assume that for some $t \geq 0$ the intersection above contains a single point. We prove the lemma by induction on the step. Observe that the function defining the first coordinate of a moving subspace of step i is obtained by projection of some point from P, hence has description complexity $\beta = f(1)$.

Asssume that the lemma holds for moving points arising from moving subspaces of step at most $j-1$, where $0 \leq j-1 \leq d-2$. Let $p_1,\ldots,p_{j+1}$ be any $(j+1)$-tuple of points from P and h any moving subspace of step j and given by $x_1 = x_{h,1},\ldots,x_j = x_{h,j}$. Then it follows from the definition of $x_{h,i}$ (see Theorem 12), the induction hypothesis, and the observation above that $x_{h,i}$ has description complexity $f(i)$. Assume $h(t)$ and $\text{aff}(\{p_1(t),\ldots,p_{j+1}(t)\})$ intersect in a unique point $p(t)$. Then we can write $p(t) = \alpha_1(t)p_1(t) + \ldots + \alpha_{j+1}(t)p_{j+1}(t)$ and from the general position assumption the points $p_1(t),\ldots,p_{j+1}(t)$ are affinely independent, so a point of $\text{aff}(\{p_1(t),\ldots p_{j+1}(t)\})$ is uniquely determined by an affine combination of the points $p_i(t)$. An immediate consequence from the unicity of $\alpha_i(t)$ is the following matricial equivalence:

$$\begin{pmatrix} [p_1(t)]_1 & \ldots & [p_{j+1}(t)]_1 \\ \vdots & & \vdots \\ [p_1(t)]_j & \ldots & [p_{j+1}(t)]_j \\ 1 & \ldots & 1 \end{pmatrix} \begin{pmatrix} \alpha_1(t) \\ \vdots \\ \vdots \\ \alpha_{j+1}(t) \end{pmatrix} = \begin{pmatrix} x_{h,1}(t) \\ \vdots \\ x_{h,j}(t) \\ 1 \end{pmatrix}$$

$$\iff$$

$$\begin{pmatrix} \alpha_1(t) \\ \vdots \\ \vdots \\ \alpha_{j+1}(t) \end{pmatrix} = \begin{pmatrix} [p_1(t)]_1 & \ldots & [p_{j+1}(t)]_1 \\ \vdots & & \vdots \\ [p_1(t)]_j & \ldots & [p_{j+1}(t)]_j \\ 1 & \ldots & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_{h,1}(t) \\ \vdots \\ x_{h,j}(t) \\ 1 \end{pmatrix}$$

It follows from the Cramer's rule that the moving point α_i, whose position at any $t \geq 0$ is $\alpha_i(t)$ given by the equation above, has description complexity depending only on j and $f(j)$. Hence, the moving point p whose position at t is $\alpha_1(t)p_1(t) + \ldots + \alpha_{j+1}(t)p_{j+1}(t)$ also has description complexity depending only on j and $f(j)$ that we denote by $f(j+1)$ (w.l.o.g. $f(j+1) \geq f(j)$). This completes the proof. ◀

Proof of Lemma 14. Define the hypergraph on P whose hyperedges are the different $(j+1)$-tuples of $\mathcal{S}$. Iteratively remove a j-tuple A from $\binom{n/r}{j}$ and remove the $(j+1)$-tuples containing it from $\mathcal{S}$ if the number of the remaining elements from $\mathcal{S}$ containing A is at most $\alpha_j n^{j+1}/2r^{j+1}\binom{n/r}{j}$. Call $\mathcal{S}'$ the remaining set of $(j+1)$-tuples. This procedure cannot remove more than $\alpha_j n^{j+1}/2r^{j+1}$ hyperedges, so the resulting hypergraph is not empty and each j-tuple contained in some element from $\mathcal{S}'$ is contained in

$$> \frac{\alpha_j n^{j+1}}{2r^{j+1}\binom{n/r}{j}} \geq \frac{\alpha_j n}{2r} = \frac{\alpha' n}{r}$$

elements from $\mathcal{S}'$, where we set $\alpha' = \alpha_j/2$.

We now project the intersections of simplices corresponding to $(j+1)$-tuples from $\mathcal{S}'$ with h onto the x_{j+1}-axis. For the sake of simplicity, the projection of the intersection point induced by the tuple $\{p_1, \ldots, p_{j+1}\}$ will still be denoted by $\{p_1, \ldots, p_{j+1}\}$. Two projections $\{p_1, \ldots, p_{j+1}\}$ and $\{q_1, \ldots, q_{j+1}\}$ give an interval of *type 1* if there is a sequence $\{p_1, \ldots, p_{j+1}\}$, $\{p_1, \ldots, p_j, q_1\}, \ldots, \{q_1, \ldots, q_{j+1}\}$, where each member of the sequence is an element of $\mathcal{S}'$ and the points $p_1, \ldots, p_{j+1}, q_1, \ldots, q_{j+1}$ are all distinct.

The following procedure gives a lower bound on the number of such intervals (we distinguish two intervals arising from different pairs of $(j+1)$-tuples): Choose any $\{p_1, \ldots, p_{j+1}\}$ in $\mathcal{S}'$. Take any q_1 such that $\{p_1, \ldots, p_j, q_1\}$ is in $\mathcal{S}'$ with q_1 different from p_{j+1}. Then take any q_2 such that q_2 is different from p_j, p_{j+1} and $\{p_1, \ldots, p_{j-1}, q_1, q_2\}$ is in $\mathcal{S}'$ etc. The lower bound below follows

$$\frac{|\mathcal{S}'|(\alpha' n/r - j - 1)^{j+1}}{2(j+1)!} \geq \frac{|\mathcal{S}'|(\alpha' n/2r)^{j+1}}{2(j+1)!}$$

given $\alpha' n/2r \geq j+1$. Indeed, starting from $\{p_1, \ldots, p_{j+1}\}$ an interval $[\{p_1, \ldots, p_{j+1}\}, \{q_1, \ldots, q_{j+1}\}]$ is counted at most once for each permutation of $q_1, \ldots, q_{j+1}$. Thus from the one dimensional selection lemma, see [2], we know that there exists a point a_{j+1} contained in at least

$$\frac{|\mathcal{S}'|^2[(\alpha' n/2r)^{j+1}/2(j+1)!]^2}{4|\mathcal{S}'|^2} = \frac{1}{4}\frac{[(\alpha' n/2r)^{j+1}]^2}{[2(j+1)!]^2} = \frac{\alpha'' n^{2j+2}}{r^{2j+2}}$$

intervals, where we set $\alpha'' = \alpha'^{2j+2}/2^{2j+6}[(j+1)!]^2$.

Clearly, if a point is contained in an interval $[\{p_1, \ldots, p_{j+1}\}, \{q_1, \ldots, q_{j+1}\}]$, it must also be contained in some interval $[\{p_1, \ldots p_s, q_1, \ldots q_{j-s+1}\}, \{p_1, \ldots p_{s-1}, q_1, \ldots q_{j-s+2}\}]$. This latter kind of intervals is refered to as *type 2*. Moreover, an interval of type 2 can be counted at most $(j+1)(jn/r)^j$ times. Indeed, there are at most $j+1$ possible positions for such an interval in a chain as above (used to define type 1 intervals), at most j possibilities of choosing a point that is replaced in a $(j+1)$-tuple while a subchain is extended, and at most n/r candidates to replace such a point. Hence, a_{j+1} is contained in at least $\alpha'' n^{2j+2}/r^{2j+2}(j+1)(jn/r)^j = \alpha''' n^{j+2}/r^{j+2}$ intervals of type 2, where $\alpha''' = \alpha''/(j+1)j^j$.

Each interval of type 2 containing the point a_{j+1} corresponds to a $(j+1)$-simplex spanned by P intersecting the affine subspace given by $x_1 = a_1, \ldots, x_{j+1} = a_{j+1}$. Finally, it is easy

to see that a spanned $(j+1)$-simplex arises from at most $(j+2)(j+1)$ intervals of type 2. Hence, there exist at least $\alpha'''n^{j+2}/(j+2)(j+1)r^{j+2}$ $(j+1)$-simplices arising from intervals of type 2 pierced by a_{j+1}, and the lemma follows. ◀

3 Open problems

This paper naturally leads to some questions. Can we restrict ourselves to points moving polynomially in order to find a kinetic net? More precisely:

▶ **Problem 1.** *Let $d \geq 2, \beta$ be integers and $r \geq 1$. Is there a pair $c(d,\beta,r)$, $g(d,\beta)$ such that for any finite set P of points moving polynomially with bounded description complexity β in $\mathbb{R}^d$ there exists a kinetic weak $\frac{1}{r}$-net for P of cardinality at most $c(d,\beta,r)$ and description complexity $g(d,\beta)$ whose points move polynomially?*

Let $d \geq 1, \beta$ be fixed integers and $c(d,\beta,r)$ be as in theorem 4. We didn't prove any lower bound on $c(d,\beta,r)$, so the current best lower bounds coincide with those in the static case which are $\Omega(r\log^{d-1} r)$, see [3]. This leads to the following research direction.

▶ **Problem 2.** *Close the gap between the lower and upper bounds on $c(d,\beta,r)$.*

References

1. N. Alon, I. Bárány, Z. Füredi, and D.J. Kleitman. Point selections and weak e-nets for convex hulls. *Combinatorics, Probability & Computing*, 1:189–200, 1992.
2. B. Aronov, B. Chazelle, and H. Edelsbrunner. Points and triangles in the plane and halving planes in space. *Discrete & Computational Geometry*, 6:435–442, 1991. doi:10.1007/BF02574700.
3. B. Bukh, J. Matoušek, and G. Nivasch. Lower bounds for weak epsilon-nets and stair-convexity. *Israel Journal of Mathematics*, 182(1):199–228, 2011.
4. J.L. De Carufel, M. Katz, M. Korman, A. van Renssen, M. Roeloffzen, and S. Smorodinsky. On kinetic range spaces and their applications. *CoRR*, abs/1507.02130, 2015. URL: http://arxiv.org/abs/1507.02130.
5. B. Chazelle, H. Edelsbrunner, M. Grigni, L.J. Guibas, M. Sharir, and E. Welzl. Improved bounds on weak epsilon-nets for convex sets. *Discrete & Computational Geometry*, 13:1–15, 1995.
6. B. Chazelle, H. Edelsbrunner, M. Grigni, L.J. Guibas, M. Sharir, and E. Welzl. Improved bounds on weak epsilon-nets for convex sets. *Discrete & Computational Geometry*, 13:1–15, 1995. doi:10.1007/BF02574025.
7. D. Haussler and E. Welzl. epsilon-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987. doi:10.1007/BF02187876.
8. J. Matoušek and U. Wagner. New constructions of weak epsilon-nets. *Discrete & Computational Geometry*, 32(2):195–206, 2004.
9. N.H. Mustafa and S. Ray. Weak epsilon-nets have basis of size o(1/epsilon log(1/epsilon)) in any dimension. *Comput. Geom.*, 40(1):84–91, 2008. doi:10.1016/j.comgeo.2007.02.006.
10. A. Ya. Chervonenkis V. N. Vapnik. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Applications of Incidence Bounds in Point Covering Problems*

Peyman Afshani[1], Edvin Berglin[2], Ingo van Duijn[3], and Jesper Sindahl Nielsen[4]

1 **MADALGO, Department of Computer Science, Aarhus University, Denmark**
peyman@cs.au.dk
2 **MADALGO, Department of Computer Science, Aarhus University, Denmark**
berglin@cs.au.dk
3 **MADALGO, Department of Computer Science, Aarhus University, Denmark**
ivd@cs.au.dk
4 **MADALGO, Department of Computer Science, Aarhus University, Denmark**
jasn@cs.au.dk

Abstract

In the *Line Cover* problem a set of n points is given and the task is to cover the points using either the minimum number of lines or at most k lines. In *Curve Cover*, a generalization of *Line Cover*, the task is to cover the points using curves with d degrees of freedom. Another generalization is the *Hyperplane Cover* problem where points in d-dimensional space are to be covered by hyperplanes. All these problems have kernels of polynomial size, where the parameter is the minimum number of lines, curves, or hyperplanes needed.

First we give a non-parameterized algorithm for both problems in $\mathcal{O}^*(2^n)$ (where the $\mathcal{O}^*(\cdot)$ notation hides polynomial factors of n) time and polynomial space, beating a previous exponential-space result. Combining this with incidence bounds similar to the famous Szemerédi-Trotter bound, we present a *Curve Cover* algorithm with running time $\mathcal{O}^*\left((Ck/\log k)^{(d-1)k}\right)$, where C is some constant. Our result improves the previous best times $\mathcal{O}^*\left((k/1.35)^k\right)$ for *Line Cover* (where $d = 2$), $\mathcal{O}^*\left(k^{dk}\right)$ for general *Curve Cover*, as well as a few other bounds for covering points by parabolas or conics. We also present an algorithm for *Hyperplane Cover* in $\mathbb{R}^3$ with running time $\mathcal{O}^*\left((Ck^2/\log^{1/5} k)^k\right)$, improving on the previous time of $\mathcal{O}^*\left((k^2/1.3)^k\right)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Point Cover, Incidence Bounds, Inclusion Exclusion, Exponential Algorithm

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.60

1 Introduction

In the *Line Cover* problem a set of points in $\mathbb{R}^2$ is given and the task is to cover them using either the minimum number of lines, or at most k lines where k is given as a parameter in the input. It is related to *Minimum Bend Euclidean TSP* and has been studied in connection with facility location problems [8, 17]. The *Line Cover* problem is one of the few low-dimensional geometric problems that are known to be NP-complete [17]. Furthermore *Line Cover* is APX-hard, i.e., it is NP-hard to approximate within factor $(1+\varepsilon)$ for arbitrarily

* Research funded by MADALGO, Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84

© Peyman Afshani, Edvin Berglin, Ingo van Duijn, and Jesper Sindahl Nielsen;
licensed under Creative Commons License CC-BY

small ε [15]. Although NP-hard, *Line Cover* is fixed-parameter tractable when parameterized by its solution size k so any solution that is "not too large" can be found quickly.

One generalization of the *Line Cover* problem is the *Hyperplane Cover* problem, where the task is to use the minimum number of hyperplanes to cover points in d-dimensional space. Another generalization is to cover points with algebraic curves, e.g. circles, ellipses, parabolas, or bounded degree polynomials. These can be categorized as covering points in an arbitrary dimension space using algebraic curves with d degrees of freedom and at most s pairwise intersections. We call this problem *Curve Cover*. The first parameterized algorithm that was presented for *Line Cover* runs in time $\mathcal{O}^*\big(k^{2k}\big)$ [16] (where $\mathcal{O}^*(.)$ hides polynomial factors). This algorithm generalizes to generic settings, such as *Curve Cover* and *Hyperplane Cover*, obtaining the running time $\mathcal{O}^*\big(k^{dk}\big)$ where d is the degree of the freedom of the curves or the dimension of the space for hyperplane cover.

The first improvement to the aforementioned generic algorithm reduced the running time to $\mathcal{O}^*\big((k/2.2)^{dk}\big)$ for the *Line Cover* problem [10]. The best algorithm for the *Hyperplane Cover* problem, including *Line Cover*, runs in $\mathcal{O}^*\big(k^{(d-1)k}/1.3^k\big)$ time [22]. A non-parameterized solution to *Line Cover* using dynamic programming has been proposed with both time and space $\mathcal{O}^*(2^n)$ [4], which is time efficient when the number of points is $\mathcal{O}(k \log k)$. Algorithms for parabola cover and conic cover appear in [21], running in time $\mathcal{O}^*\big((k/1.38)^{(d-1)k}\big)$ and $\mathcal{O}^*\big((k/1.15)^{(d-1)k}\big)$ respectively.

Incidence Bounds. Given an arrangement of n points and m lines, an *incidence* is a point-line pair where the point lies on the line. Szemerédi and Trotter gave an asymptotic (tight) upper bound of $\mathcal{O}\big((nm)^{2/3} + n + m\big)$ on the number of incidences in their seminal paper [20]. This has inspired a long list of similar upper bounds for incidences between points and several types of varieties in different spaces, e.g. [7, 9, 18, 19].

Our Results. We give a non-parameterized algorithm solving the decision versions of both *Curve Cover* and *Hyperplane Cover* in $\mathcal{O}^*(2^n)$ time and polynomial space. Furthermore we present parameterized algorithms for *Curve Cover* and *Plane Cover* (*Hyperplane Cover* in $\mathbb{R}^3$). These solve *Curve Cover* in time $\mathcal{O}^*\big((Ck/\log k)^{(d-1)k}\big)$ and *Plane Cover* in time $\mathcal{O}^*\Big((Ck^2/\log^{1/5} k)^k\Big)$, both using polynomial space. The main idea is to use Szemerédi-Trotter-type incidence bounds and using the aforementioned $\mathcal{O}^*(2^n)$ algorithm as a base case. We make heavy use of (specialized) incidence bounds and our running time is very sensitive to the maximum number of possible incidences between points and curves or hyperplanes. In general, utilization of incidence bounds for constructing algorithms is rare (see e.g. [13, 12]) and to our knowledge we are the first to do so for this type of covering problem. It is generally believed that point sets that create large number of incidences must have some "algebraic sub-structure" (see e.g. [11]) but curiously, the situation is not fully understood even in two dimensions. So, it might be possible to get better specialized incidence bounds for us in the context of covering points. Thus, we hope that this work can give further motivation to study specialized incidence bounds. This is a shortened conference version of the paper. Omitted proofs and pseudocode can be found in the full version [1].

2 Preliminaries

2.1 Definitions

We begin by briefly explaining the concept of fixed-parameter tractability before formally stating the *Curve Cover* and *Hyperplane Cover* problems.

▶ **Definition 1.** A problem is said to be *fixed-parameter tractable* if there is a parameter k to an instance I, such that I can be decided by an algorithm in time $\mathcal{O}(f(k)poly(|I|))$ for some computable function f.

The function f is allowed to be any computable function, but for NP-complete problems can be expected to be at least single exponential. The name refers to the fact that these algorithms run in polynomial time when k is (bounded by) a constant. Within this paper I will typically be a set of points and k is always a solution budget: the maximum allowed size of any solution of covering objects, but not necessarily the size of the optimal such solution.

Let P be a set of n points in any dimension, and d, s be non-negative integers.

▶ **Definition 2.** A set of algebraic curves $\mathcal{C}$ are called (d, s)-curves if (i) any pair of curves from $\mathcal{C}$ intersect in at most s points and (ii) for any d points there are at most s curves in $\mathcal{C}$ through them. The parameter d is the *degrees of freedom* and s is the *multiplicity-type*.

The set $\mathcal{C}$ could be an infinite set corresponding to a family of curves, and it is often defined implicitly. We assume two geometric predicates: First, we assume that given two curves $c_1, c_2 \in \mathcal{C}$, we can find their intersecting points in polynomial time. Second, we assume that given any set of up to $s + 1$ points, in polynomial time, we can find a curve that passes through the points or decide that no such curve exists. These two predicates are satisfied in the real RAM model of computation for many families of algebraic curves and can be approximated reasonably well in practice.

We say that a curve *covers* a point, or that a point is *covered* by a curve, if the point lies on the curve. A set of curves $H \subset \mathcal{C}$ *covers* a set of points P if every point in P is covered by a curve in H, furthermore, H is a k-cover if $|H| \leq k$.

▶ **Definition 3** (Curve Cover Problem)**.** Given a family of (d, s)-curves $\mathcal{C}$, a set of points P, and an integer k, does there exist a subset of $\mathcal{C}$ that is a k-cover of P?

Now let P be a set of points in $\mathbb{R}^d$. A hyperplane covers a point if the point lies on the hyperplane. A set H of hyperplanes covers a set of points if every point is covered by some hyperplane; H is a k-cover if $|H| \leq k$. In $\mathbb{R}^d$, a j-flat is a j-dimensional affine subset of the space, e.g., 0-flats are points, 1-flats are lines and $(d-1)$-flats are called hyperplanes.

▶ **Definition 4** (Hyperplane Cover Problem)**.** Given an integer k and a set P of points in $\mathbb{R}^d$, does there exist a set of hyperplanes that is a k-cover of P?

For $d = 3$ we call the problem *Plane Cover*. To make our parameterized *Plane Cover* algorithm work, we need to introduce a third generalization: a version of *Hyperplane Cover* where the input contains any type of flats. A hyperplane covers a j-flat for $j \leq d-2$ if the flat lies on the hyperplane; further notation follows naturally from the above.

▶ **Definition 5** (Any-flat Hyperplane Cover Problem)**.** For $k \in \mathbb{N}$ and a tuple $P = \langle P_0, \ldots, P_{d-2} \rangle$, where P_i is a set of i-flats in $\mathbb{R}^d$, does there exist a set of hyperplanes that is a k-cover of P?

We stress that our non-parameterized algorithm in Section 3 solves *Any-flat Hyperplane Cover* while the parameterized algorithm in Section 5 solves *Plane Cover*. *Line Cover* is a special case of both *Curve Cover* and *Hyperplane Cover*. Since *Line Cover* is known to be both NP-hard [17] and APX-hard [15], the same applies to its three generalizations as well.

2.2 Kernels

Central to parameterized complexity theory is the concept of *polynomial kernels*. A parameterized problem has a polynomial kernel if an instance $\langle P, k \rangle$ in polynomial time can

be reduced to an instance $\langle P', k' \rangle$ where $|P'|$ and k' are bounded by $k^{\mathcal{O}(1)}$ and $\langle P, k \rangle$ is a yes-instance if and only if $\langle P', k' \rangle$ is a yes-instance. Problems with polynomial kernels are immediately fixed-parameter tractable: simply use brute force on reduced instance.

▶ **Lemma 6.** *For a family $\mathcal{C}$ of (d, s)-curves, Curve Cover has a size sk^2 kernel where no curve in $\mathcal{C}$ covers more than sk points.*

Proof. See the full version of the paper. ◀

For *Any-flat Hyperplane Cover* a size k^d kernel is presented in [16]. It uses a *grouping* operation, removing points and replacing them with higher dimension flats, which is not acceptable for a *Hyperplane Cover* input. We present an alternative, slightly weaker hyperplane kernel containing only points; in $\mathbb{R}^3$ it contains at most $k^3 + k^2$ points.

▶ **Lemma 7.** *Hyperplane Cover in $\mathbb{R}^d, d \geq 2$, has a size $k^2(\sum_{i=0}^{d-2} k^i) = \mathcal{O}\big(k^d\big)$ kernel where for $j \leq d-2$ any j-flat covers at most $\sum_{i=0}^{j} k^i = \mathcal{O}\big(k^j\big)$ points and any hyperplane covers at most $k \sum_{i=0}^{d-2} = \mathcal{O}\big(k^{d-2}\big)$ points.*

Proof. See the full version of the paper. ◀

Our algorithms will use both properties of the kernels. Kratsch et al. [14] showed that these kernels are essentially tight under standard assumptions in computational complexity.

▶ **Theorem 8** (Kratsch et al. [14])**.** *Line Cover has no kernel of size $\mathcal{O}\big(k^{2-\varepsilon}\big)$ unless coNP $\subseteq$ NP/poly.*

2.3 Incidence bounds

Consider the *Line Cover* problem. Obviously, if the input of n points are in general position, then we need $n/2$ lines to cover them. Thus, if $k \ll \frac{n}{2}$, we expect the points to contain "some structure" if they are to be covered by k lines. Such "structures" are very relevant to the study of incidences. For a set P of points and a set L of lines, the classical Szemerédi-Trotter [20] theorem gives an upper bound on the number of point-line incidences, $I(L, P)$, in $\mathbb{R}^2$.

▶ **Theorem 9** (Szemerédi and Trotter [20])**.** *For a set P of n points and a set L of m lines in the plane, let $I(L, P) = |\{(p, \ell) \mid p \in P \cap \ell, \ell \in L\}|$. Then $I(L, P) = \mathcal{O}\big((nm)^{2/3} + n + m\big)$.*

The linear terms in the theorem arise from the cases when there are very few lines compared to points (or vice versa). In the setting of *Line Cover* these cases are not interesting since they are easy to solve. The remaining term is therefore the interesting one. Since it it large, it implies there are many ways of placing a line such that it covers many points; this demonstrates the importance of incidence bounds for covering problems. We introduce specific incidence bounds for curves and hyperplanes in their relevant sections.

3 Inclusion-exclusion algorithm

This section outlines an algorithm Inclusion-Exclusion that for both problems decides the size of the minimum cover, or the existence of a k-cover, of a point set P in $\mathcal{O}^*(2^n)$ time and polynomial space. Our algorithm improves over the one from [4] for *Line Cover* which finds the cardinality of the smallest cover of P with the same time bound but exponential space. The technique is an adaptation of the one presented in [3]; their paper immediately gives either $\mathcal{O}^*(3^n)$-time polynomial-space or $\mathcal{O}^*(2^n)$-time $\mathcal{O}^*(2^n)$-space algorithms for our problems. We give full details of the technique for completeness; to do so, we require the intersection version of the inclusion-exclusion principle.

▶ **Theorem 10** (Folklore). *Let $A_1, \ldots, A_n$ be a number of subsets of a universe $\mathcal{U}$. Using the notation that $\overline{A} = \mathcal{U} \setminus A$ and $\bigcap_{i \in \emptyset} \overline{A_i} = \mathcal{U}$, we have:*

$$\left| \bigcap_{i \in \{1,\ldots,n\}} A_i \right| = \sum_{X \subseteq \{1,\ldots,n\}} (-1)^{|X|} \left| \bigcap_{i \in X} \overline{A_i} \right|.$$

3.1 Curve Cover

Let P be the input set of points and $\mathcal{C}$ be the family of (d, s)-curves under consideration. Although we are creating a non-parameterized algorithm, we nevertheless assume that we have access to the solution parameter k. This assumption will be removed later. We say a set Q is a *coverable set in* P (or is *coverable* in P) if $Q \subseteq P$ and Q has a 1-cover.

Let a *tuple* (in P) be a k-tuple $\langle Q_1, \ldots, Q_k \rangle$ such that $\forall i : Q_i$ is coverable in P. Note that there is no restriction on pairwise intersection between two coverable sets in a tuple. Define $\mathcal{U}$ as the set of all tuples. For $p \in P$, let $A_p = \{\langle Q_1, \ldots, Q_k \rangle \mid p \in \bigcup_i Q_i\} \subseteq \mathcal{U}$ be the set of all tuples where at least one coverable set contains p.

▶ **Lemma 11.** *P has a k-cover if and only if $\left|\bigcap_{p \in P} A_p\right| \geq 1$.*

Proof. Take a tuple in $\bigcap_{p \in P} A_p$. For each coverable set Q in the tuple, place a curve that covers Q. Since the tuple was in the intersection, every point is in some coverable set, so every point is covered by a placed curve. Hence we have a k-cover.

Take a k-cover $\mathcal{C}$ and from each curve $c \in \mathcal{C}$ construct a coverable set of the points covered by c. Form a tuple out of these sets and observe that the tuple is in the intersection $\bigcap_{p \in P} A_p$, hence its cardinality is at least 1. ◀

Note that several tuples may correspond to the same k-cover, so this technique cannot be used for the counting version of the problem. Theorem 10 and Lemma 11 reduce the problem of deciding the existence of k-covers to computing a quantity $\left|\bigcap_{i \in X} \overline{A_i}\right|$. The key observation is that $\overline{A_p}$ is the set of tuples where no coverable set contains p and $\bigcap_{i \in X} \overline{A_i}$ is the set of tuples that contain no point in X, i.e. the set of tuples in $P \setminus X$. The remainder of this section shows how to compute the size of this set in polynomial time. Let $c(X) = |\{Q \mid Q \subseteq X, Q \text{ is coverable in } X\}|$ be the number of coverable sets in a point set X. A tuple in $P \setminus X$ is k coverable sets drawn from a size $c(P \setminus X)$ pool (with replacement), hence there are $c(P \setminus X)^k$ such tuples. To compute $c(X)$ we introduce the notion of *representatives*. Let π be an arbitrary ordering of P. The representative $R = \{r_1, \ldots, r_i\}$ of a coverable set Q is the $\min(|Q|, s+1)$ first points in Q as determined by the order π. Note that for any coverable set Q, it holds that $R \subseteq Q$. Let $q(X, \pi, R)$ be the number of coverable sets that have the representative R.

▶ **Lemma 12.** *$q(X, \pi, R)$ can be computed in $\mathcal{O}(|X|)$ time and $\mathcal{O}(\log |X|)$ space.*

Proof. If R is not a valid representative, $q(X, \pi, R) = 0$. If $|R| \leq s$, $q(X, \pi, R) = 1$. If $|R| = s+1$, let U be the union of every coverable set with representative R, and $X' = U \setminus R$. The number of subsets of X' is the number of coverable sets with representative R, i.e. $q(X, \pi, R) = 2^{|X'|}$. For any $p \in P$ with $\pi(p) > \pi(r_i)$, $p \in X'$ if and only if there is a curve $c \in \mathcal{C}$ such that c covers $\{r_1, \ldots, r_i, p\}$. Since $i \leq s+1$ the time complexity is $\mathcal{O}(|X|)$. The space complexity is logarithmic since we need only maintain $|X'|$ rather than X'. ◀

▶ **Lemma 13.** *$c(X)$ can be computed in $\mathcal{O}\big(|X|^{s+2}\big)$ time and $\mathcal{O}(|X|)$ space.*

Proof. Fix an ordering π. As every coverable set in X has exactly one representative under π, we get that $c(X) = \sum_R q(X, \pi, R)$. There there are only $\mathcal{O}\left(\binom{|X|}{s+1}\right) = \mathcal{O}\left(|X|^{s+1}\right)$ choices of R for which $q(X, \pi, R) > 0$, and by Lemma 12 each term of the sum is computable in $\mathcal{O}(|X|)$ time and logarithmic space. The space complexity is therefore dominated by the space to store π which is linear. ◀

▶ **Theorem 14.** *There exists a k-cover of curves from $\mathcal{C}$ for P if and only if*

$$\left|\bigcap_{p\in P} A_p\right| = \sum_{X\subseteq P} (-1)^{|X|} \left|\bigcap_{p\in X} \overline{A_p}\right| = \sum_{X\subseteq P} (-1)^{|X|} c(P \setminus X)^k \geq 1.$$

This comparison can be performed in $\mathcal{O}\left(2^n n^{s+2}\right)$ time and $\mathcal{O}(nk)$ bits of space.

Proof. Since $c(P \setminus X)^k \leq 2^{nk}$ for any X it can be stored in nk bits. The absolute value of the partial sum can be kept smaller than 2^{nk} by choosing an appropriate next X. The rest follows from Theorem 10, Lemma 11 and Lemma 13. ◀

Finally, we remove the assumption that we have the parameter k. Any input requires at most n curves. Since k is only used to compute $c(X)^k$ we can try $k = 1, 2, \ldots, n$ and return the first k with a positive sum. This increases the time by an $\mathcal{O}(n)$ factor. Alternatively, we can run n simultaneous sums, since the parameter k is only accessed when computing $c(X)^k$. This increases the space by factor $\mathcal{O}(n)$ and the time by a lower-order additive term.

3.2 Any-flat Hyperplane Cover

Here we treat all flats in the instance $\langle P_0, \ldots, P_{d-2}\rangle$ as atomic objects and P as a union $\bigcup_{i=0}^{d-2} P_i$. This algorithm is very similar to that of Section 3.1, so we only describe their differences. A set of flats $Q \subseteq P$ is a coverable set in P if there exists a hyperplane that covers every $p \in Q$. The representative of $\emptyset$ is $\emptyset$, and the representative of a non-empty coverable set Q is a set $R = \{r_1, \ldots, r_i\}$. Let r_1 be the first flat in Q and for $j \geq 2$, r_j is defined if the affine hull of $\{r_1, \ldots, r_{j-1}\}$ has lower dimension than the affine hull of Q. If so, let r_j be the first flat in Q that is not covered by the affine hull of $\{r_1, \ldots, r_{j-1}\}$.

▶ **Lemma 15.** *$q(X, \pi, R)$ can be computed in $\mathcal{O}(|X|)$ time and $\mathcal{O}(\log |X|)$ space.*

Proof. If R is not a valid representative, $q(X, \pi, R) = 0$. Otherwise, let U be the union all coverable sets with the representative R, and $X' = U \setminus S$. For every $p \in X \setminus R$, let j be the highest index such that $\pi(r_j) < \pi(p)$. Then $p \in X'$ if and only if p is on the affine hull of $\{r_1, \ldots, r_j\}$. ◀

There are $\mathcal{O}\left(\binom{n}{d}\right)$ representatives R with $q(X, \pi, R) > 0$ so the following two results hold; their proofs are analogous to Lemma 13 and Theorem 14.

▶ **Lemma 16.** *$c(X)$ may be computed in $\mathcal{O}\left(|X|^{d+1}\right)$ time and $\mathcal{O}(|X|)$ space.*

▶ **Theorem 17.** *There exists a hyperplane k-cover for P if and only if*

$$\left|\bigcap_{p\in P} A_p\right| = \sum_{X\subseteq P} (-1)^{|X|} \left|\bigcap_{p\in X} \overline{A_p}\right| = \sum_{X\subseteq P} (-1)^{|X|} c(P \setminus X)^k \geq 1.$$

This comparison may be performed in $\mathcal{O}\left(2^n n^{d+1}\right)$ time and $\mathcal{O}(nk)$ bits of space.

4 Curve Cover

Recall that we are considering (d, s)-curves, where d and s are constants. Since we have a kernel of up to sk^2 points, Inclusion-Exclusion used on its own runs in time $\mathcal{O}^*\left(2^{sk^2}\right)$ which is too slow to give an improvement. We improve this by first using a technique that reduces the number of points in the input, and then using Inclusion-Exclusion. To describe this technique and the intuition behind it, we first provide a framework based on the following theorem by Pach and Sharir.

▶ **Theorem 18** (Pach and Sharir [18]). *Let P be a set of n points and L a set of m (d, s)-curves in the plane. The number of point-curve incidences between P and L is*

$$I(P, L) = \mathcal{O}\left(n^{d/(2d-1)}m^{(2d-2)/(2d-1)} + n + m\right).$$

Note that the above holds for curves in arbitrary dimension. This can be seen by projecting the points and curves onto a random plane, which will keep the projection of distinct points, and prevent the curves from projecting to overlapping curves.

▶ **Definition 19.** Let a *candidate* be any curve in $\mathcal{C}$ that covers at least 1 point in P. Define its *richness* with respect to P as the number of points it covers. A candidate is γ-rich if its richness is at least γ, and γ-poor if its richness is at most γ.

Recall that from the kernel in Lemma 6, every candidate is sk-poor. The following gives a bound on the number of γ-rich candidates and is an immediate consequence of Theorem 18.

▶ **Lemma 20** (Folklore). *Let P be a set of n points in some finite dimension space $\mathbb{R}^x$. The number of γ-rich candidates in P is $\mathcal{O}\left(\frac{n^d}{\gamma^{2d-1}} + \frac{n}{\gamma}\right)$.*

Intuition for algorithm. We exploit the following observation: given a k-cover $\mathcal{C}$, some curves in $\mathcal{C}$ might be significantly richer than others. The main idea of our technique is to try to select (i.e. branch on) these rich curves first. Since they cover "many" points, removing these decreases the ratio $|P|/k$ and calling Inclusion-Exclusion eventually becomes viable. The idea to branch on rich curves first has another important consequence. Suppose we know that no candidate in $\mathcal{C}$ covers more than γ points in P. This immediately implies that if there are strictly more than $k\gamma$ points in P, it is impossible to cover P. Therefore we have $|P|/k \leq \gamma$. Now look at the set of $\frac{\gamma}{2}$-rich candidates and decide for each whether to include it in the cover or not. By the earlier observation, including such a candidate is good for reducing the ratio $|P|/k$. But excluding such a candidate has essentially the same effect, because that candidate will not be considered again (remove it from $\mathcal{C}$). Any remaining candidates in $\mathcal{C}$ now cover at most $\frac{\gamma}{2}$ points; we must have $|P|/k \leq \frac{\gamma}{2}$ (or the instance is not solvable) and have strengthened the bound on the ratio. Regardless of which choice we make, we make progress towards being able to call the base case.

This strategy also makes sense from a combinatorial point of view, because from Lemma 20 it follows that the search space is small for rich curves. Switching to Inclusion-Exclusion early enough lets us bypass the potentially very large search space of poor candidates.

The Algorithm. Let r be a parameter. The exact value is set in the proof of Theorem 25, for now it is enough that $r = \Theta(\log k)$. For a budget k let $\langle k_1, \ldots, k_r \rangle$ with $\sum_j k_j = k$ be a *budget partition*. We describe a main recursive algorithm CC-Recursive (see full paper for pseudocode) that takes 4 arguments: the point set P, the class of curves $\mathcal{C}$, a budget partition

$\langle k_1, \ldots, k_r \rangle$, and a recursion level i. For convenience we define $\gamma_i = sk/2^i$. A simple top-level procedure CurveCover tries all budget partitions and calls the recursive algorithm with that partition at recursion level 1.

At every recursion depth i, let $K_i = \sum_{j=i}^{r} k_j$ be the remaining budget and P_i the remaining point set. That means earlier levels have created a partial solution $\mathcal{C}_{i-1}$ of $k - K_i$ curves covering the points $P \setminus P_i$. The recursive algorithm will try to cover the remaining points using γ_{i-1}-poor curves. Specifically, at depth i let S be the set of candidates from $\mathcal{C}$ that are γ_i-rich and γ_{i-1}-poor. Since from depth i and onward it has a remaining budget of K_i and cannot pick candidates that are $(\gamma_{i-1}+1)$-rich, the algorithm rejects if strictly more than $K_i\gamma_{i-1}$ remain. If fewer than $\frac{(d-1)}{2} K_i \log k$ points remain, the sub problem is solved with inclusion-exclusion.

If neither a reject (due to too many points) or a base-case call to inclusion-exclusion has occurred, the algorithm will branch. It does so in $\binom{|S|}{k_i}$ ways by simply trying all ways of choosing k_i candidates from S. For each such choice, all points in P covered by the chosen candidates are removed and the algorithm recurses to depth $i+1$. If all those branches fail, the instance is rejected.

4.1 Analysis

▶ **Lemma 21.** *Algorithm* CurveCover *decides whether P has a k-cover of curves from $\mathcal{C}$.*

Proof. Regard CurveCover as being non-deterministic. Suppose P has a k-cover $\mathcal{C}$. The proof is by induction on the recursion. Assume as the induction hypothesis that the current partial solution $\mathcal{C}_{i-1}$ is a subset of $\mathcal{C}$ and that $\mathcal{C}$ contains no curves that are $(\gamma_{i-1}+1)$-rich when restricted to P_i. The assumption is trivially true for $i = 1$ as $\mathcal{C}_0 = \emptyset$.

By the induction hypothesis, $\mathcal{C} \setminus \mathcal{C}_{i-1}$ is a K_i-cover for P_i using only γ_{i-1}-poor curves. Therefore $|P_i| \leq \gamma_{i-1}K_i$ and the algorithm does not reject incorrectly. Furthermore, if Inclusion-Exclusion is called it accepts since we are in the case that a solution exists.

Otherwise, let $D \subseteq \mathcal{C} \setminus \mathcal{C}_{i-1}$ be the curves that are γ_i-rich when restricted to P_i. The algorithm non-deterministically picks D from the set of candidates S and constructs $\mathcal{C}_i = \mathcal{C}_{i-1} \cup D$. This leaves $\mathcal{C}_i$ to be a subset of $\mathcal{C}$. Additionally, $\mathcal{C}_i$ contains all γ_i-rich curves in $\mathcal{C}$ restricted to P_i and hence to $P_{i+1} \subseteq P_i$, upholding the induction hypothesis.

Suppose the algorithm accepts the instance $\langle P, k \rangle$. It can only accept if some call to Inclusion-Exclusion accepts. Let $\mathcal{C}_r$ be the set of curves selected by the recursive part such that Inclusion-Exclusion accepted the instance $\langle P \setminus \mathcal{C}_r, k - |\mathcal{C}_r| \rangle$. Let $\mathcal{C}_{ie}$ be any $(k - |\mathcal{C}_r|)$-cover of $P \setminus \mathcal{C}_r$. Then $\mathcal{C}_r \cup \mathcal{C}_{ie}$ is a k-cover of P. ◀

By the nature of the inclusion-exclusion algorithm, CurveCover detects the existence of a k-cover rather than producing one. But since CC-Recursive produces a partial cover during its execution, it is straight-forward to extend that into a full k-cover by using Inclusion-Exclusion as an oracle.

Running time. To analyze the running time of the algorithm we see the execution of CC-Recursive as a search tree $\mathcal{T}$. Each leaf of the tree is either an immediate reject or a call to Inclusion-Exclusion. Since the latter is obviously most costly to run, we must assume for a worst case analysis that every leaf node calls the base case algorithm. The running time is the number of leaf nodes in the search tree times the running time of Inclusion-Exclusion. Since the algorithm performs exponential work in these leaf nodes but not in inner nodes, it is insufficient to reason about the *size* of the tree. Therefore we will speak of the "running time of a subtree", which simply means the running time of the

recursive call that corresponds to the root of that subtree. We show that in the worst case, $\mathcal{T}$ is a complete tree $\mathcal{T}_1$ of depth r. That is, $\mathcal{T}_1$ has no leaf nodes at depths less than r.

Let $\mathcal{T}_j$ be a complete subtree of $\mathcal{T}_1$ rooted at depth j. To prove that $\mathcal{T}_1$ is the worst case for $\mathcal{T}$ we prove two things. First we first prove an upper bound on the running time for arbitrary $\mathcal{T}_j$. Then we prove that the running time of $\mathcal{T}_1$ can only improve if an arbitrary subtree is replaced by a leaf (i.e. a call to Inclusion-Exclusion). The most involved part is proving an upper bound on the number of leaves of $\mathcal{T}_j$.

▶ **Lemma 22.** *Let L be the number of leaves in $\mathcal{T}_j$. Then for some constant $c_2 = c_2(d, s)$, L is bounded by*

$$L \leq \left(\frac{c_2 k^d}{(k-k_r)\log^{d-1} k}\right)^{K_j - k_r}.$$

The proof is long and tedious and left for the appendix of the full version of the paper. To give an idea of how Lemma 22 is proved, we sketch a simplified worst case analysis for *Line Cover*. The analysis can be generalized to *Curve Cover* and gives (up to a constant in the base of the exponent) the same running time as the real worst case.

Analysis sketch. The branching of $\mathcal{T}_1$ at recursion level i depends on the budget k_i that is being used. That means that the structure of the whole tree depends on the complete budget partition. From Lemma 20 it follows that the lower the richness the more candidates there are. Since the richness halves after every recursive call, one could conjecture that the worst case budget partition would put as much budget in the end. It could e.g. look like $\langle 0, 0, \ldots, 0, k_{r-1}, k_r\rangle$, where $k - k_r = k_{r-1} > k_r$. That is, only in the penultimate and last recursion level is there any budget to spend. At the deepest level of recursion, the richness considered is strictly less than $\frac{\log k}{2}$ (because with this richness the base case algorithm is efficient). Therefore, at the penultimate recursion level the richness is $\log k$. At this level there are $k \log k$ points left and we can apply Lemma 20 to bound the number of $\log k$ rich lines. This yields a bound of $\frac{k^2}{\log k}$ on the number of candidates. From these we pick $k - k_r$ lines, giving a branching of roughly $\left(\frac{k^2}{(k-k_r)\log k}\right)^{k-k_r}$ (where roughly means up to a constant in the base of the exponent).

It turns out that the worst case budget partition is in fact $\langle k_0 2^1, k_0 2^2, ..., k_0 2^{r-1}, k_r\rangle$ for some k_0. However, to understand where the division by $\log^{d-1} k$ comes from in the expression of Lemma 22, it is sufficient to understand the above analysis sketch. With Lemma 22 in place, we can prove the following bound on the running time of $\mathcal{T}_j$.

▶ **Lemma 23.** *The time complexity of a complete subtree $\mathcal{T}_j$ is $\mathcal{O}^*\big((c_4 k/\log k)^{(d-1)K_j}\big)$, where $c_4 = c_4(d, s)$ is a constant that depends on the family $\mathcal{C}$.*

Proof. By Lemma 22, the number of leaves in $\mathcal{T}_j$ is $L \leq \left(\frac{c_2 k^d}{(k-k_r)\log^{d-1} k}\right)^{K_j - k_r}$. Observe that at depth r, Inclusion-Exclusion runs in time $\mathcal{O}^*\left(2^{\frac{d-1}{2} k_r \log k}\right) = \mathcal{O}^*\left(k^{\frac{d-1}{2} k_r}\right)$. Since an inner node performs polynomial time work and the leaves perform exponential time work, this immediately implies that the running time for $\mathcal{T}_j$ is

$$\mathcal{O}^*\left(\left(\frac{c_2 k^d}{(k-k_r)\log^{d-1} k}\right)^{K_j - k_r} \cdot k^{\frac{d-1}{2} k_r}\right).$$

Suppose $k - k_r = o(k)$. Then it holds that $K_j - k_r = o(K_h)$ since $k \geq K_j \geq k_r$. We get

$$\mathcal{O}^*\left(\left(\frac{c_2 k^d}{(k-k_r)\log^{d-1} k}\right)^{o(K_j)} \cdot k^{\frac{d-1}{2}(K_j - o(K_j))}\right) = \mathcal{O}^*\left(2^{o(dK_j \log k) + \frac{d-1}{2}(K_j \log k - o(k \log k))}\right).$$

With some simple algebra one gets that the exponent is bounded by $(d-1)K_j(\log k - \log\log k)$, giving the desired time bound $\mathcal{O}^*\left(2^{(d-1)K_j(\log k - \log\log k)}\right) = \mathcal{O}^*\left((k/\log k)^{(d-1)K_j}\right)$.

If $k - k_r \neq o(k)$, then $k - k_r \geq c_3 k$ for some constant $c_3 > 0$. The running time solves to:

$$\mathcal{O}^*\left(\left(\frac{c_2 k^{d-1}}{c_3 \log^{d-1} k}\right)^{K_j - k_r} \cdot k^{\frac{d-1}{2}k_r}\right) = \mathcal{O}^*\left(\left(\frac{c_4 k}{\log k}\right)^{(d-1)K_j}\right)$$

where $c_4 = (c_2/c_3)^{1/(d-1)}$. ◀

▶ **Lemma 24.** *Let L_j be a depth $j < r$ leaf of $\mathcal{T}$ that calls* Inclusion-Exclusion. *Then the running time of $\mathcal{T}_j$ dominates that of L_j.*

Proof. By Lemma 23, the time complexity of $\mathcal{T}_j$ is $\mathcal{O}^*\left((c_4 k/\log k)^{(d-1)K_j}\right)$. At depth j the algorithm has K_j remaining budget to spend. Since the algorithm called Inclusion-Exclusion at this depth, at most $\frac{d-1}{2}K_j \log k$ points remained and the call takes $\mathcal{O}^*\left(2^{\frac{d-1}{2}K_j \log k}\right) = \mathcal{O}^*\left(k^{\frac{d-1}{2}K_j}\right)$ time, which is bounded by that for $\mathcal{T}_j$. ◀

▶ **Theorem 25.** CurveCover *decides Curve Cover in time $\mathcal{O}^*\left((Ck/\log k)^{(d-1)k}\right)$ where $C = C(d,s)$ is a constant that depends on the family $\mathcal{C}$.*

Proof. Fix a budget partition $\langle k_1, \ldots, k_r \rangle$. By Lemma 24, calling Inclusion-Exclusion at a depth $j < r$ does not increase the running time of the algorithm. Therefore the time complexity of CC-Recursive is $\mathcal{O}^*\left((c_4 k/\log k)^{(d-1)K_1}\right) = (c_4 k/\log k)^{(d-1)k}$.

CurveCover runs CC-Recursive over all possible budget partitions, of which by the "stars and bars" theorem are only $\binom{k+r-1}{k}$, a quasi-polynomial in k. Therefore by letting $C = c_4 + \varepsilon$ for any $\varepsilon > 0$, the time complexity of CurveCover is $\mathcal{O}^*\left((Ck/\log k)^{(d-1)k}\right)$. ◀

▶ **Lemma 26.** *The polynomial time dependency of* CurveCover *is $\mathcal{O}\left((k\log k)^{2+s}\right)$ and its space complexity is $\mathcal{O}\left(k^4 \log^2 k\right)$ bits.*

Proof. See the full version of the paper. ◀

5 Hyperplane Cover

One generalization of *Line Cover* was discussed in the previous section. In this section we discuss its other generalization *Hyperplane Cover*, and give an algorithm for the three dimensional case. We would like to follow the same basic attack plan of using incidence bounds but here we face significant challenges and we need non-trivial changes in our approach. One major challenge is the nature of incidences in higher dimensions. For example, the asymptotically maximum number of incidences between a set of points and hyperplanes in d-dimensions is obtained by placing half of the points on one two-dimensional plane (see [2, 5]) which clearly makes it an easy instance for our algorithm (due to kernelization). Thus, in essence, we need to use specialized incidence bounds that disallow such configurations of points; unfortunately, such bounds are more difficult to prove than ordinary incidence bounds (and as it turns out, also more difficult to use).

5.1 Point-Hyperplane incidence bounds in higher dimensions

The most general bound for point-hyperplane incidences from [2, 6] yields a bound of $\Theta\left(\frac{n^d}{\gamma^3} + \frac{n^{d-1}}{\gamma}\right)$ on the number of γ-rich hyperplanes in d dimensions similar to Lemma 20 (where the left term is again the significant one). Our method requires that the exponent is greater in the denominator than in the numerator, so this bound is not usable beyond $\mathbb{R}^2$. As stated before, the constructions that make the upper bound tight are easy cases for our algorithm; they contain very low dimensional flats that have many points on them. A specialized bound appears in [7], where the authors study the number of incidences between points and hyperplanes with a certain *saturation*.

▶ **Definition 27.** Consider a point set P and a hyperplane h in $\mathbb{R}^d$. We say that h is σ-saturated, $\sigma > 0$, if $H \cap P$ spans at least $\sigma \cdot |H \cap P|^{d-1}$ distinct $(d-2)$-flats of F.

For example in three dimensions, a $(1 - \frac{1}{n})$-saturated plane contains no three collinear points. The main theorem of [7] can be stated as follows.

▶ **Theorem 28** (Elekes and Tóth [7]). *Let $d \geq 2$ be the dimension and $\sigma > 0$ a real number. There is a constant $C_1(d, \sigma)$ with the following property. For every set P of n points in $\mathbb{R}^d$, the number of γ-rich σ-saturated hyperplanes is at most:*

$$\mathcal{O}\left(C_1(d,\sigma)\left(\frac{n^d}{\gamma^{d+1}} + \frac{n^{d-1}}{\gamma^{d-1}}\right)\right).$$

The interesting term in this bound has a greater exponent in the denominator, as required. Unfortunatley it is difficult to verify if a hyperplane is σ-saturated. In the same paper, the authors give another bound based on a more manageable property called *degeneracy*.

▶ **Definition 29.** Given a point set P and a hyperplane h in $\mathbb{R}^d$, we say that h is δ-degenerate, $0 < \delta \leq 1$, if $H \cap P$ is non-empty and at most $\delta \cdot |H \cap P|$ points of $H \cap P$ lie in any $(d-2)$-flat.

For example in $\mathbb{R}^3$, any 1-degenerate plane might have all its points lying on a single line, and a plane with degeneracy strictly less than 1 must have at least 3 points not on the same line. As such it is an easy property to test.

▶ **Theorem 30** (Elekes and Tóth [7]). *For any set of n points in $\mathbb{R}^3$, the number of γ-rich δ-degenerate planes is at most*

$$\mathcal{O}\left(\frac{1}{(1-\delta)^4}\left(\frac{n^3}{\gamma^4} + \frac{n^2}{\gamma^2}\right)\right).$$

This bound is usable and relies on an easily-tested property, but unfortunately only applies to the $\mathbb{R}^3$ setting.

5.2 Algorithm for *Plane Cover*

In this section we present our algorithm PC-Recursive that solves *Plane Cover* using the bound from Theorem 30. This algorithm is similar to that for *Curve Cover*, and it is assumed that the reader is sufficiently familiar with CC-Recursive before reading this section.

Recall that by Lemma 7, *Plane Cover* has a kernel of size $k^3 + k^2$ where no plane contains more than $k(k+1) \leq 2k^2$ points and no two planes pairwise intersect in more than $k+1$ points. For convenience we define $\gamma_0 = k^2 + k$ and $\gamma_i = k^2/2^i$ for $i > 0$. We inherit the basic structure of the CC-Recursive algorithm, such that every recursion level considers γ_i-rich-γ_{i-1}-poor candidates. Additionally, any candidate considered must be *not-too-degenerate*:

▶ **Definition 31.** Let $\delta_i = 1 - \gamma_i^{-1/5}$. A γ_i-rich-γ_{i-1}-poor plane is called *not-too-degenerate* if it is δ_i-degenerate, and *too-degenerate* otherwise.

It is of no consequence that the definition does not cover all candidates considered on depth 1. The main extension of PC-Recursive compared to CC-Recursive is to first use a different technique to deal with too-degenerate candidates, which then allows normal branching on the not-too-degenerate ones. The key observation is that any too-degenerate candidate has at least $\gamma_i\delta_i = \gamma_i - \gamma_i^{4/5}$ points on a line and at most $\gamma_{i-1}(1-\delta_i) = 2\gamma_i^{4/5}$ points not on it.

Suppose a k-cover contains some too-degenerate plane h. By correctly guessing its very rich line ℓ and removing the points on the line, the algorithm makes decent progress in terms of shrinking the instance. The points on h but not ℓ will remain in the instance even though the budget for covering them has been paid. These are called the *ghost points* of h (or of ℓ), and ℓ is called a *degenerate line*. The ghost points must be removed by extending the line ℓ into a full plane. But the ghost points are few enough that the algorithm can delay this action until a later recursion level. Specifically, for a line ℓ guessed at depth i, we extend ℓ into a plane at the first recursion depth which considers $2\gamma_i^{4/5}$-poor candidates, i.e. the depth j such that $\gamma_{j-1} \geq 2\gamma_i^{4/5} \geq \gamma_j$.

Therefore the algorithm keeps a separate structure $\mathcal{L}$ of lines that have been guessed to be degenerate lines on some planes in the solution. Augment $\mathcal{L}$ to remember the recursion depth that a line was added to it. At any recursion depth, the algorithm will deal with old-enough lines in $\mathcal{L}$, then guess a new set of degenerate lines to add to $\mathcal{L}$ before finally branching on not-too-degenerate planes.

The algorithm Let $r = \Theta(\log k)$ as before. Let $\langle h_1, \ell_1, \ldots, h_r, \ell_r\rangle$ with $\sum_{i=1}^{r} h_i + \ell_i = k$ be a budget partition. The recursive algorithm PC-Recursive takes 4 arguments: the point set P, a set of lines $\mathcal{L}$, the budget partition, and a recursion level i. A top level algorithm PlaneCover tries all budget partitions and calls PC-Recursive accordingly.

Let the current recursion depth be i, and let $K_i = \sum_{j=i}^{r} h_j + \ell_j$ be the remaining budget. The sub-budget h_i will be spent on not-too-degenerate planes, and ℓ_i on degenerate lines. Let $\mathcal{L}$ be an augmented set of lines as described above. This means that earlier levels have already created a partial solution of $k - (K_i + |\mathcal{L}|)$ planes, and a set $\mathcal{L}$ of lines that still need to be covered by a plane. If strictly more than $(K_i + |\mathcal{L}|)\gamma_{i-1}$ points remain, the algorithm rejects. If at most $K_i \log k$ points remain, the algorithm switches to Inclusion-Exclusion passing on the instance $\langle P \cup \mathcal{L}, K_i + |\mathcal{L}|\rangle$.

Let $f = \left\lceil \frac{5(i-1) - 2\log k}{4} \right\rceil$. Let A be the set of all lines in $\mathcal{L}$ that were added at depth f or earlier. Remove A from $\mathcal{L}$. For each way of placing $|A|$ planes $\mathcal{H}$ such that every plane contains one line in A and at least one point in P, let $P' = P \setminus (P \cap \mathcal{H})$ be the point set not covered by these planes. For a P', let H be the set of not-too-degenerate planes and L the set of degenerate lines too-degenerate candidates.

For every P' and every way of choosing h_i planes from H and ℓ_i lines from L, branch depth $i+1$ by removing the covered points from P and adding the chosen lines of L to $\mathcal{L}$.

5.3 Analysis

Correctness. To prove that the algorithm is correct, we follow a similar strategy as for CurveCover. We build on the notion that the algorithm is building up a partial solution of planes. Removing the points covered by the partial solution yields a "residual problem" just as in CurveCover. A partial solution is *correct* if it is a subset of some k-cover. Correctness

of the algorithm follows from proving that a k-cover exists if and only if one branch maintains a correct partial solution until it reaches Inclusion-Exclusion.

The difference here is that the residual problem is an instance of *Any-flat Plane Cover* and not *Plane Cover*. Therefore, we simply consider the original problem to be an instance of *Any-flat Plane Cover*, namely $R_1 = \langle P, \emptyset \rangle$. We say that $\mathcal{C}$ covers $\langle P, \mathcal{L} \rangle$ if $\mathcal{C}$ covers both P and $\mathcal{L}$. What needs to be established is that there is a correct way to replace points with lines (Observation 32) and, conversely, that there is a correct way to extend a line in $\mathcal{L}_i$ (Observation 33). The proofs for these are elementary and we omit them. Given these two facts, we can easily show that the algorithm will call Inclusion-Exclusion on appropriate instances.

▶ **Observation 32.** *Let ℓ be a line and $\mathcal{C}$ a set of planes such that some plane $h \in \mathcal{C}$ covers ℓ. Then $\mathcal{C}$ is a cover for $\langle P, \mathcal{L} \rangle$ if and only if $\mathcal{C}$ is a cover for $\langle P \setminus \ell, \mathcal{L} \cup \{\ell\} \rangle$.*

▶ **Observation 33.** *Let ℓ be a line, $\mathcal{L} \ni \ell$ be a set of lines, and $\mathcal{C}$ be a set of planes such that some $h \in \mathcal{C}$ covers ℓ but not any other line $\ell' \in \mathcal{L}$. Then $\mathcal{C}$ is a cover for $\langle P, \mathcal{L} \rangle$ if and only if $\mathcal{C} \setminus \{h\}$ is a cover of $\langle P \setminus h, \mathcal{L} \setminus \{\ell\} \rangle$.*

The conditions for Observation 33 might seem overly restrictive. But as the following lemma shows, that situation arises when $\mathcal{L}$ contains only correctly guessed degenerate lines.

▶ **Lemma 34.** *Let h be a too-degenerate plane with degenerate line ℓ such that $h \setminus \ell$ is a too-degenerate plane with degenerate line ℓ'. Then at no point during the execution of* PC-Recursive *will $\mathcal{L}$ contain ℓ and ℓ'.*

Proof sketch. The candidate $h \setminus \ell$ is too poor to be considered before the recursion depth where ℓ gets removed from L and extended to h. Full proof in full version of the paper. ◀

▶ **Lemma 35.** *If $\mathcal{L}$ contains only the degenerate lines of some too-degenerate planes in a k-cover, the number of ghost points at depth i is at most $|\mathcal{L}|\gamma_{i-1}$.*

Proof. See full version of the paper. ◀

▶ **Lemma 36.** *Algorithm* PlaneCover *decides whether P has a k-cover of planes.*

Proof. View the algorithm as being non-deterministic. Suppose P has a k-cover. Observation 32, Observation 33 and Lemma 34 guarantee that there is a correct path, and Lemma 35 guarantees that the point set is not erroneously rejected. Therefore the algorithm will send a yes-instance to Inclusion-Exclusion and accept.

Suppose P has no k-cover. If the conditions for Observation 33 are not satisfied, removing ℓ from $\mathcal{L}$ and pairing it up with points but not with another $\ell' \in \mathcal{L}$ can only reduce the number of solutions. Therefore the algorithm detects no cover and rejects. ◀

We can now state our main theorem for Plane-Cover.

▶ **Theorem 37.** PlaneCover *decides Plane Cover in $\mathcal{O}\left((Ck^2/\log^{1/5} k)^k\right)$ time for some constant C.*

Proof. See the full version of the paper. ◀

To give an idea of how to prove the above theorem, we give a sketch of the analysis that reflects the core of the real analysis. As before, we assume a (slightly incorrect) worst case for the budget partition where all the budget is assigned to the two deepest recursion levels. This gives a bound analogous to that in Lemma 22. After achieving this bound, the same arguments as for CurveCover can be applied to achieve the bound from Theorem 37.

Analysis sketch. The branching of the analysis is twofold. First there is the branching done on picking not-too-degenerate planes. Secondly, we have the branching on too-degenerate planes. This branching is actually a combination of picking the rich lines in too-degenerate planes, and the branching done by covering these lines with planes later on.

We sketch a proof for two extreme cases: either (**i**) $\forall i, k_i = h_i$ or (**ii**) $\forall i, \ell_i = k_i$. For both cases the branching can be bounded by $\left(\frac{k^3}{(k-k_r)\log^{1/5} k}\right)^{k-k_r}$ (compare to Lemma 22). The full proof for Theorem 37 shows that if the budget is distributed between these cases, then taking the product of the worst case running times of both cases is roughly the same as what we present here. As it turns out, the first case is (up to the incidence theorem used) identical to the curve case.

For case (**ii**) we again assume a (slightly incorrect) worst case budget partition where $k_{r-1} + k_r = k$ and $k_{r-1} \geq k_r$. By the same arguments as in the analysis sketch of Section 4 we have the following two parameters at recursion level $r-1$: the number of of points remaining is $n = k \log k$ and the richness γ_{r-1} is $\log k$. The algorithm picks γ_{i+1}-rich lines at level i, and these lines are matched with points at later level j where $\gamma_j = \gamma_i^{4/5}$. The cost for branching at level j is charged to level i, so that we can more easily analyze the total branching on lines selected at level i. With the budget partition as stated above, we can now bound the branching done at level $r-1$. By the Szemerédi-Trotter theorem, there are at most $\frac{n^2}{(\log k)^3} = \frac{k^2}{\log k}$ candidates, from which we select $k - k_r$ lines. This yields a total branching of $\binom{\frac{k^2}{\log k}}{k-k_r}$, which is roughly $\left(\frac{k^2}{(k-k_r)\log k}\right)^{k-k_r}$. We then need to mach these $k - k_r$ lines with $k \log^{4/5} k$ points, yielding a further branching of $(k \log^{4/5} k)^{k-k_r}$. Taking the product of both these branching factors gives $\left(\frac{k^2}{(k-k_r)\log k}\right)^{k-k_r} \cdot (k \log^{4/5} k)^{k-k_r} = \left(\frac{k^3}{(k-k_r)\log^{1/5} k}\right)^{k-k_r}$.

▶ **Lemma 38.** *The polynomial time dependency of* PlaneCover *is* $\mathcal{O}\left(k^4 \log^4 k\right)$ *and its space complexity is* $\mathcal{O}\left(k^6 \log^2 k\right)$ *bits.*

Proof. See the full version of the paper. ◀

6 Discussion

We have presented a general algorithm that improves upon previous best algorithms for all variations of *Curve Cover* as well as for the *Hyperplane Cover* problem in $\mathbb{R}^3$. Given good incidence bounds it should not be difficult to apply this algorithm to more geometric covering problems. However, such bounds are difficult to obtain in higher dimensions and for *Hyperplane Cover* the bound $\mathcal{O}\left(n^d/\gamma^3\right)$ is tight when no constraints are placed on the input, but it is too weak to be used even in $\mathbb{R}^3$. The bound by Elekes and Tóth works when the hyperplanes are well saturated, but the convenient relationship between saturation and degeneracy on hyperplanes does not extend past the $\mathbb{R}^3$ setting. Our hyperplane kernel guarantees a bound on the number of points on any j-flat. This overcomes the worst-case constructions for known incidence bounds, which involve placing very many points on the same line. An incidence bound for a kernelized point set might provide the needed foundation for similar *Hyperplane Cover* algorithms in higher dimensions.

References

1 Peyman Afshani, Edvin Berglin, Ingo van Duijn, and Jesper Sindahl Nielsen. Applications of incidence bounds in point covering problems. *arXiv:1603.07282*, 2016.

2 Pankaj K Agarwal and Boris Aronov. Counting facets and incidences. *Discrete & Computational Geometry*, 7(1):359–369, 1992.

3 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009.

4 Cheng Cao. Study on two optimization problems: Line cover and maximum genus embedding. Master's thesis, Texas A&M University, 2012.

5 Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer Publishing Company, Incorporated, 1st edition, 2012.

6 Herbert Edelsbrunner, Leonidas Guibas, and Micha Sharir. The complexity of many cells in arrangements of planes and related problems. *Discrete & Computational Geometry*, 5(1):197–216, 1990.

7 György Elekes and Csaba D Tóth. Incidences of not-too-degenerate hyperplanes. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 16–21. ACM, 2005.

8 Vladimir Estivill-Castro, Apichat Heednacram, and Francis Suraweera. FPT-algorithms for minimum-bends tours. *International Journal of Computational Geometry & Applications*, 21(02):189–213, 2011.

9 Jacob Fox, János Pach, Adam Sheffer, Andrew Suk, and Joshua Zahl. A semi-algebraic version of Zarankiewicz's problem. *arXiv preprint arXiv:1407.5705*, 2014.

10 Magdalene Grantson and Christos Levcopoulos. *Covering a set of points with a minimum number of lines*. Springer, 2006.

11 Ben Joseph Green and Terence Tao. On sets defining few ordinary lines. *Discrete & Computational Geometry*, 50(2):409–468, 2013.

12 Leonidas J Guibas, Mark H Overmars, and Jean-Marc Robert. The exact fitting problem in higher dimensions. *Computational geometry*, 6(4):215–230, 1996.

13 LJ Guibas, Mark Overmars, and Jean-Marc Robert. The exact fitting problem for points. In *Proc. 3rd Canadian Conference on Computational Geometry*, pages 171–174, 1991.

14 Stefan Kratsch, Geevarghese Philip, and Saurabh Ray. Point line cover: The easy kernel is essentially tight. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1596–1606. SIAM, 2014.

15 VS Anil Kumar, Sunil Arya, and Hariharan Ramesh. Hardness of set cover with intersection 1. In *Automata, Languages and Programming*, pages 624–635. Springer, 2000.

16 Stefan Langerman and Pat Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005.

17 Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations research letters*, 1(5):194–197, 1982.

18 János Pach and Micha Sharir. On the number of incidences between points and curves. *Combinatorics, Probability and Computing*, 7(01):121–127, 1998.

19 József Solymosi and Terence Tao. An incidence theorem in higher dimensions. *Discrete & Computational Geometry*, 48(2):255–280, 2012.

20 Endre Szemerédi and William T Trotter Jr. Extremal problems in discrete geometry. *Combinatorica*, 3(3-4):381–392, 1983.

21 Praveen Tiwari. On covering points with conics and strips in the plane. Master's thesis, Texas A&M University, 2012.

22 Jianxin Wang, Wenjun Li, and Jianer Chen. A parameterized algorithm for the hyperplane-cover problem. *Theoretical Computer Science*, 411(44):4005–4009, 2010.

Grouping Time-Varying Data for Interactive Exploration

Arthur van Goethem[*1], **Marc van Kreveld**[2], **Maarten Löffler**[3], **Bettina Speckmann**[†4], **and Frank Staals**[‡5]

1 **Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
a.i.v.goethem@tue.nl
2 **Dept. of Computing and Information Sciences, Utrecht University, The Netherlands**
m.j.vankreveld@uu.nl
3 **Dept. of Computing and Information Sciences, Utrecht University, The Netherlands**
m.loffler@uu.nl
4 **Dept. of Mathematics and Computer Science, TU Eindhoven, The Netherlands**
b.speckmann@tue.nl
5 **MADALGO, Aarhus University, Denmark**
f.staals@cs.au.dk

Abstract

We present algorithms and data structures that support the interactive analysis of the grouping structure of one-, two-, or higher-dimensional time-varying data while varying all defining parameters. Grouping structures characterise important patterns in the temporal evaluation of sets of time-varying data. We follow Buchin et al. [9] who define groups using three parameters: group-size, group-duration, and inter-entity distance. We give upper and lower bounds on the number of maximal groups over all parameter values, and show how to compute them efficiently. Furthermore, we describe data structures that can report changes in the set of maximal groups in an output-sensitive manner. Our results hold in $\mathbb{R}^d$ for fixed d.

1998 ACM Subject Classification F.2.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases trajectory, time series, moving entity, grouping, algorithm, data structure

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.61

1 Introduction

Time-varying phenomena are ubiquitous and hence the rapid increase in available tracking, recording, and storing technologies has led to an explosive growth in time-varying data. Such data comes in various forms: time-series (tracking a one-dimensional variable such as stock prices), two- or higher-dimensional trajectories (tracking moving objects such as animals, cars, or sport players), or ensembles (sets of model runs under varying initial conditions for one-dimensional variables such as temperature or rain fall), to name a few. Efficient tools to extract information from time-varying data are needed in a variety of applications, such as

* A. v. G. is supported by the Netherlands Organisation for Scientific Research (NWO) under project nr. 612.001.102.
† B. S. is supported by the Netherlands Organisation for Scientific Research (NWO) under project nr. 639.023.208.
‡ F. S. is supported by the Danish National Research Foundation under grant nr. DNRF84.


© Arthur van Goethem, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals; licensed under Creative Commons License CC-BY

predicting traffic flow [18], understanding animal movement [7], coaching sports teams [13], or forecasting the weather [21]. Consequently, recent years have seen a flurry of algorithmic methods to analyse time-varying data which can, for example, identify important geographical locations from a set of trajectories [6, 15], determine good average representations [8], or find patterns, such as groups traveling together [9, 14, 17].

Most, if not all, of these algorithms use several parameters to model the applied problem at hand. The assumption is that the domain scientists, who are the users of the algorithm, know from years of experience which parameter values to use in their analysis. However, in many cases this assumption is not valid. Domain scientists do *not* always know the correct parameter settings and in fact need algorithmic support to interactively explore their data in, for example, a visual analytics system [3, 16].

We present algorithms and data structures that support the interactive analysis of the grouping structure of one-, two-, or higher-dimensional time-varying data while varying all defining parameters. Grouping structures (which track the formation and dissolution of groups) characterise important patterns in the temporal evaluation of sets of time-varying data. Classic examples are herds of animals or groups of people. But also for one-dimensional ensembles grouping is meaningful, for example, when detecting trends in weather models [20].

Buchin et al. [9] proposed a grouping structure for sets of moving entities. Their definition was later extended by Kostitsyna et al. [17] to geodesic distances. In this paper we use the same trajectory grouping structure. Our contributions are data structures and query algorithms that allow the parameters of the grouping structure to vary interactively and hence make it suitable for explorative analysis of sets of time-varying data. Below we first briefly review the definitions of Buchin et al. [9] and then state our contributions in detail.

Trajectory grouping structure [9]. Let $\mathcal{X}$ be a set of n entities moving in $\mathbb{R}^d$ and let $\mathbb{T}$ denote time. The entities trace trajectories in $\mathbb{T} \times \mathbb{R}^d$. We assume that each individual trajectory is piecewise linear and consists of at most τ vertices. Two entities a and b are *ε-connected* if there is a chain of entities $a = c_1, .., c_k = b$ such that for any pair of consecutive entities c_i and c_{i+1} the distance is at most ε. A set G is ε-connected, if for any pair $a, b \in G$, the entities are ε-connected. Given parameters m, ε, and δ, a set of entities G is an (m, ε, δ)-*group* during time interval I if (and only if) (i) G has size at least m, (ii) $\mathit{duration}(I) \geq \delta$, and (iii) G is *ε-connected* at any time $t \in I$. An (m, ε, δ)-group (G, I) is *maximal* if G is maximal in size or I is maximal in duration, that is, if there is no group $H \supset G$ that is also ε-connected during I, and no interval $J \supset I$ such that G is ε-connected during J.

Results and Organization. We want to create a data structure $\mathcal{D}$ that represents the grouping structure, that is, its maximal groups, while allowing us to efficiently change the parameters. As we show below, the complexity of the problem is already fully apparent for one-dimensional time-varying data. Hence we restrict our description to $\mathbb{R}^1$ in Sections 2–4 and then explain in Section 5 how to extend our results to higher dimensions.

If all three parameters m, ε, and δ can vary independently the question arises what constitutes a meaningful maximal group. Consider a maximal (m, ε, δ)-group (G, I). If we slightly increase ε to ε', and consider a slightly longer time interval $I' \supseteq I$ then (G, I') is a maximal $(m, \varepsilon', \delta)$-group. Intuitively, these groups (G, I) and (G, I') are the same. Thus, we are interested only in (maximal) groups that are "combinatorially different". Note that the set of entities G may also be a maximal (m, ε, δ)-group during a time interval J completely different from I, we also wish to consider (G, I) and (G, J) to be combinatorially different groups. In Section 2 we formally define when two (maximal) (m, ε, δ)-groups are

(combinatorially) different. We prove that there are at most $O(|\mathcal{A}|n^2)$ such groups, where $\mathcal{A}$ is the arrangement of the trajectories in $\mathbb{T} \times \mathbb{R}^1$, and $|\mathcal{A}|$ is its complexity. We also argue that the number of maximal groups may be as large as $\Omega(\tau n^3)$, even for fixed parameters m, ε, and δ and in $\mathbb{R}^1$. This significantly strengthens the lower bound of Buchin et al. [9].

In Section 3 we present an $O(|\mathcal{A}|n^2 \log^2 n)$ time algorithm to compute all combinatorially different maximal groups. In Section 4 we describe a data structure that allows us to efficiently obtain all groups for a given set of parameter values. Furthermore we also describe data structures for the interactive exploration of the data. Specifically, given the set of maximal (m, ε, δ)-groups we want to change one or more of the parameters and efficiently report only those maximal groups which either ceased to be a maximal group or became a maximal group. That is, our data structures can answer so-called *symmetric-difference queries* which are gaining in importance as part of interactive analysis systems [12]. As mentioned above, in Section 5 we extend our data structures and algorithms to $\mathbb{R}^d$, for fixed d.

2 Combinatorially Different Maximal Groups

We consider entities moving in $\mathbb{R}^1$, hence the trajectories form an arrangement $\mathcal{A}$ in $\mathbb{T}\times\mathbb{R}^1$. We assume that no three pairs of entities have equal distance at the same time. Consider the four-dimensional *parameter space* $\mathbb{P}$ with axes time, size, distance, and duration. A set of entities G defines a region A_G in this space in which it is *alive*: a point $p = (p_t, p_m, p_\varepsilon, p_\delta) = (t, m, \varepsilon, \delta)$ lies in A_G if and only if G is a (m, ε, δ)-group at time t. We use these regions to define when groups are combinatorially different. First (Section 2.1) we fix $m = 1$ and $\delta = 0$ and define and count the number of combinatorially different maximal $(1, \varepsilon, 0)$-groups, over all choices of parameter ε. We then extend our results to include other values of δ and m in Section 2.2.

2.1 The Number of Distinct Maximal $(1, \varepsilon, 0)$-Groups, over all ε

Consider the (t, ε)-plane in $\mathbb{P}$ through $\delta = 0$ and $m = 1$. The intersection of all regions A_G with this plane give us the points (t, ε) for which G is a $(1, \varepsilon, 0)$-group. Note that G is a $(1, \varepsilon, 0)$-group at time t if and only if the set G is ε-connected at time t. Hence the region A_G, restricted to this plane, corresponds to the set of points (t, ε) for which G is ε-connected. A_G, restricted to this plane, is simply connected. Furthermore, as the distance between any pair of entities moving in $\mathbb{R}^1$ varies linearly, A_G is bounded from below by a t-monotone polyline f_G. The region is unbounded from above: if G is ε-connected (at time t) for some value ε, then it is also ε'-connected for any $\varepsilon' \geq \varepsilon$ (see Figure 1). Every maximal length segment in the intersection between (the restricted) A_G and the horizontal line ℓ_ε at height ε corresponds to a (maximal) time interval I during which (G, I) is a $(1, \varepsilon, 0)$-group, or an *ε-group* for short. Every such a segment corresponds to an *instance* of ε-group G.

▶ **Observation 1.** *Set G is a maximal ε-group on I, iff the line segment $s_{\varepsilon,I} = \{(t, \varepsilon) \mid t \in I\}$ is a maximal length segment in A_G, and is not contained in A_H, for a supergroup $H \supset G$.*

Two instances of ε-group G may *merge*. Let v be a local maximum of f_G and $I_1 = [t_1, v_t]$ and $I_2 = [v_t, t_2]$ be two instances of group G meeting at v. At v_ε, the two instances G that are alive during $[t_1, v_t]$ and $[v_t, t_2]$ merge and we now have a single time interval $I = [t_1, t_2]$ on which G is a group. We say that I is a new instance of G, different from I_1 and I_2. We can thus decompose A_G into maximally-connected regions, each corresponding to a distinct instance of group G, using horizontal segments through the local maxima of f_G. We further split each region at the values ε where G changes between being maximal and being dominated. Let $\mathcal{P}_G$ denote the obtained set of regions in which G is maximal. Each such a

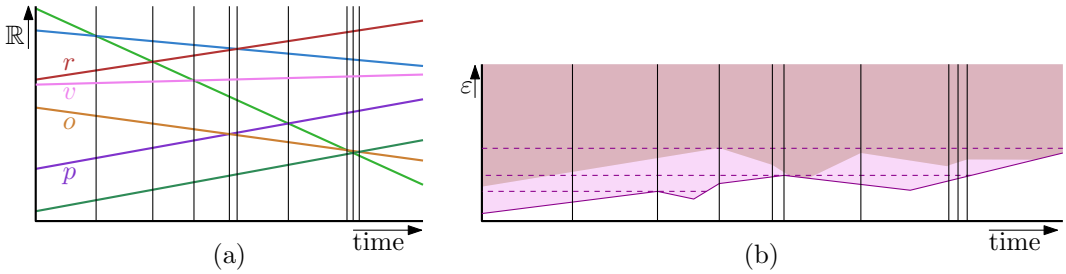


Figure 1 (a) A set of trajectories for a set of entities moving in $\mathbb{R}^1$ (b) The region $A_{\{r,v\}}$ during which $\{r, v\}$ is alive, and its decomposition into polygons, each corresponding to a distinct instance. In all such regions, except the top one $\{r, v\}$ is a maximal group: in the top region $\{r, v\}$ is dominated by $\{r, v, o\}$ (darker region).

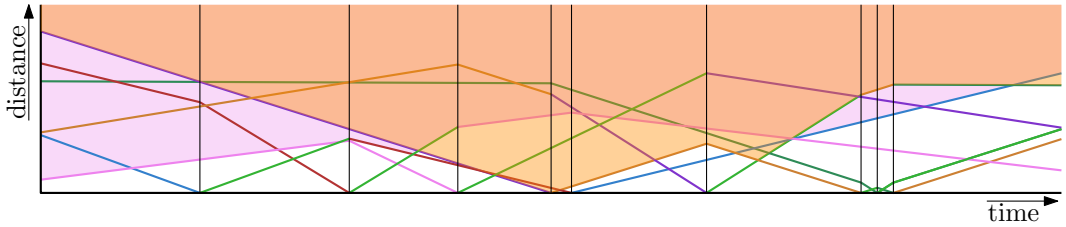


Figure 2 The arrangement $\mathcal{H}$ and the regions $A_{\{r,v\}}$ (purple) and $A_{\{p,o\}}$ (orange) for the trajectories shown in Figure 1(a). The arrangement $\mathcal{H}$ corresponds to the arrangement of functions $h_a(t)$ that represent the distance from a to the entity directly above a at time t.

region P corresponds to a *combinatorially distinct* instance on which G is a maximal group (with at least one member and duration at least zero). The region P is bounded by at most two horizontal line segments and two ε-monotone chains (see Figure 1(b)).

Counting maximal ε-groups. To bound the number of distinct maximal ε-groups, over all values of ε, we have to count the number of polygons in $\mathcal{P}_G$ over all sets G. While there are possibly exponentially many sets, there is structure in the regions A_G which we can exploit.

Consider a set of entities G and a region $P \in \mathcal{P}_G$ corresponding to a distinct instance of the maximal ε-group G. We observe that all vertices of P lie on the polyline f_G: they are either directly vertices of f_G, or they are points (t, ε) on the edges of f_G where G starts or stops being maximal. For the latter case there must be a polyline f_H, for some subgroup or supergroup of G, that intersects f_G at such a point. Furthermore, observe that any vertex (of either type) is used by at most a constant number of regions from $\mathcal{P}_G$.

Below we show that the complexity of the arrangement $\mathcal{H}$, of all polylines f_G over all G, is bounded by $O(|\mathcal{A}|n)$. Furthermore, we show that each vertex of $\mathcal{H}$ can be incident to at most $O(n)$ regions. It follows that the complexity of all polygons $P \in \mathcal{P}_G$, over all groups (sets) G, and thus also the number of such sets, is at most $O(|\mathcal{A}|n^2)$.

The complexity of $\mathcal{H}$. The *span* $S_G(t) = \{a \mid a \in \mathcal{X} \ \wedge \ a(t) \in [\min_{b\in G} b(t), \max_{b\in G} b(t))\}$ of a set of entities G at time t is the set of entities between the lowest and highest entity of G at time t (for technical reasons, we include the lowest entity of G in the span, but not the highest). Let $h_a(t)$ denote the distance from entity a to the entity directly above a at time t, that is, $h_a(t)$ is the height of the face in $\mathcal{A}$ that has a on its lower boundary at time t.

▶ **Observation 2.** *A set G is ε-connected at time t, if and only if the largest distance among consecutive entities in $S_G(t)$ is at most ε. That is,*

$$f_G(t) = \max_{a \in S_G(t)} h_a(t)$$

It follows that $\mathcal{H}$ is a subset of the arrangement of the n functions h_a, for $a \in \mathcal{X}$ (see Figure 2). We use this fact to show that $\mathcal{H}$ has complexity at most $O(|\mathcal{A}|n)$:

▶ **Lemma 3.** *Let $\mathcal{A}$ be an arrangement of n line segments, and let k be the maximum number of line segments intersected by a vertical line. The number of triplets (F, F', x) such that the faces $F \in \mathcal{A}$ and $F' \in \mathcal{A}$ have equal height h at x-coordinate x is at most $O(|\mathcal{A}|k) \subseteq O(|\mathcal{A}|n) \subseteq O(n^3)$.*

▶ Remark. Interestingly, this bound is tight in the worst case. In the full version of this paper we give a construction where there are $\Omega(n^3)$ triplets (F, F', x) such that F and F' have equal height at x, even if we use lines instead of line segments [22].

▶ **Lemma 4.** *The arrangement $\mathcal{H}$ has complexity $O(|\mathcal{A}|n)$.*

What remains to show is that each vertex v of $\mathcal{H}$ can be incident to at most $O(n)$ polygons from different sets. We use Lemma 5, which follows from Buchin et al. [9]:

▶ **Lemma 5.** *Let $\mathcal{R}$ be the Reeb graph for a fixed value ε capturing the movement of a set of n entities moving along piecewise-linear trajectories in $\mathbb{R}^d$ (for some constant d), and let v be a vertex of $\mathcal{R}$. There are at most $O(n)$ maximal groups that start or end at v.*

▶ **Lemma 6.** *Let v be a vertex of $\mathcal{H}$. Vertex v is incident to at most $O(n)$ polygons from $\mathcal{P} = \bigcup_{G \subseteq \mathcal{X}} \mathcal{P}_G$.*

Proof. Let $P \in \mathcal{P}_G$ be a region that uses v. Thus, G either starts or ends as a maximal v_ε-group at time v_t. This means, v correspond to a single vertex u in the Reeb graph, built with parameter v_ε. By Lemma 5, there are at most $O(n)$ maximal v_ε-groups that start or end at u. Hence, v can occur in regions of at most $O(n)$ different sets G. For a fixed set G, the regions in $\mathcal{P}_G$ are disjoint, so there are only $O(1)$ regions from $\mathcal{P}_G$, that contain v. ◀

▶ **Lemma 7.** *The number of distinct ε-groups, over all values ε, and the total complexity of all regions $\mathcal{P} = \bigcup_{G \subseteq \mathcal{X}} \mathcal{P}_G$, are both at most $O(|\mathcal{H}|n) = O(|\mathcal{A}|n^2)$.*

2.2 The Number of Distinct Maximal Groups, over all Parameters

Maximal groups are monotonic in m and δ (see Buchin et al. [9]); hence a maximal (m, ε, δ)-group is also a maximal $(m', \varepsilon, \delta')$-group for any parameters $m' \leq m$ and $\delta' \leq \delta$. It follows that the number of combinatorially different maximal groups is still at most $O(|\mathcal{A}|n^2)$.

For the complexity of the regions in $\bigcup \mathcal{P}_G$: fix $m = 0$, and consider the remaining subspace of $\mathbb{P}$ with axes time, distance, and duration, and the restriction of A_G, for any set G, into this space. In the $\delta = 0$ plane we simply have the regions A_G, that are bounded from below by a t-monotone polyline f_G, as described in Section 2.1. As we increase δ we observe that the local minima in the boundary f_G get replaced by a horizontal line segment of width δ (see Figure 3). For arbitrarily small values of $\delta > 0$, the total complexity of this boundary is still $O(|\mathcal{A}|n^2)$. Further increasing δ, monotonically decreases the number of vertices on the functions f_G. It follows that the regions A_G, restricted to the time, distance, duration space also have total complexity $O(|\mathcal{A}|n^2)$. Finally, consider the regions A_G in the full four dimensional space. Clearly, $A_G \cap \{p \mid p \in \mathbb{P} \wedge p_m < |G|\} = \emptyset$. For values $m \geq |G|$, the boundary of A_G is constant in m. We conclude:

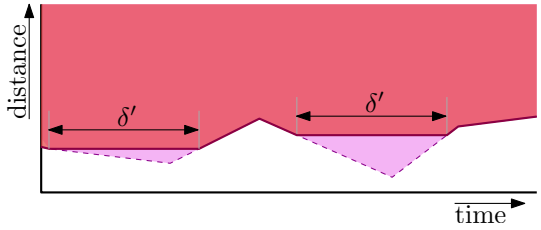


Figure 3 A cross section of the region $A_{\{r,v\}}$ with the plane through $\delta = \delta'$. The boundary of the original region (i.e. the cross section with the plane through $\delta = 0$) is dashed.

▶ **Theorem 8.** *Let $\mathcal{X}$ be a set of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^1$, and let $\mathcal{A}$ be the resulting trajectory arrangement. The number of distinct maximal groups is at most $O(|\mathcal{A}|n^2) = O(\tau n^4)$, and the total complexity of all regions in the parameter space corresponding to these groups is also $O(|\mathcal{A}|n^2) = O(\tau n^4)$.*

In the full version [22] we prove Lemma 9: even for fixed parameters ε, m, and δ, the number of maximal (m, ε, δ)-groups, for entities moving in $\mathbb{R}^1$, may be as large as $\Omega(\tau n^3)$. This strengthens the result of Buchin et al. [9], who established this bound for entities in $\mathbb{R}^2$.

▶ **Lemma 9.** *For a set $\mathcal{X}$ of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^1$, there can be $\Omega(\tau n^3)$ maximal ε-groups.*

3 Algorithm

In the following we refer to combinatorially different maximal groups simply as groups. Our algorithm computes a representation (of size $O(|\mathcal{A}|n^2)$) of all groups, which we can use to list all groups and, given a pointer to a group G, list all its members and the polygon $Q_G \in \mathcal{P}_G$. We assume $\delta = 0$ and $m = 1$, since the sets of maximal groups for $\delta > 0$ and $m > 1$ are a subset of the set for $\delta = 0$ and $m = 1$.

3.1 Overview

Our algorithm uses the arrangement $\mathcal{H}$ located in the (t, ε)-plane. Line segments in $\mathcal{H}$ correspond to the height function of the faces in $\mathcal{A}$. Let $a, b \in S_G(t)$ be the pair of consecutive entities in the span of a group G with maximum vertical distance at time t. We refer to (a, b) as the *critical pair* of G at time t. The pair (a, b) determines the minimal value of ε that is required for the group G to be ε-connected at time t. The distance between a critical pair (a, b) defines an edge of the polygon bounding G in $\mathcal{H}$.

Our representation will consist of the arrangement $\mathcal{H}$ in which each edge e is annotated with a data structure $\mathcal{T}_e$, a list $\mathcal{L}$ (or array) with the top edge in each *group polygon* $Q_G \in \mathcal{P}_G$, and an additional data structure $\mathcal{S}$ to support reconstructing the grouping polygons. We start by computing the arrangement $\mathcal{H}$. This takes $O(|\mathcal{H}|) = O(\tau n^3)$ time [2]. The arrangement is built from the set of height-functions of the faces of $\mathcal{A}$. With each edge we store the pair of edges in $\mathcal{A}$ responsible for it.

Given arrangement $\mathcal{H}$ we use a sweep line algorithm to construct the rest of the representation. A horizontal line $\ell(\varepsilon)$ is swept at height ε upwards, and all groups G whose group polygon Q_G currently intersects ℓ are maintained. To achieve this we maintain a two-part status structure. First, a set $\mathcal{S}$ with for each group G the time interval $I(G, \varepsilon) = Q_G \cap \ell(\varepsilon)$. Second, for each edge $e \in \mathcal{H}$ intersected by $\ell(\varepsilon)$ a data structure $\mathcal{T}_e$ with the sets of entities

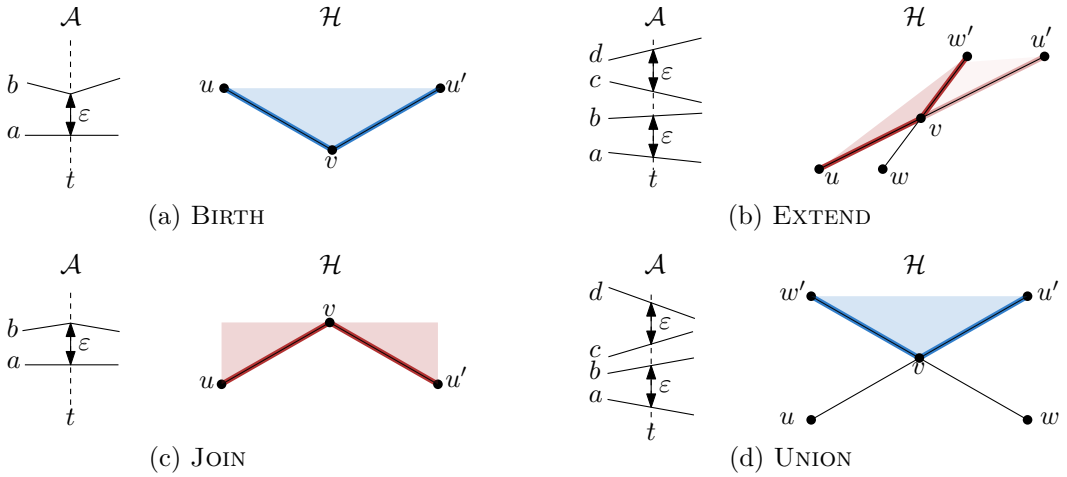


Figure 4 The different types of vertex events shown both in the arrangement $\mathcal{A}$ and in $\mathcal{H}$. The Extend event has a horizontally symmetric case.

whose time interval starts or ends at e, that is, $G \in \mathcal{T}_e$ if and only if $I(G, \varepsilon) = [s, t]$ with $s = e \cap \ell(\varepsilon)$ or $t = e \cap \ell(\varepsilon)$. We postpone the implementation of $\mathcal{T}$ to Section 3.3. The data structures support the following operations:

Operation	Input	Action
Filter$(\mathcal{T}_e, X)$	A data structure $\mathcal{T}_e$ A set of entities X	Create a data structure $\mathcal{T}' = \{G \cap X \mid G \in \mathcal{T}_e\}$
Insert$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G	Create a data structure $\mathcal{T}' = \mathcal{T}_e \cup \{G\}$.
Delete$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G	Create a data structure $\mathcal{T}' = \mathcal{T}_e \setminus \{G\}$.
Merge$(\mathcal{T}_e, \mathcal{T}_f)$	Two data structures $\mathcal{T}_e, \mathcal{T}_f$, belonging to two edges e, f having the same starting or ending vertex	Create a data structure $\mathcal{T}' = \mathcal{T}_e \cup \mathcal{T}_f$.
Contains$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G ending or starting on edge e	Test if $\mathcal{T}_e$ contains set G.
HasSuperSet$(\mathcal{T}_e, G)$	A data structure $\mathcal{T}_e$ A pointer to a representation of G ending or starting on edge e	Test if $\mathcal{T}_e$ contains a set $H \supseteq G$, and return the smallest such set if so.

The end points of the time interval $I(G, \varepsilon) = [start(G, \varepsilon), end(G, \varepsilon)]$ vary non-stop along the sweep. For each group G, the set $\mathcal{S}$ instead stores the edges e and f of $\mathcal{H}$ that contain the starting time $start(G, \varepsilon)$ and ending time $end(G, \varepsilon)$, respectively, and pointers to the representation of G in $\mathcal{T}_e$ and $\mathcal{T}_f$. We refer to e and f as the *starting edge* and *ending edge* of G. In addition, we store with each interval $I(G, \varepsilon)$ a pointer to the previous version of the interval $I(G, \varepsilon')$ if (and only if) the starting time (ending time) of G changed to edge e (edge f) at ε'. Note that updates for both $\mathcal{S}$ and $\mathcal{T}$ occur only when a vertex is hit by the sweep line $\ell(\varepsilon)$. For all unbounded groups we add $I(G, \infty)$ to $\mathcal{L}$ after the sweep line algorithm.

3.2 Sweepline Events

The sweep line algorithm results in four different vertex events (see Figure 4). The EXTEND-event has a symmetrical version in which $\overline{uu'}$ and $\overline{ww'}$ both have a negative incline. We describe how to update our hypothetical data structures in all cases.

Case I – Birth. Vertex v is a local minimum of one of the functions h_a, with $a \in \mathcal{X}$ (see Figure 4(a)). When the sweep line intersects v a new maximal group G is born. We can find the maximal group spawned in $O(|G|)$ time by checking which trajectories are ε-connected for this value of t and ε. To this end we traverse the (vertical decomposition of) $\mathcal{A}$ starting at the entities defining v.

Case II – Extend. Vertex v is the intersection of two line segments $s_{ab} = \overline{uu'}$ and $s_{cd} = \overline{ww'}$, both with a positive incline (see Figure 4(b)). The case in which s_{ab} and s_{cd} have negative incline can be handled symmetrically. Assume without loss of generality that s_{cd} is steeper than s_{ab}. We start with the following observation:

▶ **Observation 10.** *None of the groups arriving on edge (w, v) continue on edge (v, u').*

Proof. Let G be a group that arrives at v using edge (w, v). As G uses (w, v), it must contain entities both above and below the face F defined by critical pair (c, d). We know that u_ε and w_ε are strictly smaller than v_ε and ε is never smaller than zero. Thus, v_ε is strictly positive and F has a strictly positive height at t. Therefore, G still contains entities above and below F after time t. But then the critical pair (c, d) is still part of G and s_{cd} is a lower bound for the group. It follows that G must use edge (v, w'). ◀

We first compute the groups on outgoing edge (v, u'). By Observation 10 all these groups arrive on edge (u, v). In particular, they are the maximal size subsets from $\mathcal{T}_{(u,v)}$ for which all entities lie below entity d at time t, that is, $\mathcal{T}_{(v,u')} = \textsc{Filter}(\mathcal{T}_{(u,v)}, \mathit{below}(d, t))$, where $\mathit{below}(y, t) = \{x \mid x \in \mathcal{X} \wedge x(t) < y(t)\}$. For each group G in $\mathcal{T}_{(v,u')}$ we update the time-interval in $\mathcal{S}$. If G was dominated by a maximal group $H \supset G$ on incoming edge (u, v), we insert a new time interval with starting edge $f = \mathit{start}(H, \varepsilon)$ and ending edge (v, u') into $\mathcal{S}$, and insert G into $\mathcal{T}_f$. Note that G and H indeed have the same starting time: G is a subset of H, and is thus ε-connected at any time where H is ε-connected. Since G was not maximal before, it did not start earlier than H either.

The groups from $\mathcal{T}_{(u,v)}$ that contain entities on both sides of critical pair (c, d), continue onto edge (v, w'). Let $\mathcal{T}'$ denote these groups. We update the interval $I(G)$ in $\mathcal{S}$ for each group $G \in \mathcal{T}'$ by setting the ending edge to (v, w').

Next, we determine which groups from $\mathcal{T}_{(w,v)}$ die at v. A maximal group $G \in \mathcal{T}_{(w,v)}$ dies at v if there is a group H on (v, w') that dominates G. Any such group H must arrive at v by edge (u, v). Hence, for each group $G \in \mathcal{T}_{(w,v)}$ we check if there is a group $H \in \mathcal{T}'$ with $H \supset G$ and $I(H) \supseteq I(G)$. For each of these groups we remove the interval $I(G, \varepsilon)$ from $\mathcal{S}$, add $I(G, \varepsilon)$ to $\mathcal{L}$, and delete the set G from the data structure $\mathcal{T}_f$, where f is the starting edge of G (at height ε).

The remaining (not dominated) groups from $\mathcal{T}_{(w,v)}$ continue onto edge (v, w'). Let $\mathcal{T}''$ denote this set. We obtain $\mathcal{T}_{(v,w')}$ by merging $\mathcal{T}'$ and $\mathcal{T}''$, that is, $\mathcal{T}_{(v,w')} = \textsc{Merge}(\mathcal{T}', \mathcal{T}'')$. Since we now have the data structures $\mathcal{T}_{(v,u')}$ and $\mathcal{T}_{(v,w')}$, and we updated $\mathcal{S}$ accordingly, our status structure again reflects the maximal groups currently intersected by the sweep line.

Case III – Join. Vertex v is a local maximum of one of the functions h_a, with $a \in \mathcal{X}$ (see Figure 4(c)). Two combinatorially different maximal groups G_u and G_w with the same set of entities die at v (and get replaced by a new maximal group G^*) if and only if G_u is a maximal group in $\mathcal{T}_{(u,v)}$ and G_w is a maximal group in $\mathcal{T}_{(w,v)}$. We test this with a call to CONTAINS($\mathcal{T}_{(w,v)}, G_u$) for each group $G_u \in \mathcal{T}_{(u,v)}$. Let G be a group in $\mathcal{T}_{(u,v)}$, and let $H \in \mathcal{T}_{(w,v)}$ be the smallest supergroup of G, if such a group exists. At v the group G will immediately extend to the ending edge of H. We can find H by using a HASSUPERSET($\mathcal{T}_{(w,v)}, G$) call. If H exists we insert G into $\mathcal{T}_e$, and update $I(G, \varepsilon)$ in $\mathcal{S}$ accordingly. We process the groups G in $\mathcal{T}_{(w,v)}$ that have a group $H \in \mathcal{T}_{(u,v)}$ whose starting time jumps at v analogously.

Case IV – Union. Vertex v is the intersection of a line segment $s_{ab} = \overline{uu'}$ with positive incline and a line segment $s_{cd} = \overline{ww'}$, with negative incline (see Figure 4(d)). The UNION event is a special case of the BIRTH event. Incoming groups on edge (u, v) are below the line segment s_{cd} and, hence, can not contain any elements that are above c. As a consequence the line segment s_{cd} does not limit these groups and for a group $G \in \mathcal{T}_{(u,v)}$ we can safely add it to $\mathcal{T}_{(v,u')}$. We also update the interval $I(G)$ in $\mathcal{S}$ by setting the ending edge to (v, u'). An analogous argument can be made for groups arriving on edge (w, v).

Furthermore a new maximal group is formed. Let H be the set of all entities ε-connected to entity a at time t. We insert H into $\mathcal{T}_{(v,u')}$ and $\mathcal{T}_{(v,w')}$ and we insert a time interval $I(H)$ into $\mathcal{S}$ with starting edge (v, w') and ending edge (v, u').

3.3 Data Structure

We can implement $\mathcal{S}$ using any standard balanced binary search tree, the only requirement is that, given a (representation of) set G in a data structure $\mathcal{T}_e$, we can efficiently find its corresponding interval in $\mathcal{S}$.

The data structure $\mathcal{T}_e$. We need a data structure $\mathcal{T} = \mathcal{T}_e$ that supports FILTER, INSERT, DELETE, MERGE, CONTAINS, and HASSUPERSET efficiently. We describe a structure of size $O(n)$, that supports CONTAINS and HASSUPERSET in $O(\log n)$ time, FILTER in $O(n)$ time, and INSERT and DELETE in amortized $O(\log^2 n)$ time. In general, answering CONTAINS and HASSUPERSET queries in a dynamic setting is hard and may require $O(n^2)$ space [24].

▶ **Lemma 11.** *Let G and H be two non-empty ε-groups that both end at time t. We have:*

$$(G \cap H \neq \emptyset \wedge |G| \leq |H|) \iff G \subseteq H \wedge G \neq \emptyset.$$

We implement $\mathcal{T}$ with a tree similar to the *grouping-tree* used by Buchin et al. [9]. Let $\{G_1, .., G_k\}$ denote the groups stored in $\mathcal{T}$, and let $\mathcal{X}' = \bigcup_{i \in [1,..,k]} G_i$ denote the entities in these groups. Our tree $\mathcal{T}$ has a leaf for every entity in $\mathcal{X}'$. Each group G_i is represented by an internal node v_i. For each internal node v_i the set of leaves in the subtree rooted at v_i corresponds exactly to the entities in G_i. By Lemma 11 these sets indeed form a tree. With each node v_i, we store the size of the group G_i, and (a pointer to) an arbitrary entity in G_i. Next to the tree we store an array containing for each entity a pointer to the leaf in the tree that represents it (or NIL if the entity does not occur in any group). We preprocess $\mathcal{T}$ in $O(n)$ time to support level-ancestor (LA) queries as well as lowest common ancestor (LCA) queries, using the methods of Bender and Farach-Colton [4, 5]. Both methods work only for *static* trees, whereas we need to allow updates to $\mathcal{T}$ as well. However, as we need to query $\mathcal{T}_e$ only when processing the upper end vertex of e, we can be lazy in updating $\mathcal{T}_e$. More specifically, we delay all updates, and simply rebuild $\mathcal{T}_e$ when we handle its upper end vertex.

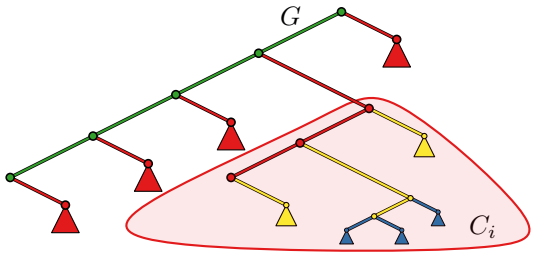


Figure 5 $\mathcal{T}$ is built top-down in several rounds. Edges and nodes are colored by round.

HasSuperSet and Contains queries. Using LA queries we can do a binary search on the ancestors of a given node. This allows us to implement both HasSuperSet($\mathcal{T}_e, G$) queries and Contains($\mathcal{T}_e, G$) in $O(\log n)$ time for a group G ending or starting on edge e. Let a be an arbitrary element from group G. If the datastructure $\mathcal{T}_e$ contains a node matching the elements in G then it must be an ancestor of the leaf containing a in $\mathcal{T}$. That is, it is the ancestor that has exactly $|G|$ elements. By Lemma 11 there is at most one such node. As ancestors only get more elements as we move up the tree, we find this node in $O(\log n)$ time by binary search. Similarly, we can implement the HasSuperSet function in $O(\log n)$ time.

Insert, Delete, and Merge queries. The Insert, Delete, and Merge operations on $\mathcal{T}_e$ are performed lazily; We execute them only when we get to the upper vertex of edge e. At such a time we may have to process a batch of $O(n)$ such operations. We now show that we can handle such a batch in $O(n \log^2 n)$ time.

▶ **Lemma 12.** *Let $G_1,..,G_m$ be maximal ε-groups, ordered by decreasing size, such that: (i) all groups end at time t, (ii) $G_1 \supseteq G_i$, for all i, (iii) the largest group G_1 has size s, and (iv) the smallest group has size $|G_m| > s/2$. We then have that $G_i \supseteq G_{i+1}$ for all $i \in [1,..,m-1]$.*

▶ **Lemma 13.** *Given two nodes $v_G \in \mathcal{T}$ and $v_H \in \mathcal{T}'$, representing the set G respectively H, both ending at time t, we can test if $G \subseteq H$ in $O(1)$ time.*

▶ **Lemma 14.** *Given $m = O(n)$ nodes representing maximal ε-groups $G_1,..,G_m$, possibly in different data structures $\mathcal{T}_1,..,\mathcal{T}_m$, that all share ending time t, we can construct a new data structure $\mathcal{T}$ representing $G_1,..,G_m$ in $O(n \log^2 n)$ time.*

Proof. Sort the groups $G_1,..,G_m$ on decreasing group size. Let $G_1 \in \mathcal{T}_1$ denote the largest group and let it have size s. We assume for now that G_1 is a superset of all other groups. If this is not the case we add a dummy group G_0 containing all elements. We process the groups in order of decreasing size. By Lemma 12 it follows that all groups $G_1,..,G_k$ that are larger than $s/2$ form a path P in $\mathcal{T}$, rooted at G.

For all remaining (small) groups G_i we then find the smallest group in P that is a super set of G_i. By Lemma 13, we can test in $O(1)$ time if a group $H \in P$ is a supergroup of G_i by performing a LCA query in the tree H originated from. We can then find the smallest super set of G_i in $O(\log n)$ time using a binary search. Once all groups are partitioned into clusters with the same ancestor G_i, we process the clusters recursively. When the largest group in a cluster has size one we are done (see Figure 5).

The algorithm goes through a series of rounds. In each round the remaining clusters are handled recursively. Because all (unhandled) clusters jointly contain no more than $O(n)$ groups, each round takes only $O(n \log n)$ time in total. As in each round the size of the largest group left is reduced by half, it follows that after $O(\log n)$ rounds the algorithm must

has constructed the complete tree. Updating the array with pointers to the leaves takes $O(n)$ time, as does rebuilding the tree for future LA and LCA queries. ◀

The final function FILTER can easily be implemented in linear time by pruning the tree from the bottom up. We thus conclude:

▶ **Lemma 15.** *We can handle each event in $O(n \log^2 n)$ time.*

3.4 Maximal Groups

Reconstructing the grouping polygons. Given a group G, represented by a pointer to the top edge of Q_G in $\mathcal{L}$, we can construct the complete group polygon Q_G in $O(|Q_G|)$ time, and list all group members of G in $O(|G|)$ time. We have access to the top edge of Q_G. This is an interval $I(G, \hat{\varepsilon})$ in $\mathcal{S}$, specifically, the version corresponding to $\hat{\varepsilon}$, where $\hat{\varepsilon}$ is the value at which G dies as a maximal group. We then follow the pointers to the previous versions of $I(G, \cdot)$ to construct the left and right chains of Q_G. When we encounter the value $\check{\varepsilon}$ at which G is born, these chains either meet at the same vertex, or we add the final bottom edge of Q_G connecting them. To report the group members of G, we follow the pointer to $I(G, \hat{\varepsilon})$ in $\mathcal{S}$. This interval stores a pointer to its starting edge e, and to a subtree in $\mathcal{T}_e$ of which the leaves represent the entities in G.

Analysis. The list $\mathcal{L}$ contains $O(g) = O(|\mathcal{A}|n^2)$ entries (Theorem 8), each of constant size. The total size of all $\mathcal{S}$'s is $O(|\mathcal{H}|n)$: at each vertex of $\mathcal{H}$, there are only a linear number of changes in the intervals in $\mathcal{S}$. Each edge e of $\mathcal{H}$ stores a data structure $\mathcal{T}_e$ of size $O(n)$. It follows that our representation uses a total of $O(|\mathcal{H}|n) = O(|\mathcal{A}|n^2)$ space. Handling each of the $O(|\mathcal{H}|)$ nodes requires $O(n \log^2 n)$ time, so the total running time is $O(|\mathcal{A}|n^2 \log^2 n)$.

▶ **Theorem 16.** *Given a set $\mathcal{X}$ of n entities, in which each entity travels along a trajectory of τ edges, we can compute a representation of all $g = O(|\mathcal{A}|n^2)$ combinatorial maximal groups $\mathcal{G}$ such that for each group in $\mathcal{G}$ we can report its grouping polygon and its members in time linear in its complexity and size, respectively. The representation has size $O(|\mathcal{A}|n^2)$ and takes $O(|\mathcal{A}|n^2 \log^2 n)$ time to compute, where $|\mathcal{A}| = O(\tau n^2)$ is the complexity of the trajectory arrangement.*

4 Data Structures for Maximal Group Queries

In this section we present data structures that allow us to efficiently obtain all groups for a given set of parameter values. Throughout this section, n denotes the number of entities considered, τ the number of vertices in any trajectory, k the output complexity, i.e. the number of groups reported, g the number of maximal groups, g' the maximum number of maximal groups for a given (fixed) value of ε, and Π the total complexity of the regions corresponding to the g combinatorially different maximal groups. So we have $g' = O(\tau n^3)$ and $g \leq \Pi = O(\tau n^4)$. When g', g, or Π appear as the argument of a logarithm, we write $O(\log n\tau)$. We show that we can store all groups in a data structure of size $O(\Pi \log n\tau \log n)$ that can be built in $O(\Pi \log^2 n\tau \log n)$ time, and allows reporting all (m, ε, δ)-groups in $O(\log^2 n\tau \log n + k)$ time. We use the following three-level tree to achieve this.

On the first level we have a balanced binary tree with in the leaves the group sizes $1...n$. Each internal node v corresponds to a range R_v of group sizes and stores all groups whose size lies in the range R_v. Let $\mathcal{G}_v$ denote this set of groups, and for each such group let D_G denote the duration of group G as a function of ε. The functions D_G are piecewise-linear,

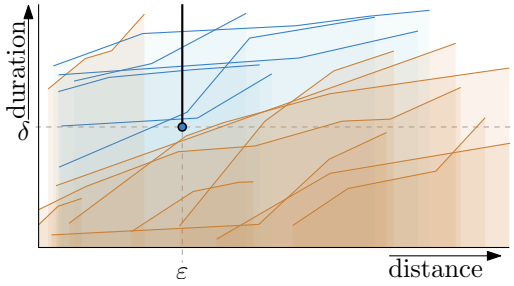


Figure 6 The functions D_G expressing the duration of group G as a function of ε. Assuming all groups have size at least m, all (m, ε, δ)-groups intersect the upward vertical half-ray starting in point (ε, δ).

δ-monotone, and may intersect (see Figure 6). By Theorem 8 the total complexity of these functions is $O(\Pi)$. We store all functions D_G, with $G \in \mathcal{G}_v$, in a data structure that can answer the following *polyline stabbing* queries in $O(\log^2 n\tau + k)$ time: Given a query point $q = (\varepsilon, \delta)$, report all polylines that pass above point q, that is, for which $D_G(\varepsilon) \geq \delta$. Thus, given parameters m, ε, and δ, finding all (m, ε, δ)-groups takes $O(\log^2 n\tau \log n + k)$ time.

We build a segment tree storing the (ε-extent of the) individual edges of all polylines stored at v. An internal node u of the segment tree corresponds to an interval $I(u)$, and stores the set of edges $Ints(u)$ that completely span $I(u)$. Hence, with respect to u, we can consider these segments as lines. For a query with a point q, we have to be able to report all (possibly intersecting) lines from $Ints(u)$ that pass above q. We use a duality transform to map each line ℓ to a point ℓ^* and query point q to a line q^*. The problem is then to report all points ℓ^* in the half-plane below q^*. Such queries can be answered in $O(\log h + k)$ time, using $O(h)$ space and $O(h \log h)$ preprocessing time, where h is the number of points stored [11]. It follows that we can find all k polylines that pass above q in $O(\log^2 n\tau + k)$ time, using $O(\Pi \log n\tau)$ space, and $O(\Pi \log^2 n\tau)$ preprocessing time. We conclude:

▶ **Theorem 17.** *Given parameters m, ε, and δ, we can build a data structure of size $O(\Pi \log n\tau \log n)$, using $O(\Pi \log^2 n\tau \log n)$ preprocessing time, which can report all (m, ε, δ)-groups in $O(\log^2 n\tau \log n + k)$ time, where k is the output complexity.*

In the full version [22], we also describe data structures for the interactive exploration of the data. Here, we have all (m, ε, δ)-groups, for some parameters m, ε, and δ, and we want to change (some of) the parameters. Say to m', ε', and δ', respectively. This requires us to solve *symmetric difference queries*, in which we want to efficiently report all maximal (m, ε, δ)-groups that are no longer maximal for parameters m', ε', and δ', and all maximal $(m', \varepsilon', \delta')$-groups that were not maximal for parameters m, ε, and δ. That is, we wish to report $\mathcal{G}(m, \varepsilon, \delta) \,\Delta\, \mathcal{G}(m', \varepsilon', \delta')$. The following theorem summarizes our results.

▶ **Theorem 18.** *Let (m, ε, δ) and $(m', \varepsilon', \delta')$ be two configurations of parameters. In $O(P(\Pi, g', n))$ time we can build a data structure of size $O(S(\Pi, g', n))$ for symmetric difference queries, that is, we can report all groups in $\mathcal{G}(\varepsilon, m, \delta) \,\Delta\, \mathcal{G}(\varepsilon', m', \delta')$, in $O(Q(\Pi, g', n, k))$ time. In these results Π denotes the total complexity of all combinatorially different maximal groups (over all values ε), g' the number of maximal groups for a fixed value ε, n the number of entities, τ the number of vertices in a trajectory, and k the output complexity. The functions P, S, and Q depend on which of the parameters are allowed to change (other parameters are assumed to be fixed and known at preprocessing time). We have*

Variable Param.	*Query time* $Q(\Pi, g', n, k)$	*Space* $S(\Pi, g', n)$	*Preproc.* $P(\Pi, g', n)$
Changing one parameter at a time			
δ	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε	$\log n\tau + k$	$\Pi \log n\tau$	$\Pi \log n\tau$
δ, m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε, m	$\log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^2 n\tau$
ε, δ	$\sqrt{g'} 2^{\log^* n\tau} \log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
ε, δ, m	$\sqrt{g'} 2^{\log^* n\tau} \log^2 n\tau \log n + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
Changing multiple parameters at the same time			
δ, m	$\log n\tau + k$	$g' \log n\tau$	$g' \log n\tau$
ε, m	$\log^2 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^2 n\tau$
ε, δ	$\sqrt{g'} 2^{\log^* n\tau} \log^3 n\tau + k$	$\Pi \log^2 n\tau$	$\Pi \log^3 n\tau$
ε, δ, m	$\sqrt{g'} 2^{\log^* n\tau} \log^3 n\tau \log^2 n + k$	$\Pi \log^2 n\tau \log^2 n$	$\Pi \log^3 n\tau \log^2 n$

5 Entities Moving in $\mathbb{R}^d$

We now describe how our results can be extended to entities moving in $\mathbb{R}^d$, for any constant dimension d.

5.1 Bounding the Complexity

Recall that $\mathcal{X}(t)$ denotes the locations of the entities at time t. We still consider the (t, ε)-plane in $\mathbb{P}$, and the regions A_G, for subsets $G \subseteq \mathcal{X}$, in which set G is alive. Such a region is still bounded from below by a function f_G that expresses the minimum distance for which G is ε-connected. We again consider the arrangement $\mathcal{H}$ of these functions f_G, over all sets G.

Let $\mathcal{H}''$ be the arrangement of all pairwise distance functions $h_{ab}(t) = \|a(t)b(t)\|$. For any subset of entities G and any time t, $f_G(t) = \|a(t)b(t)\|$ for some pair of entities a and b. Thus, $\mathcal{H}$ is a sub-arrangement of $\mathcal{H}''$. This immediately gives an $O(\tau n^4)$ bound on the complexity of $\mathcal{H}$. We instead show that $\mathcal{H}$ has complexity at most $O(\tau n^3 \beta_4(n))$, where $\beta_s(n) = \lambda_s(n)/n$, and $\lambda_s(n)$ is the maximum length of a Davenport-Schinzel sequence of order s on n symbols. Using exactly the same argument as in Lemma 6 we then get a bound of $O(\tau n^4 \beta_4(n))$ on the number of combinatorially different groups.

Let $\mathcal{E}(t)$ be the Euclidean minimum spanning tree (EMST) of the points in $\mathcal{X}(t)$.

▶ **Observation 19.** *A subset of entities $G \subseteq \mathcal{X}$ is ε-connected at time t if and only if for any two entities $p, q \in G$ the longest edge in the Euclidean minimum spanning tree $\mathcal{E}(t)$ on the path between p and q has length at most ε.*

Specifically, let $\mathcal{E}^G(t)$ be the minimum (connected) subtree of $\mathcal{E}(t)$ containing all points in $G(t)$, and let $\check{\varepsilon}_G(t)$ be the length of the longest edge e in $\mathcal{E}^G(t)$ (see Figure 7). We have that G is an ε-group for all $\varepsilon \geq \check{\varepsilon}_G(t)$, and that $f_G(t) = \max_{(a,b)\in\mathcal{E}^G(t)} \|a(t)b(t)\|$.

It follows from Observation 19 that we are interested in the distance function h_{ab} on the time intervals during which (a, b) is part of the EMST. Hence, we need to consider only the arrangement of such partial functions. It is, however, difficult to bound the complexity of the resulting arrangement directly. Instead, we consider h_{ab} only during those time intervals in which (a, b) is an edge in the Yao-graph [23]. Let $\mathcal{H}'$ be the resulting arrangement. Since the EMST is a subgraph of the Yao-graph it follows that $\mathcal{H}$ is a sub-arrangement of $\mathcal{H}'$ [23].

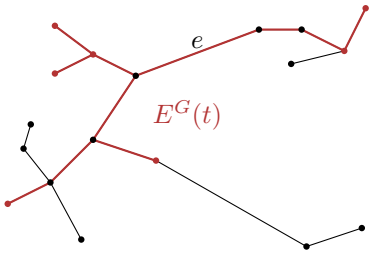


Figure 7 $\mathcal{E}(t)$ and its minimum subtree $E^G(t)$ (red edges) for a subset of entities G (red vertices). Edge e determines the minimum ε for which G is ε-connected at t.

▶ **Lemma 20.** *$\mathcal{H}'$ has complexity $O(\tau n^3 \beta_4(n))$.*

Proof. Fix an entity a, and consider the movement of the other entities with respect to a. This gives us a set of (piecewise linear) trajectories. Entity a is fixed at the origin. Partition this space into $k = O(2^d) = O(1)$ equal size polyhedral cones $C_1,..,C_k$ that have their common apex at the origin. For each such cone C_i, let $\eta_a^i(t)$ denote the distance from a to the nearest entity in the cone. It is easy to show that η_a^i is piecewise hyperbolic, and consists of $O(\tau\lambda_4(n))$ pieces [17].

Let $\mathcal{H}^*$ be the arrangement of all functions η_a^i, over all entities $a \in \mathcal{X}$ and all cones C_i. The total number of pieces (hyperbolic arcs), over all entities and all cones, is $O(\tau n\lambda_4(n))$. Partition time into $O(\tau\lambda_4(n))$ time intervals, with $O(n)$ pieces each. This may require splitting some of the pieces, but the total number of pieces remains $O(\tau n\lambda_4(n))$. In each time interval we now have $O(n)$ hyperbolic arc pieces, that intersect at most $O(n^2)$ times in total. It follows that $\mathcal{H}^*$ has complexity $O(\tau\lambda_4(n)n^2) = O(\tau n^3\beta_4(n))$.

Fix a time t, and consider the graph $Y(t)$ that has an edge (a, b) if and only if b is the nearest neighbor of a in one of the cones C_i at time t, that is, $\|a(t)b(t)\| = \eta_a^i(t)$. Indeed, $Y(t)$ is the Yao-graph of the entities at time t [23]. It follows that $\mathcal{H}^* = \mathcal{H}'$. ◀

Since $\mathcal{H}$ is a sub arrangement of $\mathcal{H}'$, it follows that $\mathcal{H}$ also has complexity $O(\tau n^3\beta_4(n))$. Using exactly the same argument as in Lemma 6 we then get that the number of combinatorially different maximal groups is $O(\tau n^4\beta_4(n))$. We conclude:

▶ **Theorem 21.** *Let $\mathcal{X}$ be a set of n entities, in which each entity travels along a piecewise-linear trajectory of τ edges in $\mathbb{R}^d$, for any constant d. The number of maximal combinatorial groups as well as the total complexity of all their group polygons is at most $O(\tau n^4\beta_4(n))$.*

5.2 Algorithm

We can almost directly apply our algorithm from Section 3 in higher dimensions as well. Instead of the arrangement $\mathcal{H}$, we now use $\mathcal{H}'$. The only differences involve discovering the set entities involved in a BIRTH-event, and splitting the set of entities in case of an EXTEND-event. Let v denote the vertex of $\mathcal{H}'$ we are processing. We use that at time v_t, $\mathcal{H}'$ encodes the Yao-graph Y. Using a breadth first search in Y we can find the entities connected to v. If an edge has length larger than ε we stop the exploration along it. Since Y is planar, and has $O(n)$ vertices this takes $O(n)$ time. This does not affect the running time, hence we get the same result as in Theorem 16 for entities moving in $\mathbb{R}^d$, for any constant d.

5.3 Data Structures

Finding all maximal (m, ε, δ)-groups. We use the same approach as in Section 4. However, the functions $duration_G$ are no longer (piecewise) linear functions in ε. Let $start_G(\varepsilon) = f^{-1}(\varepsilon)$ and $end_G(\varepsilon) = h^{-1}(\varepsilon)$ be some hyperbolic functions f and h corresponding to curves in $\mathcal{H}$. We have that $duration_G(\varepsilon) = end_G(\varepsilon) - start_G(\varepsilon)$. The function $duration_G$ corresponds to a piecewise curve with pieces defined by polynomials of degree at most four. Hence, we have to solve the following sub-problem: given a set of g' algebraic curves of degree at most four, and query point q, report all curves that pass above q.

We can solve such queries as follows. We transform the curves into hyperplanes in $\mathbb{R}^\ell$, where ℓ is the linearization dimension. We then apply a duality transform, after which the problem can be solved using a half-space range query. Since we have curves of degree at most four in $\mathbb{R}^2$, the linearization dimension is seven: the set of points above a curve can be described using a seven-variate polynomial (the five coefficients of the degree four curve, and the two coordinates of the point) [1]. It follows that we can find all curves above query point q in $O(g'^{1-1/\lfloor 7/2 \rfloor} \operatorname{polylog} n\tau + k) = O(g'^{2/3} \operatorname{polylog} n\tau + k)$ time, using linear space [19]. Reporting all maximal (m, ε, δ)-groups thus takes $O(g'^{2/3} \operatorname{polylog} n\tau + k)$ time, using $O(\Pi \log n\tau \log n)$ space and $O(\Pi \log^2 n\tau \log n)$ preprocessing time.

Alternatively, we can maintain the upper envelope of the curves in a dynamic data structure. To solve a query, we repeatedly delete the curve realizing the upper envelope at q_ε. This allows us to find all (m, ε, δ)-groups in $O(k\beta_4(g') \operatorname{polylog} n\tau)$ time [10].

Symmetric Difference Queries. Only the versions of the problem that involve changing both ε and δ are affected. Instead of piecewise linear functions $duration_G$ we again have piecewise curves of degree at most four. We use a similar approach as above to find the curves that intersect a vertical or horizontal query segment in $O(g'^{2/3} \operatorname{polylog} n\tau + k)$ time. Thus, we essentially replace the $\sqrt{g'}$ terms in Theorem 18 by a $g'^{2/3}$ term.

References

1 P. Agarwal and J. Matoušek. On Range Searching with Semialgebraic Sets. *Disc. & Comput. Geom.*, 11(4):393–418, 1994.

2 N. Amato, M. Goodrich, and E. Ramos. Computing the arrangement of curve segments: Divide-and-conquer algorithms via sampling. In *Proc. 11th ACM-SIAM Symp. on Disc. Algorithms*, pages 705–706, 2000.

3 G. Andrienko, N. Andrienko, and S. Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46, 2007.

4 M. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN 2000: Theoret. Informatics*, volume 1776 of *LNCS*, pages 88–94. Springer, 2000.

5 M. Bender and M. Farach-Colton. The level ancestor problem simplified. *Theoret. Computer Science*, 321(1):5–12, 2004.

6 M. Benkert, B. Djordjevic, J. Gudmundsson, and T. Wolle. Finding popular places. *Int. J. of Comput. Geom. & Appl.*, 20(1):19–42, 2010.

7 P. Bovet and S. Benhamou. Spatial analysis of animals' movements using a correlated random walk model. *J. of Theoret. Biology*, 131(4):419–433, 1988.

8 K. Buchin, M. Buchin, M. van Kreveld, M. Löffler, R. Silveira, C. Wenk, and L. Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

9 K. Buchin, M. Buchin, M. van Kreveld, B. Speckmann, and F. Staals. Trajectory grouping structure. *J. of Comput. Geom.*, 6(1):75–98, 2015.

10 T. Chan. A dynamic data structure for 3-d convex hulls and 2-d nearest neighbor queries. *J. of the ACM*, 57(3):16:1–16:15, 2010.

11 B. Chazelle, L. Guibas, and D. Lee. The power of geometric duality. *BIT Numerical Mathematics*, 25(1):76–90, 1985.

12 D. Eppstein, M. Goodrich, and J. Simons. Set-difference range queries. In *Proc. 2013 Canadian Conf. on Comput. Geom.*, 2013.

13 A. Fujimura and K. Sugihara. Geometric analysis and quantitative evaluation of sport teamwork. *Systems and Computers in Japan*, 36(6):49–58, 2005.

14 J. Gudmundsson, M. van Kreveld, and B. Speckmann. Efficient detection of patterns in 2D trajectories of moving points. *GeoInformatica*, 11:195–215, 2007.

15 J. Gudmundsson, M. van Kreveld, and F. Staals. Algorithms for hotspot computation on trajectory data. In *Proc. 21st ACM SIGSPATIAL GIS*, pages 134–143, 2013.

16 D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual analytics: Definition, process, and challenges. In A. Kerren, J. Stasko, J.-D. Fekete, and C. North, editors, *Information Visualization*, volume 4950 of *LNCS*, pages 154–175. Springer, 2008.

17 I. Kostitsyna, M. van Kreveld, M. Löffler, B. Speckmann, and F. Staals. Trajectory grouping structure under geodesic distance. In *Proc. 31th Symp. Computat. Geom.* Lipics, 2015.

18 X. Li, X. Li, D. Tang, and X. Xu. Deriving features of traffic flow around an intersection from trajectories of vehicles. In *Proc. IEEE 18th Int. Conf. Geoinformatics*, pages 1–5, 2010.

19 J. Matoušek. Efficient partition trees. *Disc. & Comput. Geom.*, 8(3):315–334, 1992.

20 M. Mirzargar, R. Whitaker, and R. Kirby. Curve Boxplot: generalization of Boxplot for ensembles of curves. *IEEE Trans. on Vis. and Comp. Graphics*, 20(12):2654–2663, 2014.

21 A. Stohl. Computation, accuracy and applications of trajectories – a review and bibliography. *Atmospheric Environment*, 32(6):947–966, 1998.

22 Arthur van Goethem, Marc van Kreveld, Maarten Löffler, Bettina Speckmann, and Frank Staals. Grouping time-varying data for interactive exploration. *CoRR*, abs/1603.06252, 2016.

23 A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.

24 D. Yellin. Representing sets with constant time equality testing. *J. of Algorithms*, 13(3):353–373, 1992.

On the Separability of Stochastic Geometric Objects, with Applications

Jie Xue[1], Yuan Li[2], and Ravi Janardan[3]

1 **Department of Computer Science and Engineering, University of Minnesota – Twin Cities, 4-192 Keller Hall, 200 Union St. SE, Minneapolis, MN 55455, USA**
xuexx193@umn.edu

2 **Department of Computer Science and Engineering, University of Minnesota – Twin Cities, 4-192 Keller Hall, 200 Union St. SE, Minneapolis, MN 55455, USA**
lixx2100@umn.edu

3 **Department of Computer Science and Engineering, University of Minnesota – Twin Cities, 4-192 Keller Hall, 200 Union St. SE, Minneapolis, MN 55455, USA**
janardan@umn.edu

Abstract

In this paper, we study the linear separability problem for stochastic geometric objects under the well-known unipoint/multipoint uncertainty models. Let $S = S_R \cup S_B$ be a given set of stochastic bichromatic points, and define $n = \min\{|S_R|, |S_B|\}$ and $N = \max\{|S_R|, |S_B|\}$. We show that the *separable-probability* (SP) of S can be computed in $O(nN^{d-1})$ time for $d \geq 3$ and $O(\min\{nN \log N, N^2\})$ time for $d = 2$, while the *expected separation-margin* (ESM) of S can be computed in $O(nN^d)$ time for $d \geq 2$. In addition, we give an $\Omega(nN^{d-1})$ *witness-based lower bound* for computing SP, which implies the optimality of our algorithm among all those in this category. Also, a hardness result for computing ESM is given to show the difficulty of further improving our algorithm. As an extension, we generalize the same problems from points to general geometric objects, i.e., polytopes and/or balls, and extend our algorithms to solve the generalized SP and ESM problems in $O(nN^d)$ and $O(nN^{d+1})$ time, respectively. Finally, we present some applications of our algorithms to stochastic convex-hull related problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems - Geometrical problems and computations

Keywords and phrases Stochastic objects, linear separability, separable-probability, expected separation-margin, convex hull

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.62

1 Introduction

Linear separability describes the property that a set of d-dimensional bichromatic (red and blue) points can be separated by a hyperplane such that all the red points lie on one side of the hyperplane and all the blue points lie on the other side. This problem has been well studied for years in computational geometry, and is widely used in machine learning and data mining for data classification. However, existing linear-separation algorithms require that all the input points must have fixed locations, which is rarely true in reality due to imprecise sampling from GPS, robotics sensors, or some other probabilistic systems. It is therefore essential to study the conventional linear separability problem under uncertainty.


© Jie Xue, Yuan Li, and Ravi Janardan;
licensed under Creative Commons License CC-BY

In this paper, we study the linear separability problem under two different uncertainty models, i.e., the *unipoint* and *multipoint* models [4]. In the former, each stochastic data point has a fixed and known location, and has a positive probability to exist at that location; whereas in the latter, each stochastic data point occurs in one of discretely-many possible locations with known probabilities, and the existence probabilities of each point sum up to at most 1 to allow for its absence. Our focus is to compute the *separable-probability* (SP) and the *expected separation-margin* (ESM) for a given set of bichromatic stochastic points (or general geometric objects) in $\mathbb{R}^d$ for $d \geq 2$, where the former is the probability that the existent points (or objects) are linearly separable, and the latter is the expectation of the separation-margin of the existent points (or objects). (See Section 3.1 for a detailed and formal definition of the latter.)

The main contributions of this paper are:

1. We propose an $O(nN^{d-1})$-time algorithm, which uses linear space, for solving the SP problem when given a set of bichromatic stochastic points in $\mathbb{R}^d$ for $d \geq 3$. (The runtime is $O(\min\{N^2, nN \log N\})$ for $d = 2$.) We also show an $\Omega(nN^{d-1})$ lower bound for all witness-based algorithms, which implies the optimality of our algorithm among all witness-based methods for $d \geq 3$. (See Section 2.)
2. We show that the ESM of the above dataset can be computed in $O(nN^d)$ time for $d \geq 2$, using linear space. A hardness result is also given to show the total number of distinct possible separation-margins is $\Theta(nN^d)$, which implies that it may be difficult to achieve a better runtime. (See Section 3.)
3. We extend our algorithms to compute the SP and the ESM for datasets containing general stochastic geometric objects, such as polytopes and/or balls. Our generalized algorithms solve the former problem in $O(nN^d)$ time, and the latter in $O(nN^{d+1})$ time, using linear space. (See Section 4.)
4. We provide some applications of our algorithms to stochastic convex-hull (SCH) related problems. Specifically, by taking advantage of our SP algorithm, we give a method to compute the SCH membership probability, which matches the best known bound but is more direct. Also, we consider some generalized versions of this problem and show how to apply our separability algorithms to solve them efficiently. (See Section 5.)

Due to space limitations, most proofs and some details are omitted here, but can be found in the full version [18].

1.1 Related work

The study of computational geometry problems under uncertainty is a relatively new topic, and has attracted a lot of attention; see [5] and [7] for two surveys. Different uncertainty models and related problems have been investigated in recent years. See [1, 2, 3, 8, 14, 15, 19] for example. The unipoint/multipoint uncertainty model, which we use in this paper, was first defined in [4, 17], and has been applied in many recent papers. For instance, in [12], Kamousi et al. studied the stochastic minimum spanning tree problem, and computed its expected length. Suri et al. investigated the most likely convex hull problem over uncertain data in [17]; the similar topic was revisited by Agarwal et al. in [4] to compute the probability that a query point is inside the uncertain hull. In [16], Suri and Verbeek studied the most likely Voronoi Diagram (LVD) in $\mathbb{R}^1$ under the unipoint model, and the expected complexity of LVD was further improved by Li et al. in [13], who explored the stochastic line arrangement problem in $\mathbb{R}^2$. In [6], Agrawal et al. proposed efficient algorithms for the most likely skyline problem in $\mathbb{R}^2$ and gave NP-hardness results in higher dimensions.

Recently, in [8], de Berg et al. studied the separability problem given a set of bichromatic imprecise points in $\mathbb{R}^2$ in a setting that each point is drawn from an imprecision region.

Very recently, in an unpublished manuscript [11], one of our proposed problems, the SP computing problem, was independently studied by Fink et al. under the same uncertainty model, and similar results were obtained, i.e., an $O((n+N)^d) = O(N^d)$-time and $O(n+N) = O(N)$-space algorithm for computing the SP of a given set of bichromatic stochastic points in $\mathbb{R}^d$. In fact, before the final step of the improvement, the time bound achieved was $O(N^d \log N)$, and then *duality* [9] and *topological sweep* [10] techniques were applied to eliminate the log factor. On the other hand, our algorithm runs initially in $O(nN^{d-1} \log N)$ time using linear space, and can be further improved to $O(nN^{d-1})$ runtime by using the same techniques. (A careful discussion will be given in Section 2.) Note that the algorithm in [11] always runs in $\Theta(N^d)$ time (no matter how small n is) and, more importantly, this runtime appears to be intrinsic: all possible d-tuples of (distinct) points have to be enumerated in order to correctly compute the SP. Our time bound matches the bound in [11] when $n = \Theta(N)$. However, when $n \ll N$, our method is much faster, and, in fact, this is usually the case in many real classification problems in machine learning and data mining.

In terms of how to solve the problem, Fink et al.'s method is very different from ours. Their computation of SP relies on an additional dummy anchor point, and based on this point, the probability is computed in an inclusion-exclusion manner. On the other hand, our method solves the problem more directly: it does not introduce any additional points and the SP is eventually computed using a simple addition principle. Furthermore, our algorithm can be easily extended to solve many generalized problems (e.g., multiple colors, general geometric objects, etc.)

1.2 Basic notations and preliminaries

Throughout this paper, the basic notations we use are the following. We use $S = S_R \cup S_B$ to denote the given stochastic bichromatic dataset, where S_R (resp. S_B) is a set of red (resp. blue) stochastic points (or general geometric objects in Section 4). The notations n and N are used to denote the sizes of the smaller and larger classes of S respectively, i.e., $n = \min\{|S_R|, |S_B|\}$ and $N = \max\{|S_R|, |S_B|\}$, and d is used to denote the dimension. In this paper, we always assume that d is a constant. When we need to denote a normal bichromatic dataset (without considering the existence probabilities), we usually use the notation $T = T_R \cup T_B$. The coordinates of a point $x \in \mathbb{R}^d$ are denoted as $x^{(1)}, x^{(2)}, \ldots, x^{(d)}$. If T is a dataset in $\mathbb{R}^d$ and U is some linear subspace of $\mathbb{R}^d$, we use T^U to denote a new dataset in the space U, which is obtained by orthogonally projecting T from $\mathbb{R}^d$ onto U.

We say a set of bichromatic points is *strongly separable* iff there exists a hyperplane, h, so that all the red points strictly lie on one side of h while all the blue points strictly lie on the other side. Also, we can define the concept of *weakly separable* similarly, except that we allow points to lie on the hyperplane. A hyperplane that strongly (resp., weakly) separates a set of bichromatic points is called a *strong* (resp., *weak*) *separator*. The following is a fundamental result we will use in various places of the paper.

▶ **Theorem 1.** *Suppose $T = T_R \cup T_B$ is a set of bichromatic points in $\mathbb{R}^d$. T is strongly separable by a hyperplane iff $\mathcal{CH}(T_R) \cap \mathcal{CH}(T_B) = \emptyset$, where $\mathcal{CH}(\cdot)$ denote the convex hull of the indicated point-set.*

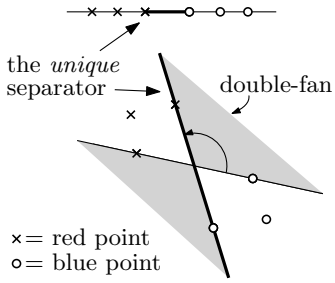


Figure 1 Illustrating the unique separator we choose for separable instances in $\mathbb{R}^1$ and $\mathbb{R}^2$.

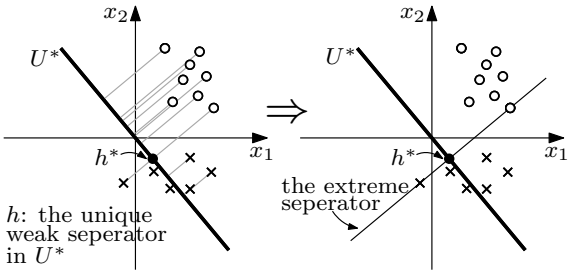


Figure 2 Illustrating the extreme separator in $\mathbb{R}^2$.

2 Separable-probability

The separable-probability (SP) of a stochastic bichromatic dataset $S = S_R \cup S_B$ in $\mathbb{R}^d$ refers to the probability that the existent points in S (obtained by a random experiment) are (strongly) separable. For simplicity, we describe here the details of our algorithm under the unipoint model only. The generalization of our algorithm to the multipoint model is quite straightforward and can be found in the full version [18].

Trivially, given a dataset S, its SP can be computed by simply enumerating all the $2^{|S|}$ possible instances of S and summing up the probabilities of the separable ones, which takes exponential time. In order to solve the problem more efficiently than by brute-force, one has to categorize all the separable instances of S into a reasonable number of groups such that the sum of the probabilities of the instances in each group can be easily computed. A natural approach is to charge each separable instance to a *unique* separator, and use that as the key to do the grouping. The uniqueness requirement here is to avoid over-counting. In addition, all these separators should be easy to enumerate and the sum of the probabilities of those separable instances charged to each separator should be efficiently computable. In $\mathbb{R}^1$ and $\mathbb{R}^2$, this is easy to achieve. For example, in $\mathbb{R}^1$, given a separable instance, all the possible separators form a segment, and we can choose the leftmost endpoint as the unique separator; in $\mathbb{R}^2$, all the possible separators of a separable instance form a double-fan, and we can choose the most counterclockwise one, which goes through exactly one red and one blue point, as the unique separator. (See Figure 1 for an illustration.) It is easy to see that, with the separators chosen above, the SP of the stochastic dataset can be easily computed by considering the sum of the probabilities of the instances charged to each such separator. However, to define such a separator for a separable instance beyond $\mathbb{R}^2$ turns out to be hard and challenging. To solve this problem, we define an important concept called *extreme separator* in Section 2.1, and apply this concept to compute the SP of S in Section 2.2.

For convenience, we assume that the points given in S have the *strong general position* property (SGPP), which is defined as follows. Let $I = \{i_1, i_2, \ldots, i_{|I|}\}$ be any subset of

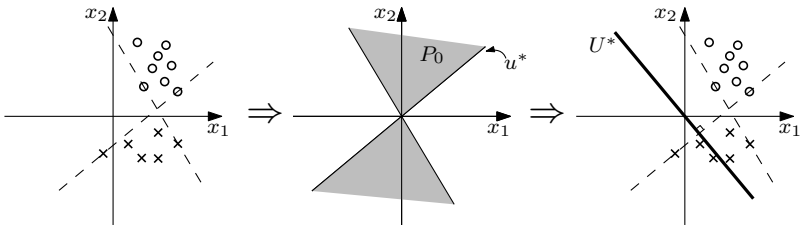


Figure 3 Illustrating U^* in $\mathbb{R}^2$. Note that P_1 is not shown to avoid confusion.

the index set $\{1, 2, \ldots, d\}$ where $i_1 < i_2 < \cdots < i_{|I|}$. We define a projection function $\phi_I : \mathbb{R}^d \to \mathbb{R}^{|I|}$ as $\phi_I(x) = (x^{(i_1)}, x^{(i_2)}, \ldots, x^{(i_{|I|})})$. Also, for any $X \subseteq \mathbb{R}^d$, we define $\Phi_I(X) = \{\phi_I(x) : x \in X\}$.

Let T be a set of points in $\mathbb{R}^d$. When $d \leq 2$, we say T has SGPP iff it is in general (linear) position, i.e., affinely independent. When $d \geq 3$, we say T has SGPP iff

1. T is in general (linear) position;
2. $\Phi_J(T)$ has SGPP for $J = \{3, 4, \ldots, d\}$.

▶ Remark. This assumption is actually not stronger than the conventional general position assumption, since one can easily apply linear transformations to make a set of points in general position have SGPP (the separability of a dataset is invariant under non-singular linear transformations). See the full version [18] for a detailed algorithm to achieve this.

2.1 Extreme separator

To solve the SP problem, we define a very important concept called *extreme separator* through a sequence of steps. Suppose a separable bichromatic dataset $T = T_R \cup T_B$ with SGPP is given in $\mathbb{R}^d$ ($d \geq 2$). Assume that both T_R and T_B are nonempty. Let $\mathcal{V}$ be the collection of the $(d-1)$-dim linear subspaces of $\mathbb{R}^d$ whose equation is of the form $ax_1 + bx_2 = 0$, where a and b are constants not equal to 0 simultaneously. In other words, $\mathcal{V}$ contains all the $(d-1)$-dim linear subspaces that are perpendicular to the x_1x_2-plane and go through the origin. Then there is a natural one-to-one correspondence between $\mathcal{V}$ and $\mathbb{P}^1$ (the 1-dim projective space), $\sigma : [ax_1 + bx_2 = 0] \longleftrightarrow [a : b]$. For convenience, we use σ to denote the maps in both directions in the rest of this paper. We now define a map $\pi_T : \mathcal{V} \to \{0, 1\}$ as follows. For any $V \in \mathcal{V}$, we orthogonally project all the points in T onto V and use $T^V = T_R^V \cup T_B^V$ to denote the new dataset after projection. If T^V is strongly separable, we set $\pi_T(V) = 1$. Otherwise, we set $\pi_T(V) = 0$. The map π_T induces another map $\pi_T^* : \mathbb{P}^1 \to \{0, 1\}$ by composing with the correspondence σ. Let P_0 and P_1 be the pre-images of $\{0\}$ and $\{1\}$ under π_T^*, respectively (see Figure 3). By applying Theorem 1, it is easy to prove the following.

▶ **Theorem 2.** *P_0 is a connected closed subspace of $\mathbb{P}^1$. $P_0 = \emptyset$ iff $\Phi_J(T)$ is strongly separable in $\mathbb{R}^{d-2}$ for $J = \{3, 4, \ldots, d\}$.*

We now have two cases, i.e., $P_0 \neq \emptyset$ and $P_0 = \emptyset$. If $P_0 \neq \emptyset$, we define the extreme separator of T as follows. Since P_0 is a connected closed subspace of $\mathbb{P}^1$, it has a unique clockwise boundary point u^* (i.e., u^* is the last point of P_0 in the clockwise direction). Let $U^* = \sigma(u^*)$ be the linear subspace in $\mathcal{V}$ corresponding to u^* (see Figure 3 again). The following theorem reveals the separability property of T^{U^*}.

▶ **Theorem 3.** *T^{U^*} is weakly separable and there exists only one weak separator. Furthermore, the unique separator of T^{U^*} goes through exactly d points, of which at least one is in $T_R^{U^*}$ and one is in $T_B^{U^*}$.*

▶ **Definition 4** (Derived Separator)**.** Let U be a k-dim linear subspace ($k < d$) of $\mathbb{R}^d$. Suppose h is a strong (resp., weak) separator of T^U in the space U. It is easy to see that the pre-image, h', of h under the orthogonal projection $\mathbb{R}^d \to U$ is a strong (resp., weak) separator of T in $\mathbb{R}^d$. We call h' the *derived separator* of h in $\mathbb{R}^d$.

Let h^* be the unique weak separator of T^{U^*}. We define the *extreme separator* of T as the derived separator of h^* in $\mathbb{R}^d$. (See Figure 2.) At the same time, we call U^* the *auxiliary subspace* defining the extreme separator. Clearly, the extreme separator and the auxiliary subspace are perpendicular to each other.

On the other hand, if $P_0 = \emptyset$, we recursively define the extreme separator of T as the derived separator of the extreme separator of $\Phi_J(T)$, for $J = \{3, 4, \ldots, d\}$. Note that P_0 is nonempty when $d = 2$. To complete this recursive definition, we define the extreme separator in $\mathbb{R}^1$ as the weak separator (which is a point) with the smallest coordinate.

Note that the above definition of the extreme separator is only for the case that both T_R and T_B are nonempty. In the trivial case where T_R and/or T_B is empty, we simply define the extreme separator as $x_d = \infty$.

To understand the intuition for the extreme separator, let us consider the case $d = 3$. Imagine there is a plane rotating clockwise around the x_3-axis. We keep projecting the points in T (orthogonally) to that plane and track the separability of the projection images. If the images are always separable, then the extreme separator is defined recursively. Otherwise, there is a closed period of time in which the images are inseparable, which is subsequently followed by an open period in which the images are separable. At the connection of the two periods (from the inseparable one to the separable one), the images are weakly separable by a unique weak separator. Then the rotating plane at this point is just the auxiliary subspace, and the extreme separator is obtained by orthogonally "extending" the unique weak separator to $\mathbb{R}^3$.

2.2 Computing the separable-probability

We remind the reader that $n = \min\{|S_R|, |S_B|\}$ and $N = \max\{|S_R|, |S_B|\}$. Set $J = \{3, 4, \ldots, d\}$. If the existent points in S are separable, then there are two cases: 1) the extreme separator of the existent points is defined recursively (the case of $P_0 = \emptyset$) or equal to $x_d = \infty$ (the trivial case); 2) the extreme separator of the existent points is directly defined in $\mathbb{R}^d$ (the case of $P_0 \neq \emptyset$). These two cases are clearly disjoint so that the SP can be computed as the sum of their probabilities. By applying Theorem 2, the probability of the first case is equal to the SP of $\Phi_J(S)$. In the second case, according to Theorem 3, the extreme separator goes through exactly d points (of which at least one is red and one is blue). Thus, the SP of S can be computed as

$$Sep(S) = Sep(\Phi_J(S)) + \sum_{h \in H_S} \tau_S(h),$$

where H_S is the set of the hyperplanes that go through exactly d points (of which at least one is red and one is blue) in S and, for $h \in H_S$, $\tau_S(h)$ is the probability that the extreme separator of the existent points is h.

Clearly, for each $h \in H_S$, there is a unique element $U^* \in \mathcal{V}$ perpendicular to it (h can never be parallel to the x_1x_2-plane due to the SGPP of S). If h is indeed the extreme separator of the existent points, then U^* must be the auxiliary subspace. Let $E = E_R \cup E_B$ be the set of the points on h. In order to compute $\tau_S(h)$, we investigate the conditions for h to be the extreme separator of the existent points. First, as the d points on h, the points

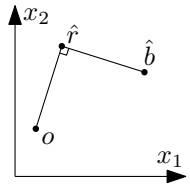


Figure 4 Illustrating the location of o. The space in the figure is the 2-dim subspace of $\mathbb{R}^d$ that is parallel to the x_1x_2-plane and contains $\hat{r}$, $\hat{b}$.

in E must exist. Second, because the existent points should be *weakly* (but not strongly) separable after being projected onto U^*, there must exist $\hat{r} \in \mathcal{CH}(E_R)$ and $\hat{b} \in \mathcal{CH}(E_B)$ whose projection images on U^* coincide, according to Theorem 1. (Actually, such $\hat{r}$ and $\hat{b}$ are unique if they exist, due to the SGPP of S.) Finally, since the extreme separator should weakly separate the existent points, all the red existent points must lie on one side of h while all the blue ones must lie on the other side, except the points in E. Also, the side for red/blue points is specific, as $\sigma(U^*)$ must be the *clockwise* boundary of P_0. To distinguish the red/blue side of h, we define, based on $\hat{r}$ and $\hat{b}$, an indicator $o = (o^{(1)}, o^{(2)}, \ldots, o^{(d)})$, where

$$o^{(1)} = \hat{r}^{(1)} + (\hat{b}^{(2)} - \hat{r}^{(2)}),$$
$$o^{(2)} = \hat{r}^{(2)} + (\hat{r}^{(1)} - \hat{b}^{(1)}),$$
$$o^{(i)} = \hat{r}^{(i)} = \hat{b}^{(i)} \text{ for } i \in J.$$

(See Figure 4 for the location of o.) It is easy to see that, when all the red (resp., blue) points appear on the same (resp., opposite) side of h w.r.t. o, $\sigma(U^*)$ is the *clockwise* boundary of P_0. In sum, h is the extreme separator of the existent points iff

1. all the points in E exist;
2. there are $\hat{r} \in \mathcal{CH}(E_R)$ and $\hat{b} \in \mathcal{CH}(E_B)$ such that their projection images on U^* coincide;
3. no red (resp. blue) point on the opposite (resp. same) side of h w.r.t. o exists.

Among the three conditions, the second one has nothing to do with the existences of the stochastic points in S and can be verified in constant time. If h violates this condition, then $\tau_S(h) = 0$. Otherwise, $\tau_S(h)$ is just equal to the product of the existence probabilities of the points in E and the non-existence probabilities of the points which should not exist due to the third condition. The simplest way to compute it is to scan every point in S once, which takes linear time. This then leads to an $O(nN^d)$ overall time for computing the SP of S, since $|H_S|$ is bounded by $O(nN^{d-1})$.

By applying the idea of of *radial-order sort* in [4], we are able to reduce the runtime to $O(nN^{d-1} \log N)$. Furthermore, inspired by [11], it is possible to further eliminate the log factor in the time bound by taking advantage of *duality* [9] and *topological sweep* [10] techniques. The time complexity of our algorithm is then improved to $O(nN^{d-1})$ for $d \geq 3$ and $O(\min\{nN \log N, N^2\})$ for $d = 2$. See the full version [18] for details.

2.3 Witness-based lower bound for computing separable-probability

When solving the SP problem, the key idea of our algorithm is to group the probabilities of those separable instances that share the same extreme separator so that the SP can be efficiently computed by considering the extreme separators instead of single instances. Actually, by extending and abstracting this idea, we are able to get a general framework for computing SP, which we call the *witness-based framework*. Let S be the given stochastic dataset and $\mathcal{I}_S$ be the set of all the separable instances of S. The witness-based framework for computing the SP of S is the following. (Here, $\mathcal{P}(\cdot)$ denotes the powerset function.)

1. Define a set $W = \{h_1, \ldots, h_m\}$ of hyperplanes (called *witness separators*) with specified weights $w_1, \ldots, w_m$ and an implicitly specified witness rule $f : W \to \mathcal{P}(\mathcal{I}_S)$ such that
   - the instances in $f(h_i)$ are (either strongly or weakly) separated by h_i;
   - the witness probability (see Step **2** below) of each h_i is efficiently computable;
   - any instance $I \in \mathcal{I}_S$ satisfies $\sum_{\forall i(I \in f(h_i))} w_i = 1$.

   We say the witness separator h_i *witnesses* the instances in $f(h_i)$.
2. Compute **efficiently** the *witness probability* of each $h_i \in W$, which is defined as

$$witP(h_i) = \sum_{I \in f(h_i)} Pr(I),$$

   i.e., the sum of the probabilities of all the instances witnessed by h_i.
3. Compute the SP of S by linearly combining the witness probabilities with the specified weights, i.e.,

$$Sep(S) = \sum_{i=1}^{m} (w_i \cdot witP(h_i)) = \sum_{I \in \mathcal{I}_S} Pr(I).$$

Note that the witness-based framework is very general. The ways of defining witness separators and specifying witness rules may vary a lot among different witness-based algorithms. Our algorithm and the one introduced in [11], which are the only two known algorithms for computing SP at this time, both belong to the witness-based framework. Similar frameworks are also used to solve other probability-computing problems. For example, the two algorithms in [4] for computing convex-hull membership probability are both implemented by defining witness edges/facets and summing up the witness probabilities. To our best knowledge, up to now, most probability-computing problems under unipoint/multipoint model are solved by applying ideas close to this framework.

Now we show that any SP computing algorithm following the witness-based framework takes at least $\Omega(nN^{d-1})$ time in the worst case, and thus our algorithm is optimal among this category of algorithms for any $d \geq 3$. Clearly, the runtime of a witness-based algorithm is at least $|W| = m$, i.e., the number of the witness separators. Then a question naturally arises: how many witness separators do we need for computing SP? From the above framework, one restriction for W is that each separable instance of S must be witnessed by at least one witness separator $h_i \in W$, i.e., $\mathcal{I}_S = \bigcup_{i=1}^{m} f(h_i)$. Otherwise, the probabilities of the unwitnessed instances in $\mathcal{I}_S$ will not be counted when computing the SP of S. It then follows that each separable instance of S must be separated by some $h_i \in W$. We prove that, in the worst case, we always need $\Omega(nN^{d-1})$ hyperplanes to separate all the separable instances of S, which implies an $\Omega(nN^{d-1})$ lower bound on the runtime of any witness-based SP computing algorithm. We say a hyperplane set H *covers* a bichromatic dataset $T = T_R \cup T_B$ iff for any non-trivial separable subset $V \subseteq T$ (i.e., V contains at least one red point and one blue point), there exists $h \in H$ that separates V. The following theorem completes the discussion, and is also of independent interest.

▶ **Theorem 5.** *For a bichromatic dataset T, define $\chi(T)$ to be the size of the smallest hyperplane set that covers T. Let $\mathcal{T}^d_{n,N}$ be the collection of all the bichromatic datasets in $\mathbb{R}^d$ containing n red points and N blue points ($n \leq N$). Define*

$$\Gamma_d(n, N) = \sup_{T \in \mathcal{T}^d_{n,N}} \chi(T).$$

Then for any constant d, we have $\Gamma_d(n, N) = \Omega(nN^{d-1})$.

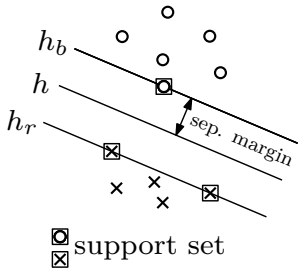


Figure 5 An example in $\mathbb{R}^2$.

3 Expected separation-margin

In this section, we discuss how to compute the expected separation-margin (ESM) of a stochastic dataset $S = S_R \cup S_B$. Again, we only describe the details of our algorithm under the unipoint model. The generalization to the multipoint model is straightforward and can be found in the full version [18]. We assume that S has (conventional) general position property.

3.1 Definitions

Let $T = T_R \cup T_B$ be a separable bichromatic dataset and h be a separator. We define the margin of h w.r.t. T as $M_h(T) = \min_{a \in T} dist(a, h)$. The separator which maximizes the margin is called the *maximum-margin separator* and the corresponding margin is called the *separation-margin* of T, denoted by $Mar(T)$. If T is not separable or if $T_R = \emptyset$ or $T_B = \emptyset$, we define its separation-margin to be 0 for convenience. The ESM of a stochastic dataset $S = S_R \cup S_B$ is the expectation of the separation-margin of the existent points.

▶ **Theorem 6.** *For any separable dataset $T = T_R \cup T_B$ with $T_R \neq \emptyset$ and $T_B \neq \emptyset$, the maximum-margin separator of T is unique. Furthermore, for any closest pair (r, b) where $r \in \mathcal{CH}(T_R)$ and $b \in \mathcal{CH}(T_B)$, the maximum-margin separator of T is the bisector of the segment $\overline{rb}$.*

Let h be the maximum-margin separator of T and $M = Mar(T)$ be its separation-margin. Define $C_R = \{r \in T_R : dist(r, h) = M\}$ and $C_B = \{b \in T_B : dist(b, h) = M\}$. We call $C = C_R \cup C_B$ the *support set* of T and the points in it the *support points*. All the support points have the same distance to the maximum-margin separator. Thus, there should exist two parallel hyperplanes h_r and h_b (both parallel to the maximum-margin separator) where h_r goes through all the red support points and h_b goes through all the blue ones. We call h_r and h_b the *support planes* of T. Including the maximum-margin separator h, they form a group of three parallel and equidistant hyperplanes (h_r, h, h_b). (See Figure 5.) Since the maximum-margin separator is unique, the support set and support planes are also unique. We shall show that the maximum-margin separator can be uniquely determined via the support set.

▶ **Theorem 7.** *Suppose C is the support set of T. Then T and C share the same maximum-margin separator (also the same separation-margin) and the support set of C is just itself.*

3.2 Computing the expected separation-margin

According to Theorem 7, the separation-margin of a separable dataset is equal to that of its support set. Thus, the ESM of S can be computed as

$$Emar(S) = \sum_{C \subseteq S} (\xi_S(C) \cdot Mar(C)),$$

where $\xi_S(C)$ is the probability that the existent points in S are separable with the support set C. Since S has the general position property, the size of the support set of the existent points can be at most $2d$ (d red points and d blue points at most). It follows that the total number of the possible C to be considered is bounded by $O(n^d N^d)$. Indeed, we can further improve this bound.

▶ **Theorem 8.** *For a given stochastic dataset S with general position property, the total number of the possible support sets is bounded by $O(nN^d)$. As a result, the number of the (distinct) possible separation-margins is also bounded by $O(nN^d)$.*

Main proof idea. It is clear that we only need to bound the number of the possible support sets of sizes larger than d. We first arbitrarily label all the points in S from 1 to $(n+N)$. For any $C \subseteq S$ with $|C| > d$, we define the *representation* of C as the $(d+1)$ points in C with the smallest labels (we say those $(d+1)$ points represent C). Let $a_1, a_2, \ldots, a_{d+1}$ be a tuple of $(d+1)$ points in S where $a_1, \ldots, a_k$ are red and $a_{k+1}, \ldots, a_{d+1}$ are blue. If $k = 0$ or $k = d+1$, there is no possible support set represented by these $(d+1)$ points because the number of the blue/red points in the support set can at most be d. Now consider the case that $1 \leq k \leq d$. It is easy to see that there exist a unique pair of parallel hyperplanes (h_r, h_b) such that h_r goes through $a_1, \ldots, a_k$ and h_b goes through $a_{k+1}, \ldots, a_{d+1}$, since S is in general position. If C is a possible support set represented by $a_1, a_2, \ldots, a_{d+1}$, then h_r and h_b must be the corresponding support planes. That means all the red/blue points in C must lie on h_r/h_b. Note that there are at most $2d$ points on h_r and h_b, which implies that the number of such C is constant. Consequently, the total number of the possible support sets of S is bounded by $O(nN^d)$. Since the separation-margin is uniquely determined by the support set, the number of the possible separation-margins is also bounded by $O(nN^d)$. □

By applying the previous formula for $Emar(S)$, we can enumerate all the $O(nN^d)$ possible support sets to compute the ESM of S. The $O(nN^d)$ possible support sets can be enumerated as follows. For the ones of the sizes less than $(d+1)$, we enumerate them in the obvious way. For the ones of sizes larger than or equal to $(d+1)$, we first enumerate a tuple of $(d+1)$ points (of which at least one is red and one is blue), which would be the representation of the support sets (see the main proof idea of Theorem 8 above). Via these $(d+1)$ points, we can determine two parallel hyperplanes h_r and h_b where h_r goes through the red ones and h_b goes through the blue ones. We then find all the points on h_r and h_b, the number of which is at most $2d$ (including the original $(d+1)$ points). Once we have those points, we are able to enumerate all the possible support sets represented by the original $(d+1)$ points. For each such possible support set C, $Mar(C)$ can be straightforwardly computed in constant time since $|C| \leq 2d$. To compute $\xi_S(C)$, we observe that C is the support set of the existent points iff

1. all the points in C_R (resp., C_B) lie on h_r (resp., h_b);
2. all the points in C exist;
3. none of the red (resp., blue) points on the same side of h_r (resp., h_b) w.r.t. h exists;
4. except the points in C, none of the red (resp, blue) points on h_r (resp., h_b) exists.

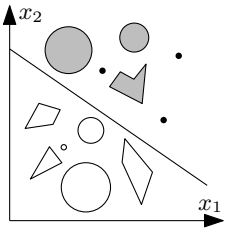


Figure 6 A separability problem for a set of bichromatic general objects in $\mathbb{R}^2$.

Among the four conditions, the first one has nothing to do with the existences of the stochastic points. If the enumerated set, C, violates this condition, then $\xi_S(C) = 0$. Otherwise, $\xi_S(C)$ is just equal to the product of the existence probabilities of the points in C (the second condition) and the non-existence probabilities of those points which should not exist (the last two conditions). If we use the simplest way, i.e., scanning all the points in S, to find the points on h_r and h_b (for enumerating the possible support sets represented by a set of $(d+1)$ points) as well as to compute each $\xi_S(C)$, the total time for computing $Emar(S)$ is $O(nN^{d+1})$. In fact, the runtime can be improved to $O(nN^d)$ by applying tricks similar to the ones used previously for improving the efficiency of our SP computing algorithm. See the full version [18] for details.

3.3 Hardness of computing expected separation-margin

We show that the bound achieved in Theorem 8 is tight, which suggests that our algorithm for computing ESM may be difficult to improve further.

▶ **Theorem 9.** *For any stochastic dataset S, define $\kappa(S)$ to be the total number of its (distinct) possible separation-margins. Then for any constant d, there exists some dataset S containing n red points and N blue points ($n \leq N$) in $\mathbb{R}^d$ with $\kappa(S) = \Theta(nN^d)$.*

From the above theorem, we can conclude that any algorithm that explicitly considers every possible separation-margin of the stochastic dataset requires at least $\Omega(nN^d)$ time to compute the ESM. This then implies that our algorithm is optimal among this category of algorithms. To do better, the only hope is to avoid considering every possible separation-margin explicitly. However, this is fairly difficult (though may not be impossible) because of the lack of an explicit relationship among distinct separation-margins.

4 Extension to general geometric objects

In the previous sections, we considered the separability related problems for stochastic points only. In fact, the two problems can be naturally generalized to the case of general stochastic geometric objects (see Figure 6). In this paper, the general objects to be considered include polytopes with constant number of vertices, and/or d-dim closed balls with various radii. We show that, with some effort, our methods can be extended to solve the generalized versions of the SP and ESM problems. The stochastic model used is similar to the unipoint model: each object has a fixed location with an associated existence probability. For convenience, we still use $S = S_R \cup S_B$ to denote the given stochastic dataset, in which each element is either a polytope or a ball.

4.1 Reducing polytopes to points

To deal with polytopes is easy, because of the fact that the entire polytope is on one side of a (hyperplane) separator iff all its vertices are. Thus, we can simply replace each polytope with its vertices and associate with each vertex an existence probability equal to that of the polytope. In this way, the polytopes in S can be reduced to points. One thing should be noted is that, once we reduce the polytopes to points, the existences of the vertices of each polytope are dependent upon each other. However, this issue can be easily handled without any increase in time complexity, because each polytope only has $O(1)$ vertices.

4.2 Handling balls

Once we are able to use the vertices to replace the polytopes, it suffices to consider the separability problems for datasets containing only stochastic balls (points can be regarded as 0-radius balls). Before we discuss how to handle balls, we need a definition of general position for a ball-dataset. We say a set of balls in $\mathbb{R}^d$ is in general position (or has the general position property) if

1. the centers of the balls are in general position;
2. no $(d+1)$ balls have a common tangent hyperplane.

Furthermore, we say a ball-dataset has strong general position property (SGPP) if it satisfies the two conditions above and all of the 0-radius balls in it have SGPP (as defined in Section 2) when regarded as points. In Section 4.2.1, the given ball-dataset S is required to have SGPP. In Section 4.2.2, we only need the assumption that S has the (usual) general position property.

4.2.1 Separable-probability (ball-version)

Let $T = T_R \cup T_B$ be a set of bichromatic balls with SGPP and set $J = \{3, 4, \ldots, d\}$. Theorem 1 and 2 can be easily generalized to ball-datasets (the meaning of $\mathcal{CH}(T_R)/\mathcal{CH}(T_B)$ should be modified as the convex hull of all the balls in T_R/T_B). The ball-version of Theorem 3 (and also its proof) is slightly different, which we present as follows (the proof can be found in the full version [18]).

▶ **Theorem 10.** *T^{U^*} is weakly separable and there exists only one weak separator. Furthermore, the unique weak separator of T^{U^*} either goes through exactly d 0-radius balls (of which at least one is in $T_R^{U^*}$ and one is $T_B^{U^*}$) or is tangent to at least one ball with radius larger than 0.*

Once we generalize these results, we are immediately able to generalize the concept of extreme separator to ball-datasets. As we do in Section 2.1, if $P_0 \neq \emptyset$, we define the extreme separator of T as the derived separator of the unique weak separator of T^{U^*}. If $P_0 = \emptyset$, we recursively define the extreme separator of T as the derived separator of the extreme separator of $\Phi_J(T)$. If the extreme separator is directly defined (i.e., the case of $P_0 \neq \emptyset$), we call the subset consisting of all the balls tangent to extreme separator the *critical set*. We shall use the following theorem later for solving the ball version of the SP problem.

▶ **Theorem 11.** *Let $T = T_R \cup T_B$ be a separable ball-dataset whose extreme separator is directly defined and let C be its critical set. Then the extreme separator of C is also directly defined. Furthermore, T and C share the same extreme separator and auxiliary subspace.*

Theorem 11 implies that the extreme separator is uniquely determined by the critical set. This then gives us the basic idea to solve the problem, enumerating all the possibilities for

the critical set. As in Section 2.2, we can compute the SP of S as

$$Sep(S) = Sep(\Phi_J(S)) + \sum_{C \subseteq S} \lambda_S(C),$$

where $\lambda_S(C)$ is the probability that the critical set of the existent balls is C. Since the balls in S have SGPP, the size of the critical set of the existent balls can be at most d. Furthermore, the critical set should contain at least one red ball and one blue ball. Thus, it suffices to compute $\lambda_S(C)$ for all the subsets $C \subseteq S$ with $|C| \leq d$ which contain balls of both colors. We consider two cases separately. First, all the balls in C have radius 0. Second, there is at least one ball in C with radius larger than 0.

In the first case, according to Theorem 10, $\lambda_S(C) > 0$ only if $|C| = d$. Since the balls in C are actually points, the situation here is almost the same as what we confronted in the point-version of the problem. We can uniquely determine a hyperplane h which goes through the d points in C, and a subspace $U^* \in \mathcal{V}$ perpendicular to h. Then $\lambda_S(C)$ is just equal to the probability that h is the extreme separator of the existent balls. Also, the conditions for h to be the extreme separator are very similar to those in Section 2.2, which are

1. all the balls in C exist;
2. there exist $r \in \mathcal{CH}(C_R)$ and $b \in \mathcal{CH}(C_B)$ such that their projection images on U^* coincide;
3. no red (resp. blue) ball on the opposite (resp. same) side of h w.r.t. the point o exists, where the definition of o is similar to that in Section 2.2;
4. no ball intersecting with h exists, except the ones in C.

If C violates the second condition, then $\lambda_S(C) = 0$. Otherwise, $\lambda_S(C)$ is just equal to the product of the existence probabilities of the balls in C and the non-existence probabilities of the balls that should not exist.

In the second case, however, the size of C may be less than d. According to Theorem 11, if C is the critical set of the existent points, then the extreme separator and auxiliary subspace of the existent points are the same as those of C. This implies that $\lambda_S(C) = 0$ if C is not separable or the extreme separator of C is defined recursively. So we only need to consider the situation that the extreme separator of C is directly defined. Assume that C has the extreme separator h (directly defined) with the auxiliary subspace $U^* \in \mathcal{V}$. Let c be any ball in C with radius larger than 0. Then it is easy to see that C is the critical set of the existent balls iff

1. all the balls in C exist;
2. all the balls in C are tangent to h;
3. no ball with the same color as (resp. different color than) c but on the opposite (resp. same) side of h^* w.r.t. c exists;
4. no ball intersecting with h exists, except the ones in C.

Because $|C| = O(1)$, h and U^* can be computed in constant time using brute-force. Similarly, if C satisfies the second condition, $\lambda_S(C)$ is equal to the product of the existence probabilities of the balls in C and the non-existence probabilities of the balls that should not exist.

In both the cases, $\lambda_S(C)$ can be computed in linear time by simply scanning all the balls in S. Thus, the SP can be finally computed in $O(nN^d)$ time, as the number of the subsets C considered is bounded by $O(nN^{d-1})$. Unfortunately, the improvement techniques used in the point-version of the problem cannot be generalized to ball-datasets so that our eventual time bound for computing the SP of general geometric objects remains $O(nN^d)$.

4.2.2 Expected separation-margin (ball-version)

Let $T = T_R \cup T_B$ be a set of bichromatic balls in general position. Clearly, the definitions given in Section 3.1 (maximum-margin separator, separation-margin, support set/points/planes, etc.) can be directly generalized to ball-datasets. Also, with these definitions, the ball-versions of Theorem 6 and 7 can be easily verified (by using the same proofs).

To extend the previous algorithm, we need to prove the ball version of Theorem 8. The first step is the same as that in the original proof of Theorem 8: we arbitrarily label the balls in S and define the representation of C as the $(d+1)$ balls in C with the smallest labels, for any $C \subseteq S$ with $|C| > d$. We show that the number of possible support sets represented by any group of $(d+1)$ balls is $O(1)$. Let $a_1, a_2, \ldots, a_{d+1}$ be any $(d+1)$ balls in S where $a_1, \ldots, a_k$ are red and $a_{k+1}, \ldots, a_{d+1}$ are blue, where $1 \leq k \leq d$ as before. Let each ball a_i have center c_i and radius δ_i. If some possible support set C is represented by these $(d+1)$ balls, then the support plane h_r (resp. h_b) must be tangent to $a_1, \ldots, a_k$ (resp. $a_{k+1}, \ldots, a_{d+1}$). Furthermore, the balls $a_1, \ldots, a_k$ (resp. $a_{k+1}, \ldots, a_{d+1}$) must be on the open side of h_r (resp. h_b), i.e., the side different from the one containing the area between h_r and h_b. It can be shown that there are at most two possibilities for the support planes (h_r, h_b) (see the full version [18]). Then by following the logic in the proof of Theorem 8, we know the number of the possible support sets represented by these $(d+1)$ balls is constant, which immediately implies that the total number of all possible support sets is $O(nN^d)$.

To enumerate these possible support sets, we can directly use the same method as in Section 3.2, i.e., first enumerate $(d+1)$ balls and then enumerate the possible support sets represented by them. Again, because the improvement techniques used in the point-version of the problem do not work for ball-datasets, we have to scan all the balls once for computing the corresponding probability of each possible support set, which makes the overall time $O(nN^{d+1})$ for computing the ESM of general geometric objects.

5 Applications

In this section, we present some applications of our algorithms to stochastic convex-hull (SCH) related problems. Given a stochastic (point) dataset A, the SCH of A refers to the convex-hull of the existent points in A, which is an uncertain convex shape.

5.1 SCH membership probability problem

The SCH membership probability problem was introduced for the first time in [4]. The problem can be described as follows: given a stochastic dataset $A = \{a_1, \ldots, a_m\} \subset \mathbb{R}^d$ and a query point $q \in \mathbb{R}^d$, compute the probability that q is inside the SCH of A, which we call the *SCH membership probability* (SCHMP) of q w.r.t. A.

It is shown in [11] that the SCHMP problem in $\mathbb{R}^d$ can be reduced to the SP problem in $\mathbb{R}^{d-1}$. Due to this, by plugging in our SP computing algorithm presented in Section 2, we immediately obtain an $O(m^{d-1})$-time algorithm to compute SCHMP for $d \geq 3$, which matches the best known bound in [11]. Indeed, this bound can be achieved by applying any SP computing algorithm with runtime bounded by $O(N^d)$.

More interestingly, we show that our SP computing algorithm yields a more direct and natural method to solve the SCHMP problem in $O(m^{d-1})$ time for $d \geq 3$ and $O(m \log m)$ time for $d = 2$, which does not involve any non-trivial reduction between the two problems. Given a SCHMP problem instance (A, q), clearly, the query point q is outside the SCH of A iff it can be separated from the existent points in A by a hyperplane. Thus, we construct

a stochastic bichromatic dataset $S = S_R \cup S_B$, where S_R contains only one point, q, with existence probability 1 and $S_B = A$. Then the SCHMP of q w.r.t. A is just equal to $1 - Sep(S)$. This can be computed in $O(m^{d-1})$ time for $d \geq 3$ and $O(m \log m)$ time for $d = 2$ by applying our SP computing algorithm, since $|S_R| = 1$ and $|S_B| = m$. Note that the $O(m^{d-1})$ runtime of this simple method relies on the $O(nN^{d-1})$ time bound of our SP computing algorithm for $d \geq 3$. If we plug in an $O(N^d)$-time SP computing algorithm, the time cost will become $O(m^d)$. Interestingly enough, this method for computing SCHMP is a generalization of the witness-edge method in [4] to the case $d > 2$, where the latter was the first known approach that solves this problem in $\mathbb{R}^2$ and was thought to be difficult to be generalized to higher dimensions [4]. This can be seen as follows. When plugging in our SP computing algorithm, we enumerate all the possible extreme separators of $\{q\} \cup \Gamma$, where Γ denotes the set of the existent points in A. If the extreme separator is finally defined in $\mathbb{R}^{d-2k}$, it goes through $(d - 2k)$ points, of which one is q. These $(d - 2k)$ points form a $(d - 2k - 1)$-dim face of $\mathcal{CH}(\{q\} \cup \Gamma)$ about the vertex q. It is evident that this face is uniquely determined by the convex polytope $\mathcal{CH}(\{q\} \cup \Gamma)$. We call it the *witness-face* of q in $\mathcal{CH}(\{q\} \cup \Gamma)$. Then enumerating the possible extreme separators is equivalent to enumerating the possible witness-faces of q in $\mathcal{CH}(\{q\} \cup \Gamma)$. When $d = 2$, the concept of witness-face coincides with that of *witness-edge* defined in [4]. Thus, in this case, our method is identical to the witness-edge method.

5.2 Other SCH-related problems

Our algorithms presented in the previous sections can also be applied to solve some new problems related to SCH. Here we propose three such problems and show how to solve them.

- **SCH intersection probability problem**. This problem is a natural generalization of the SCHMP problem. Given a stochastic dataset $A = \{a_1, \ldots, a_m\} \subset \mathbb{R}^d$ and a query object Q which is a convex polytope with constant complexity (e.g., segment, simplex, etc.) in $\mathbb{R}^d$, the goal is to compute the probability that Q has non-empty intersection with the SCH of A. When Q is a single point, this is just the SCHMP problem. To solve this problem, we extend the method described in the preceding subsection. According to Theorem 1, Q has no intersection with the SCH iff its vertices can be separated from the existent points in A by a hyperplane. Based on this, by constructing a stochastic bichromatic dataset $S = S_R \cup S_B$, where S_R contains the vertices of Q with existence probability 1 and $S_B = A$, we can apply our SP computing algorithm to compute the desired probability in $O(m^{d-1})$ time (note that $|S_R|$ is $O(1)$ for Q has constant complexity).
- **SCH ε-distant probability problem**. This problem is another natural generalization of the SCHMP problem. Given a stochastic dataset $A = \{a_1, \ldots, a_m\} \subset \mathbb{R}^d$, a query point $q \in \mathbb{R}^d$, and a parameter $\varepsilon \geq 0$, the goal is to compute the probability that the distance from q to the SCH of A is greater than ε. When $\varepsilon = 0$, this is equivalent to the SCHMP problem. To solve this problem, we need to apply our generalized SP computing algorithm presented in Section 4. Clearly, q has a distance greater than ε to the SCH of A iff the ε-ball centered at q can be separated from the existent points in A by a hyperplane. Thus, by constructing a generalized stochastic bichromatic dataset $S = S_R \cup S_B$, where S_R contains the ε-ball centered at q with existence probability 1 and $S_B = A$, we can apply our generalized SP computing algorithm to compute the desired probability in $O(m^d)$ time.
- **Expected distance to a SCH**. Given a stochastic dataset $A = \{a_1, \ldots, a_m\} \subset \mathbb{R}^d$ and a query point $q \in \mathbb{R}^d$, the goal of this problem is to compute the expected distance from q to the SCH of A. To achieve this, we notice that the distance from q to the SCH of A is

just equal to the separation-margin of $\{q\} \cup \Gamma$, where Γ denotes the set of the existent points in A. Thus, we construct a stochastic bichromatic dataset $S = S_R \cup S_B$, where S_R contains only one point q with existence probability 1 and $S_B = A$. Then the problem can be solved in $O(m^d)$ time by plugging in our ESM computing algorithm presented in Section 3.

References

1 P.K. Agarwal, B. Aronov, S. Har-Peled, J.M. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty II. In *Proceedings of the 32nd Symposium on Principles of Database Systems*, pages 115–126. ACM, 2013.

2 P.K. Agarwal, S-W. Cheng, Y. Tao, and K. Yi. Indexing uncertain data. In *Proceedings of the Twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 137–146. ACM, 2009.

3 P.K. Agarwal, S-W. Cheng, and K. Yi. Range searching on uncertain data. *ACM Transactions on Algorithms (TALG)*, 8(4):43, 2012.

4 P.K. Agarwal, S. Har-Peled, S. Suri, H. Yıldız, and W. Zhang. Convex hulls under uncertainty. In *Algorithms-ESA 2014*, pages 37–48. Springer, 2014.

5 C.C. Aggarwal and P.S. Yu. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):609–623, 2009.

6 A. Agrawal, Y. Li, J. Xue, and R. Janardan. The most-likely skyline problem for stochastic points. *Submitted to a journal*, 2015.

7 N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Communications of the ACM*, 52(7):86–94, 2009.

8 M. de Berg, A.D. Mehrabi, and F. Sheikhi. Separability of imprecise points. In *Algorithm theory – SWAT 2014*, volume 8503 of *LNCS*, pages 146–157. Springer, 2014.

9 M. de Berg, M. van Kreveld, M. Overmars, and O.C. Schwarzkopf. *Computational Geometry*. Springer, 2000.

10 H. Edelsbrunner and L.J. Guibas. Topologically sweeping an arrangement. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, pages 389–403, 1986.

11 M. Fink, J. Hershberger, N. Kumar, and Suri S. Hyperplane separability and convexity of probabilistic points. *32nd International Symposium on Computational Geometry*, 2016.

12 P. Kamousi, T.M. Chan, and S. Suri. Stochastic minimum spanning trees in euclidean spaces. In *Proceedings of the Twenty-seventh Annual Symposium on Computational geometry*, pages 65–74. ACM, 2011.

13 Y. Li, J. Xue, A. Agrawal, and R. Janardan. On the arrangement of stochastic lines in $\mathbb{R}^2$. *Submitted to a journal*, 2015.

14 M. Löffler. *Data imprecision in computational geometry*. PhD thesis, Utrecht Univ., 2009.

15 M. Löffler and M. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

16 S. Suri and K. Verbeek. On the most likely Voronoi Diagram and nearest neighbor searching. In *Algorithms and Computation*, pages 338–350. Springer, 2014.

17 S. Suri, K. Verbeek, and H. Yıldız. On the most likely convex hull of uncertain points. In *Algorithms–ESA 2013*, pages 791–802. Springer, 2013.

18 J. Xue, Y. Li, and R. Janardan. On the separability of stochastic geometric objects, with applications. (Full version). *ArXiv:1603.07021 [cs.CG]*, 2016.

19 W. Zhang. *Nearest-neighbor searching under uncertainty*. PhD thesis, Duke Univ., 2012.

Approximating Convex Shapes With Respect to Symmetric Difference Under Homotheties

Juyoung Yon[*1], Sang Won Bae[†2], Siu-Wing Cheng[3], Otfried Cheong[‡4], and Bryan T. Wilkinson[§5]

1 **KAIST, Daejeon, Korea**
2 **Kyonggi University, Suwon, Korea**
3 **HKUST, Hong Kong**
4 **KAIST, Daejeon, Korea**
5 **MADALGO, Aarhus, Denmark**

Abstract

The symmetric difference is a robust operator for measuring the error of approximating one shape by another. Given two convex shapes P and C, we study the problem of minimizing the volume of their symmetric difference under all possible scalings and translations of C. We prove that the problem can be solved by convex programming. We also present a combinatorial algorithm for convex polygons in the plane that runs in $O((m+n)\log^3(m+n))$ expected time, where n and m denote the number of vertices of P and C, respectively.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases shape matching, convexity, symmetric difference, homotheties

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.63

1 Introduction

The *shape matching* problem is the task of evaluating the similarity of two objects, and the *shape approximation* problem is to find a simpler representation of a given shape. These two problems arise in various fields and have numerous applications. Both problems call for a quantification of the approximation error, which requires a definition of the "distance" between a pair of shapes. The symmetric difference is one such measure, and it is robust against noise [16]. The symmetric difference of two sets X and Y is defined as $X \triangle Y = (X \setminus Y) \cup (Y \setminus X) = (X \cup Y) \setminus (X \cap Y)$. We use $|X|$ to denote the volume of a set X. The symmetric difference is symmetric, $X \triangle X = \emptyset$ for every set X, and the volume of the symmetric difference satisfies the triangle inequality, that is $|X \triangle Y| \leqslant |X \triangle Z| + |Z \triangle Y|$ as $X \triangle Y = (X \triangle Z) \triangle (Z \triangle Y) \subseteq (X \triangle Z) \cup (Z \triangle Y)$. If we restrict ourselves to a family of sets where $|X \triangle Y| = 0$ only if $X = Y$, then the symmetric difference is a metric over that family. This is, for instance, the case for convex compact shapes [16].

* Yon is supported by ICT R&D program of MSIP/IITP [R0126-15-1108] and by the Research Grant Council, Hong Kong, China, under the HKPF and GRF project 611711, respectively.
† Bae was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2013R1A1A1A05006927) and by the Ministry of Education (NRF-2015R1D1A1A01057220).
‡ Cheong is supported by ICT R&D program of MSIP/IITP [R0126-15-1108] and by the Research Grant Council, Hong Kong, China, under the HKPF and GRF project 611711, respectively.
§ Wilkinson is supported in part by the Danish National Research Foundation grant DNRF84 through the Center for Massive Data Algorithmics and in part by the Natural Sciences and Engineering Research Council of Canada. Part of this research was done while he visited KAIST.

© Juyoung Yon, Sang Won Bae, Siu-Wing Cheng, Otfried Cheong, and Bryan T. Wilkinson;
licensed under Creative Commons License CC-BY

The shape approximation problem often comes in the following form: Given shapes P and C and a family of transformations Φ (translations, rigid motions, scalings, etc.), find the transformation $\varphi \in \Phi$ that minimizes the distance between P and φC. In this paper, we consider the class Φ of *homotheties* and (a generalization of) the symmetric difference distance. (Two sets are *homothets* if they differ by a scaling factor and possibly a translation.) In other words, we compute $\min_{\varphi\in\Phi} |P \triangle \varphi C|$, for two convex shapes P and C in $\mathbb{R}^d$.

Our results are the first that allow transformations beyond translations in minimizing the volume of the symmetric difference of two convex shapes. In fact, scaling is rarely considered in previous shape matching results.

We optimize a more general function Δ defined as follows. Let κ be a parameter from $[0, 1]$ chosen *a priori*. We use rX to denote the set X scaled by a factor r.

$$\begin{aligned} \operatorname{dist}(P, Q) &:= (1-\kappa)\,|P \setminus Q| + \kappa\,|Q \setminus P| \\ \Delta(\lambda, t) &:= \operatorname{dist}(P, \lambda C + t). \end{aligned}$$

The problem is finding $\lambda \geqslant 0$ and $t \in \mathbb{R}^d$ that minimize $\Delta(\lambda, t)$. For $\kappa = \frac{1}{2}$, $\operatorname{dist}(P, Q)$ is equal to $\frac{1}{2}|P \triangle Q|$. There is a bias for P or Q if $\kappa < \frac{1}{2}$ or $\kappa > \frac{1}{2}$, respectively.

The parametrized measure $\operatorname{dist}(\cdot)$ may find applications in determining a simple data description in machine learning (e.g. [15]). Assume that we are given a set of uniformly distributed examples (data points) in some high dimensional feature space. The positive examples cover some region P and the examples outside P are negative. If we seek a simpler model C to classify the positive examples, we may minimize the number of false positives in $C \setminus P$ and the number of false negatives in $P \setminus C$, which can be approximated by $|C \setminus P|$ and $|P \setminus C|$, respectively, because of the uniform distribution assumption. If P is a convex body enclosing the positive examples (e.g., convex hull) and C is a simpler convex shape (e.g. ellipsoid), then the measure $\operatorname{dist}(\cdot)$ is applicable. False positives and false negatives are often not equally important; for example, false positives are more serious in spam filtering because they interfere with mail delivery. This difference in importance can be captured by using an appropriate κ in the definition of $\operatorname{dist}(\cdot)$.

Main results

Although Δ is not even quasiconvex, we prove that Δ can be minimized by convex programming. Moreover, if P and C are convex polytopes, then every local minimum of Δ is also a global minimum. We also present a combinatorial algorithm that minimizes Δ for a pair of convex polygons P and C in the plane. The algorithm runs in $O((m+n)\log^3(m+n))$ expected time, where n and m denote the number of vertices in P and C, respectively. We also establish several necessary conditions for (λ, t) to be a local minimum of Δ.

Related work

A lot of previous results consider bounds on $|P \triangle Q|$ for convex shapes P and Q that are fixed in place, or finding a good approximation of P in a given class of convex shapes. Groemer [10] studied some inequalities that relate different similarity measures. Let $\delta(P, Q)$ denote the Hausdorff distance between P and Q. Groemer proved that for a given pair of compact convex sets P and Q in $\mathbb{R}^d$, if $P \cap Q \neq \emptyset$, then $|P \triangle Q| = O(\delta(P, Q))$ and $\delta(P, Q) = O(|P \triangle Q|^{1/d})$ for $d \geqslant 2$, where the big-O constants depend on P, Q, and d.

Brass and Lassak [5] described several known bounds on $|P \triangle T|$, where T is a specific triangle chosen for an arbitrary convex polygon P: (1) for all P, there exists $T \subseteq P$ such that $|P \triangle T| \leqslant \left(1 - \frac{3\sqrt{3}}{4\pi}\right)|P| \approx 0.586|P|$, and (2) for all P, there exists T containing P

such that $|P \triangle T| \leqslant 2|P|$. Schymura [14] expressed the effect of two rigid motions φ_1 and φ_2 on the overlap of two bounded sets X and Y in $\mathbb{R}^d$ (not necessarily convex) in terms of $|\varphi_1(X) \triangle \varphi_2(X)|$. For $i \in \{1, 2\}$, φ_i consists of a translation v_i and a rotation ρ_i. Let α denote the norm of the vector $v_1 - v_2$, and β be the maximum distance between $\rho_1(x)$ and $\rho_2(x)$ among the points x in the boundary of X. Schymura showed that the difference between $|\varphi_1(X) \cap Y|$ and $|\varphi_2(X) \cap Y|$ is at most $\frac{1}{2}|\varphi_1(X) \triangle \varphi_2(X)|$, which is at most $\frac{1}{2}(\alpha+\beta)$ times the $(d-1)$-volume of the boundary of X.

Fleischer et al. [8] studied the problem of approximating a convex polygon by a homothetic pair of circumscribing and inscribed k-gons (not necessarily regular). They are interested in two quantities: $\lambda(P, Q) = \min\{\frac{s}{r} \mid rQ + t \subseteq P \subseteq sQ + t' \text{ for some translations } t \text{ and } t'\}$ and $\lambda_k = \min\{\lambda(P, Q_k) \mid \text{convex polygon } P \text{ and convex } k\text{-gon } Q\}$. They showed that $1 + \frac{\sqrt{5}}{2} = 2.118\cdots \leqslant \lambda_3 \leqslant 2.25$ and $\lambda_k = 1 + \Theta(k^{-2})$. They developed an algorithm that, for any convex polygon P with n vertices, finds a triangle T in $O(n^2 \log^2 n)$ time such that $\lambda(P, T)$ is minimized. They also presented another algorithm that finds a triangle T in $O(n)$ time such that $\lambda(P, T) \leqslant 2.25$.

Alt et al. [2] studied the problem of determining an axis-parallel rectangle or circle Q to minimize $|P \triangle Q|$ for any convex polygon P. Suppose that P has n vertices and they are stored in an array in clockwise order around the polygon boundary. Alt et al.'s algorithms find a solution rectangle in $O(\log^3 n)$ time and a solution circle in $O(n^5 \log n)$ time.

Before our work in this paper, previous algorithms for computing the exact similarity of two convex shapes with respect to the symmetric difference allow translations only. These algorithms solve the equivalent problem of maximizing the volume of the intersection of the two shapes. Given two convex polygons in the plane with a total of n vertices, de Berg et al. [7] proposed an algorithm that finds the translation that maximizes the area of the intersection of the two polygons in $O(n \log n)$ time. For two convex polytopes in $\mathbb{R}^d$ with a total of n facets, Ahn, Cheng, and Reinbacher [1] presented an algorithm that finds the translation that maximizes the intersection volume in $O(n \log^{3.5} n)$ time for $d = 3$ and $O(n^{\lfloor d/2 \rfloor + 1} \log^d n)$ time for $d \geqslant 4$ under some genericity assumption. In the plane, if one settles for a constant factor upper bound on the similarity of two convex shapes, Alt et al. [3] showed that a much larger class Φ of transformations can be allowed. The class Φ can be as general as the class of affine mappings, the class of similarities (combinations of scaling and rigid motion), the class of homotheties, the class of rigid motions, or just the class of translations. Alt et al. showed that there exists a transformation $\sigma \in \Phi$ such that the centroids of P and $\sigma(Q)$ are aligned, and $|P \triangle \sigma(Q)| \leqslant \frac{11}{3} \min_{\varphi \in \Phi} |P \triangle \varphi(Q)|$. For two convex polygons with a total of n vertices, Alt et al. developed two algorithms that find such a transformation σ, one for the class of homotheties and another for the class of rigid motions. The running times are $O(n)$ and $O(n^4)$ for homotheties and rigid motions, respectively.

2 Minimizing Δ by convex programming

We show how to minimize Δ by convex programming. Without loss of generality, we assume that the origin lies in C. Let us define

$$f(\lambda, t) := |(\lambda C + t) \cap P|, \qquad g(\lambda, t) := \big(f(\lambda, t)\big)^{1/d}.$$

The function $g(\lambda, t)$ is well defined as $f(\lambda, t) \geqslant 0$. We rewrite $\text{dist}(P, Q)$ as $\text{dist}(P, Q) = \kappa|Q| + (1 - \kappa)|P| - |Q \cap P|$. Let us define

$$q(\lambda, t) := f(\lambda, t) - \kappa|C|\lambda^d. \tag{1}$$

We then have

$$\begin{aligned}\Delta(\lambda,t) &= \operatorname{dist}(P,\lambda C+t) = \kappa|\lambda C+t| + (1-\kappa)|P| - f(\lambda,t)\\ &= \kappa|C|\lambda^d + (1-\kappa)|P| - f(\lambda,t) \qquad (2)\\ &= (1-\kappa)|P| - q(\lambda,t). \qquad (3)\end{aligned}$$

Since $(1-\kappa)|P|$ is a constant, minimizing $\Delta(\lambda,t)$ is equivalent to maximizing $q(\lambda,t)$. We can always make $\operatorname{dist}(P,\lambda C+t)$ equal to $(1-\kappa)|P|$ by setting λ to zero. It follows that the minimum of $\Delta(\lambda,t)$ is at most $(1-\kappa)|P|$, and any (λ,t) that makes $q(\lambda,t)<0$ is uninteresting because $\Delta(\lambda,t)$ would then be greater than $(1-\kappa)|P|$. Hence, we can restrict the domain to the following subset when looking for the minimum of Δ:

$$\mathfrak{D} := \{(\lambda,t) \mid q(\lambda,t) \geqslant 0\}.$$

We need to establish several properties of the domain $\mathfrak{D}$ and the functions f and q in order to conclude that convex programming is applicable.

2.1 Convexity of $\mathfrak{D}$

We first show that g is a concave function, that is the volume below the graph of g is convex. The lemma follows quite easily from the Brunn-Minkowski theorem [13], and the same argument has been used before [7].

▶ **Lemma 2.1.** *The function $g(\lambda,t)$ is concave on the domain where it is non-zero.*

Proof. It suffices to prove the claim along a line through the (λ,t)-space. Let this line be parametrized as $(\lambda(\xi),t(\xi))$, for $\xi\in\mathbb{R}$, with linear functions $\lambda(\xi)$ and $t(\xi)$. We extend our d-dimensional space to $d+1$ dimensions by adding a coordinate axis z, and write a point as (x,ξ), where $x\in\mathbb{R}^d$ and $\xi\in\mathbb{R}$. Let $\mathfrak{P}=\{(x,\xi)\in\mathbb{R}^d\times\mathbb{R} \mid x\in P\}$ and let $\mathfrak{C}=\{(x,\xi)\in\mathbb{R}^d\times\mathbb{R} \mid x\in\lambda(\xi)Q+t(\xi)\}$. Both $\mathfrak{P}$ and $\mathfrak{C}$ are convex sets, and so $\mathfrak{P}\cap\mathfrak{C}$ is a convex set as well. We note that the d-dimensional volume of the intersection of $\mathfrak{P}\cap\mathfrak{C}$ with the hyperplane $z=\xi$ is exactly $f(\lambda(\xi),t(\xi))$. By the Brunn-Minkowski Theorem in $\mathbb{R}^{d+1}$, the function $\xi\mapsto\big(f(\lambda(\xi),t(\xi))\big)^{1/d}=g(\lambda(\xi),t(\xi))$ is therefore a concave function. ◀

We show that $\mathfrak{D}$ is convex which is necessary for applying convex programming.

▶ **Lemma 2.2.** *The set $\mathfrak{D}=\{(\lambda,t) \mid q(\lambda,t)\geqslant 0\}$ is convex.*

Proof. It suffices to show that any line ℓ in the (λ,t)-space intersects $\mathfrak{D}$ in a single interval. Consider such a line ℓ. We parametrize it by λ, that is, we write $t=t(\lambda)$ for some linear function $t(\lambda)$. Recall that $g(\lambda,t)=\big(f(\lambda,t)\big)^{1/d}$. (We ignore the case where λ remains constant over ℓ, as this case follows immediately from the concavity of the function g.) We use $q(\lambda)$, $f(\lambda)$, and $g(\lambda)$ to denote $q(\lambda,t(\lambda))$, $f(\lambda,t(\lambda))$, and $g(\lambda,t(\lambda))$, respectively. We have $(\lambda,t(\lambda))\in\mathfrak{D}$ if and only if $0\leqslant q(\lambda)=f(\lambda)-\kappa|C|\lambda^d=\big(g(\lambda)\big)^d-\kappa|C|\lambda^d$, which is equivalent to $\big(g(\lambda)\big)^d\geqslant\kappa|C|\lambda^d$. Since both sides are positive, this is equivalent to $g(\lambda)\geqslant(\kappa|C|)^{1/d}\lambda$. Since g is concave by Lemma 2.1, the set of λ satisfying this inequality is an interval. ◀

2.2 Derivatives of the function f

For every choice of (λ,t), define the *rhizome* of C to be

$$\rho(\lambda,t) := \{x\in\partial C \mid \lambda x+t\in P\},$$

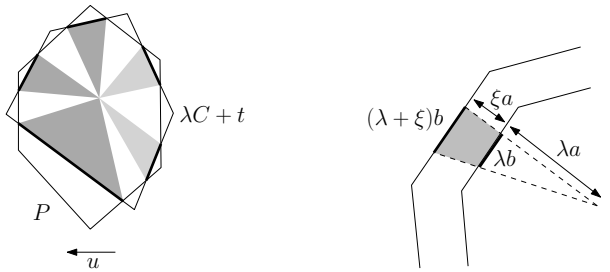


Figure 1 (a) Let B_1 and B_2 be the sets of points on the bold segments of the darkly and lightly shaded triangles, respectively. Let R be the union of all shaded triangles. The front and back rhizomes with respect to u are $\{\frac{1}{\lambda}(x-t) \mid x \in B_1\}$ and $\{\frac{1}{\lambda}(x-t) \mid x \in B_2\}$, respectively. And $\mathfrak{R}(\lambda, t) = \{\frac{1}{\lambda}(x-t) \mid x \in R\}$. (b) The shaded region is the volume swept by an infinitesimal expansion of a boundary element λb of λC.

where ∂A for any subset $A \subset \mathbb{R}^d$ denotes the boundary of A. In other words, the rhizome $\rho(\lambda, t)$ of C is the part of ∂C that lies in P under the homothety (λ, t). We will now argue that the partial derivatives of $f(\lambda, t)$ is closely related to the size of the rhizome.

Consider first a unit direction vector $u \in \mathbb{R}^d$ and the partial derivative of f at (λ, t) in translation direction u. We distinguish the *front rhizome*, which are the points $x \in \rho(\lambda, t)$ such that $\lambda x + t + \delta u \notin \lambda C + t$ for an arbitrarily small positive δ, and the *back rhizome*, which are the points $x \in \rho(\lambda, t)$ such that $\lambda x + t + \delta u$ lies in the interior of $\lambda C + t$ for an arbitrarily small positive δ. Figure 1(a) gives an illustration.

▶ **Lemma 2.3.** *Suppose that the intersection of the boundaries of $\lambda C + t$ and P has zero $(d-1)$-dimensional volume. For every unit direction vector $u \in \mathbb{R}^d$, $\frac{\partial}{\partial u} f(\lambda, t) = F - B$, where F and B denote the $(d-1)$-dimensional volumes of the projections in direction u of the front and back rhizomes at (λ, t) with respect to direction u, respectively.*

Proof. Fix some choices of $u \in \mathbb{R}^d$, $\lambda > 0$, and $t \in \mathbb{R}^d$. We denote by γ^+ and γ^- the front and back rhizomes of C at (λ, t) with respect to direction u, respectively. Consider $\lambda C + t + \delta u$. As δ increases infinitesimally from 0, a point on the front rhizome γ^+ contributes to a positive change, while a point on the back rhizome γ^- contributes to a negative change in $f(\lambda, t)$. On the other hand, if δ decreases infinitesimally from 0, then a point on the front rhizome γ^+ contributes to a negative change, while a point on the back rhizome γ^- contributes to a positive change in $f(\lambda, t)$. Thus, $\frac{\partial}{\partial u} f(\lambda, t) = f - b$, where f and b denote the $(d-1)$-dimensional volumes of the projections of γ^+ and γ^- in direction u, respectively. ◀

We now turn to the partial derivative $\frac{\partial}{\partial \lambda} f(\lambda, t)$. This is closely related to the union of segments that connect the points in the rhizome $\rho(\lambda, t)$ to the origin, which we assume to lie inside C. We denote this union of segments by $\mathfrak{R}(\lambda, t)$. Figure 1(a) gives an example.

▶ **Lemma 2.4.** *Suppose that the origin lies in the interior of C, and the intersection of the boundaries of $\lambda C + t$ and P has zero $(d-1)$-dimensional volume. Then, $\frac{\partial}{\partial \lambda} f(\lambda, t) = d\lambda^{d-1} |\mathfrak{R}(\lambda, t)|$.*

Proof. It suffices to prove the claim for a convex polytope C as the general case of C being a compact convex set then follows by a limit argument. Fix some choices of $\lambda > 0$ and $t \in \mathbb{R}^d$. Consider a boundary element b of $\rho(\lambda, t)$ that lies on a facet ℓ of C at distance a from the origin. Let β be the $(d-1)$-dimensional volume of b. The $(d-1)$-dimensional

volume of the boundary element λb on λC is $\lambda^{d-1}\beta$. For any $\xi \neq 0$ arbitrarily close to zero, being positive or negative, the contribution of this boundary element b to the difference $f(\lambda+\xi,t)-f(\lambda,t)$ is exactly $\frac{1}{d}(\lambda+\xi)^d a\beta - \frac{1}{d}\lambda^d a\beta$. Figure 1(b) shows an illustration. Taking the limit of this difference divided by ξ as ξ goes to 0, we have $\lim_{\xi\to 0}\frac{(\lambda+\xi)^d-\lambda^d}{d\xi}a\beta = \lambda^{d-1}a\beta$. Summing up this limit over all such boundary elements b of $\rho(\lambda,t)$ results in the derivative $\frac{\partial}{\partial\lambda}f(\lambda,t)$. On the other hand, $b \subseteq \rho(\lambda,t)$ contributes $\frac{1}{d}a\beta$ to the volume $|\mathfrak{R}(\lambda,t)|$, so we have $\frac{\partial}{\partial\lambda}f(\lambda,t) = d\lambda^{d-1}|\mathfrak{R}(\lambda,t)|$. ◀

We will need one more observation:

▶ **Lemma 2.5.** *Suppose that C and P are convex polytopes such that no facet of C is parallel to any facet of P. Then f is continuously differentiable at every point (λ,t).*

Proof. Since no facet of C is parallel to any facet of P, the intersection of the boundaries of $\lambda C+t$ and P has zero $(d-1)$-dimensional volume for all (λ,t). By Lemmas 2.3 and 2.4, the partial derivatives of f depend continuously on the rhizome $\rho(\lambda,t)$, which is a continuous function of (λ,t) under the assumption about the facets. ◀

2.3 Convex programming

Let us define the following function s over $\mathfrak{D}$:

$$s(\lambda,t) = \big(q(\lambda,t)\big)^{1/d}. \tag{4}$$

Note that $q(\lambda,t) \geqslant 0$ over $\mathfrak{D}$, so $s(\lambda,t)$ is well defined. By (3), minimizing $\Delta(\lambda,t)$ is equivalent to maximizing $q(\lambda,t)$ which is equivalent to maximizing $s(\lambda,t)$. We will prove in the following that $s(\lambda,t)$ is concave over $\mathfrak{D}$, which implies that we can maximize $s(\lambda,t)$ by convex programming.

To show the concavity of s in the domain $\mathfrak{D}$, it is sufficient to show concavity along all lines through $\mathfrak{D}$. Let $(\lambda,t(\lambda))$ be such a line, for a linear function $t(\lambda)$. For simplicity, we use $s(\lambda)$, $f(\lambda)$, $g(\lambda)$, and $q(\lambda)$ to denote $s(\lambda,t(\lambda))$, $f(\lambda,t(\lambda))$, $g(\lambda,t(\lambda))$, and $q(\lambda,t(\lambda))$, respectively. We first show that $s''(\lambda) \leqslant 0$ assuming that f is smooth enough.

▶ **Lemma 2.6.** *If $f(\lambda)$ is twice differentiable at λ, then $s''(\lambda) \leqslant 0$.*

Proof. We first collect some derivatives:

$$\begin{aligned}
f(\lambda) &= (g(\lambda))^d \\
f'(\lambda) &= d(g(\lambda))^{d-1}g'(\lambda) && (5)\\
f''(\lambda) &= d(g(\lambda))^{d-1}g''(\lambda) + d(d-1)(g(\lambda))^{d-2}(g'(\lambda))^2 && (6)\\
q(\lambda) &= f(\lambda) - \kappa|C|\lambda^d = (s(\lambda))^d && (7)\\
q'(\lambda) &= f'(\lambda) - d\kappa|C|\lambda^{d-1} = d(s(\lambda))^{d-1}s'(\lambda) && (8)\\
q''(\lambda) &= f''(\lambda) - d(d-1)\kappa|C|\lambda^{d-2} = d(d-1)(s(\lambda))^{d-2}(s'(\lambda))^2 + d(s(\lambda))^{d-1}s''(\lambda) && (9)
\end{aligned}$$

From (8) we have

$$s'(\lambda) = \frac{q'(\lambda)}{d(s(\lambda))^{d-1}}$$

and substituting this into (9) we have

$$d(s(\lambda))^{d-1}s''(\lambda) = q''(\lambda) - d(d-1)(s(\lambda))^{d-2}\frac{(q'(\lambda))^2}{d^2(s(\lambda))^{2d-2}} = q''(\lambda) - \frac{d-1}{d}\frac{(q'(\lambda))^2}{q(\lambda)}$$

and therefore

$$d^2(s(\lambda))^{2d-1}s''(\lambda) = dq(\lambda)q''(\lambda) - (d-1)(q'(\lambda))^2.$$

We will show that the right hand side of the above equation is at most 0, which implies $s''(\lambda) \leqslant 0$. By (7)–(9), we have

$$\begin{aligned}
&dq(\lambda)q''(\lambda) - (d-1)(q'(\lambda))^2 \\
&\quad = d\left(f(\lambda) - \kappa|C|\lambda^d\right)\left(f''(\lambda) - d(d-1)\kappa|C|\lambda^{d-2}\right) - (d-1)\left(f'(\lambda) - d\kappa|C|\lambda^{d-1}\right)^2 \\
&\quad = df(\lambda)f''(\lambda) - d\kappa|C|\lambda^d f''(\lambda) - d^2(d-1)\kappa|C|\lambda^{d-2}f(\lambda) + d^2(d-1)\kappa^2|C|^2\lambda^{2d-2} \\
&\qquad - (d-1)f'(\lambda)^2 + 2(d-1)d\kappa|C|\lambda^{d-1}f'(\lambda) - d^2(d-1)\kappa^2|C|^2\lambda^{2d-2}.
\end{aligned}$$

Hence,

$$\begin{aligned}
dq(\lambda)q''(\lambda) - (d-1)(q'(\lambda))^2 = &\left(df(\lambda)f''(\lambda) - (d-1)(f'(\lambda))^2\right) \\
&- d\kappa|C|\lambda^{d-2}\left(\lambda^2 f''(\lambda) - 2(d-1)\lambda f'(\lambda) + d(d-1)f(\lambda)\right) \qquad (10)
\end{aligned}$$

By (6) and (5) we have

$$\begin{aligned}
d^2(g(\lambda))^{2d-1}g''(\lambda) &= df(\lambda)f''(\lambda) - d^2(d-1)(g(\lambda))^{2d-2}(g'(\lambda))^2 \\
&= df(\lambda)f''(\lambda) - d^2(d-1)(g(\lambda))^{2d-2}\frac{(f'(\lambda))^2}{d^2(g(\lambda))^{2d-2}} \\
&= df(\lambda)f''(\lambda) - (d-1)(f'(\lambda))^2.
\end{aligned}$$

Plugging this into (10) and applying again (5) and (6) we have

$$\begin{aligned}
&dq(\lambda)q''(\lambda) - (d-1)(q'(\lambda))^2 \\
&= \left(d^2(g(\lambda))^{2d-1}g''(\lambda)\right) - d\kappa|C|\lambda^{d-2}\big(\lambda^2 d(g(\lambda))^{d-1}g''(\lambda) + d(d-1)(g(\lambda))^{d-2}(g'(\lambda))^2\lambda^2 \\
&\qquad\qquad - 2(d-1)\lambda d(g(\lambda))^{d-1}g'(\lambda) + d(d-1)(g(\lambda))^d\big) \\
&= d^2(g(\lambda))^{d-1}\left(f(\lambda) - \kappa|C|\lambda^d\right)g''(\lambda) \\
&\qquad - d^2(d-1)\kappa|C|(g(\lambda))^{d-2}\lambda^{d-2}\left((g'(\lambda))^2\lambda^2 - 2g'(\lambda)\lambda g(\lambda) + (g(\lambda))^2\right) \\
&= d^2 g(\lambda)^{d-1}q(\lambda)g''(\lambda) - d^2(d-1)\kappa|C|(g(\lambda))^{d-2}\lambda^{d-2}\left(g'(\lambda)\lambda - g(\lambda)\right)^2 \\
&\leqslant 0,
\end{aligned}$$

since $q(\lambda) \geqslant 0$, $g(\lambda) \geqslant 0$, and $g''(\lambda) \leqslant 0$ on $\mathfrak{D}$. ◀

We are ready to prove that $s(\lambda, t)$ is concave over $\mathfrak{D}$.

▶ **Lemma 2.7.** *The function* $(\lambda, t) \mapsto s(\lambda, t) = \left(f(\lambda, t) - \kappa|C|\lambda^d\right)^{1/d}$ *is concave on the domain* $\mathfrak{D}$ *where* $f(\lambda, t) - \kappa|C|\lambda^d \geqslant 0$.

Proof. We first prove the theorem for the case where P and C are convex polytopes such that no facet of C is parallel to a facet of P. By Lemma 2.5, this implies that $f(\lambda, t)$ is continuously differentiable at every point (λ, t). In particular, this means that the function $\lambda \mapsto f(\lambda) = f(\lambda, t(\lambda))$ is continuously differentiable at any λ, and therefore $s(\lambda)$ is continuously differentiable everywhere on the domain $\mathfrak{D}$. In order to prove the lemma, it suffices to show that $s'(\lambda)$ is monotonously decreasing.

Consider the intersection polytope $I(\lambda) = (\lambda C + t(\lambda)) \cap P$. For a fixed λ, we can triangulate this polytope $I(\lambda)$ into d-dimensional simplices and express its volume $|I(\lambda)|$ as

the sum of volumes of those simplices. Note that the volume of a d-dimensional simplex is represented as a determinant involving $d+1$ vertices of the intersection polytope $I(\lambda)$.

As λ changes, the combinatorial structure of $I(\lambda)$ and its triangulation only changes at certain breakpoints. Between two consecutive breakpoints, the expression of the volume $|I(\lambda)|$ of the intersection polytope $I(\lambda)$ remains the same. On the other hand, since $t(\lambda)$ is a linear function of λ, each vertex of $I(\lambda)$ between any two consecutive breakpoints is represented by a linear function of λ. This implies that the volume $|I(\lambda)|$ is expressed by a polynomial in λ with degree at most d. The intersection volume $|I(\lambda)|$ is thus twice differentiable at any λ between two consecutive breakpoints, so by Lemma 2.6 we have $s''(\lambda) \leqslant 0$ between two consecutive breakpoints. This implies that $s'(\lambda)$ is monotonously decreasing in any maximal interval defined by two consecutive breakpoints. Since $s'(\lambda)$ is also continuous, $s'(\lambda)$ is monotonously decreasing for any λ on the domain $\mathfrak{D}$. This completes the proof of the lemma when C and P are polytopes such that no facet of C is parallel to any facet of P.

We then turn to the case where C and P are arbitrary convex shapes. Suppose to the contrary that the claim was false, that is, the function $s(\lambda)$ is not concave. This implies the existence of three values $\lambda_0 < \lambda_1 < \lambda_2$, where $\lambda_1 = (1-\tau)\lambda_0 + \tau\lambda_2$ with $0 < \tau < 1$, such that $s(\lambda_1) < (1-\tau)s(\lambda_0) + \tau s(\lambda_2)$, that is, the concavity of s is violated.

Let $\varepsilon = (1-\tau)s(\lambda_0) + \tau s(\lambda_2) - s(\lambda_1) > 0$. Since $s(\lambda)$ depends continuously on the objects P and C, we can approximate them by convex polytopes P' and C' such that $s(\lambda)$ changes by at most $\varepsilon/3$ for $\lambda \in \{\lambda_0, \lambda_1, \lambda_2\}$. This approximation can be done such that no facet of C' is parallel to any facet of P'. Since the concavity of s at the values $\lambda_0, \lambda_1, \lambda_2$ is still violated, this is a contradiction to our proof for the polytope case. ◀

The above lemma immediately implies one of the main results of this paper.

▶ **Theorem 2.8.** *The function $\Delta(\lambda, t)$ attains its global minimum within the domain $\mathfrak{D} = \{(\lambda, t) \mid f(\lambda, t) - \kappa|C|\lambda^d \geqslant 0\}$, and this minimum can be computed using convex programming.*

Proof. Recall that $\Delta(\lambda, t) = (1-\kappa)|P| - q(\lambda, t)$ by (3) and $s(\lambda, t) = (q(\lambda, t))^{1/d}$. Since Δ is greater than $(1-\kappa)|P|$ outside $\mathfrak{D}$, the global minimum of Δ is in $\mathfrak{D}$. This minimum corresponds to a maximum of s. Since $\mathfrak{D}$ is convex (Lemma 2.2) and s is concave over $\mathfrak{D}$ (Lemma 2.7), one can maximize s by convex programming to obtain the minimum of Δ. ◀

3 Additional properties of the intersection volume function

We refine the analysis of the rhizome and the partial derivatives of f for the case that P and C are convex polytopes. We show that every local minimum of Δ lies in the domain $\mathfrak{D}$, and is therefore a global minimum of Δ. We also prove that the function $\lambda \mapsto \min_{t \in \mathbb{R}^d} \Delta(\lambda, t)$ is unimodal. The partial derivatives of f and the unimodality of $\min_{t \in \mathbb{R}^d} \Delta(\lambda, t)$ will be useful for developing the combinatorial algorithm for convex polygons in the next section.

Recall that the rhizome $\rho(\lambda, t)$ of C is the part of ∂C that lies in P under the homothety (λ, t). Some portion of the rhizome $\rho(\lambda, t)$ can be mapped to points in ∂P. We define

$$\rho_0(\lambda, t) := \{x \in \partial C \mid \lambda x + t \in \partial P\},$$

and call it the *critical rhizome* of C at (λ, t). Note that $\rho_0(\lambda, t) \subseteq \rho(\lambda, t)$.

Consider first a unit direction vector $u \in \mathbb{R}^d$, and the partial derivative of f at (λ, t) in direction u, denoted by $\frac{\partial}{\partial u} f(\lambda, t)$. As we will see, the derivative $\frac{\partial}{\partial u} f(\lambda, t)$ of f in direction u is not always well defined, while both its left and right derivatives are well defined everywhere. Specifically, the left and the right partial derivatives of f at (λ, t) in direction

u are defined to be the following one-sided limits: $\frac{\partial_-}{\partial u} f(\lambda,t) = \lim_{\delta\to 0-} \frac{f(\lambda,t+\delta u)-f(\lambda,t)}{\delta}$ and $\frac{\partial_+}{\partial u} f(\lambda,t) = \lim_{\delta\to 0+} \frac{f(\lambda,t+\delta u)-f(\lambda,t)}{\delta}$. If the left and right derivatives are equal, then $\frac{\partial}{\partial u} f(\lambda,t)$ is well defined as $\frac{\partial}{\partial u} f(\lambda,t) = \frac{\partial_-}{\partial u} f(\lambda,t) = \frac{\partial_+}{\partial u} f(\lambda,t)$.

Like front and back rhizomes, we also distinguish between *front and back critical rhizomes*. The front critical rhizome consists of points $x \in \rho_0(\lambda,t)$ such that $\lambda x + t + \delta u \notin \lambda C + t$ for an arbitrary small positive δ, and the back critical rhizome consists of points $x \in \rho_0(\lambda,t)$ such that $\lambda x + t + \delta u$ lies in the interior of $\lambda C + t$ for an arbitrary small positive δ. The proof of this and the following lemma can be found in the full version of our paper.

▶ **Lemma 3.1.** *Assume that C and P are convex polytopes. For every unit direction vector $u \in \mathbb{R}^d$, $\frac{\partial_-}{\partial u} f(\lambda,t) = F - B + B_0$, $\frac{\partial_+}{\partial u} f(\lambda,t) = F - B - F_0$ and where F, B, F_0, and B_0 denote the $(d-1)$-dimensional volumes of the projections in direction u of the front, back, front critical, and back critical rhizomes of C at (λ,t) with respect to direction u, respectively.*

We now turn to the left and right partial derivatives of f at (λ,t) with respect to the scaling λ. Recall that $\mathfrak{R}(\lambda,t)$ denotes the union of segments that connect the points in the rhizome $\rho(\lambda,t)$ to the origin, which we assume to lie inside C. Analogously, we denote by $\mathfrak{R}_0(\lambda,t)$ the union of segments that connect the points in $\rho_0(\lambda,t)$ to the origin.

As done above, we analyze the partial derivative of f with respect to λ by looking into its one-sided versions $\frac{\partial_-}{\partial \lambda} f(\lambda,t) = \lim_{\delta\to 0-} \frac{f(\lambda+\delta,t)-f(\lambda,t)}{\delta}$ and $\frac{\partial_+}{\partial u} f(\lambda,t) = \lim_{\delta\to 0+} \frac{f(\lambda+\delta,t)-f(\lambda,t)}{\delta}$.

▶ **Lemma 3.2.** *Assume that C and P are convex polytopes. Suppose that the origin lies in the interior of C. Then, $\frac{\partial_-}{\partial \lambda} f(\lambda,t) = d\lambda^{d-1}|\mathfrak{R}(\lambda,t)|$ and $\frac{\partial_+}{\partial \lambda} f(\lambda,t) = d\lambda^{d-1}(|\mathfrak{R}(\lambda,t)| - |\mathfrak{R}_0(\lambda,t)|)$.*

Lemma 3.2 leads to bounds for the volume of the rhizome at a local minimum of Δ.

▶ **Lemma 3.3.** *Assume that C and P are convex polytopes. Suppose that $t \in \mathbb{R}^d$ is fixed at some point. If Δ achieves a local minimum under scaling at (λ,t), then $\kappa|C| \leqslant |\mathfrak{R}(\lambda,t)| \leqslant \kappa|C| + |\mathfrak{R}_0(\lambda,t)|$, and this holds for all choices of the origin inside C.*

Proof. A local minimum of $\Delta(\lambda,t)$ under scaling is a local maximum of $q(\lambda,t)$ under scaling. By Lemma 3.2, the one-sided derivatives $\frac{\partial_-}{\partial \lambda} f(\lambda,t)$ and $\frac{\partial_+}{\partial \lambda} f(\lambda,t)$ of f are always well defined, so those of q are well defined as well since $q(\lambda,t) = f(\lambda,t) - \kappa|C|\lambda^d$. Suppose that q attains a local maximum at (λ,t). Then, either one of the one-sided derivatives of q is zero or they have opposite signs. Therefore, $\left(\frac{\partial_-}{\partial \lambda} q(\lambda,t)\right) \cdot \left(\frac{\partial_+}{\partial \lambda} q(\lambda,t)\right) \leqslant 0$. By Lemma 3.2, we have $\frac{\partial_-}{\partial \lambda} q(\lambda,t) = \frac{\partial_-}{\partial \lambda}\left(f(\lambda,t) - \kappa|C|\lambda^d|\right) = d\lambda^{d-1}|\mathfrak{R}(\lambda,t)| - \kappa d|C|\lambda^{d-1}$, and $\frac{\partial_+}{\partial \lambda} q(\lambda,t) = \frac{\partial_+}{\partial \lambda}\left(f(\lambda,t) - \kappa|C|\lambda^d|\right) = d\lambda^{d-1}(|\mathfrak{R}(\lambda,t)| - |\mathfrak{R}_0(\lambda,t)|) - \kappa d|C|\lambda^{d-1}$. Since $|\mathfrak{R}_0(\lambda,t)| \geqslant 0$, we must have that $\frac{\partial_-}{\partial \lambda} q(\lambda,t) \geqslant 0$ and $\frac{\partial_+}{\partial \lambda} q(\lambda,t) \leqslant 0$. This implies that $\kappa|C| \leqslant |\mathfrak{R}(\lambda,t)| \leqslant \kappa|C| + |\mathfrak{R}_0(\lambda,t)|$. ◀

This immediately implies that any local minimum of Δ under scaling is attained in $\mathfrak{D}$.

▶ **Lemma 3.4.** *Assume that C and P are convex polytopes. Suppose that $t \in \mathbb{R}^d$ is fixed at some point. If Δ achieves a local minimum under scaling at (λ,t), then $q(\lambda,t) \geqslant 0$.*

Proof. Without loss of generality, we can choose an origin in C such that its image t lies in P. By Lemma 3.3, if Δ achieves a local minimum under scaling at (λ,t), then $|\mathfrak{R}(\lambda,t)| \geqslant \kappa|C|$, and therefore $f(\lambda,t) = |(\lambda C + t) \cap P| \geqslant \lambda^d|\mathfrak{R}(\lambda,t)| \geqslant \kappa|C|\lambda^d$. So, $q(\lambda,t) = f(\lambda,t) - \kappa|C|\lambda^d \geqslant 0$. ◀

Combining Lemma 3.4 with Lemma 2.7 and Theorem 2.8 gives the following result.

▶ **Theorem 3.5.** *Assume that C and P are convex polytopes. Every local minimum of $\Delta(\lambda, t)$ is also a global minimum, and it is attained in the convex domain $\mathfrak{D} = \{(\lambda, t) \mid f(\lambda, t) - \kappa|C|\lambda^d \geqslant 0\}$. This global minimum can be computed using convex programming.*

We can also analyze the dependence of the optimal approximation on the scaling factor λ.

▶ **Theorem 3.6.** *Assume that C and P are convex polytopes. The function $\Delta_{\min}(\lambda) = \min_{t \in \mathbb{R}^d} \Delta(\lambda, t)$ decreases monotonously from $\lambda = 0$ to its minimum at $\lambda = \lambda^*$, and then increases monotonously for $\lambda \geqslant \lambda^*$.*

Proof. The function $\Delta_{\min}$ is clearly decreasing for λ close to zero, and increasing for large λ. Let λ' be a local minimum of $\Delta_{\min}$, and let t' be such that $\Delta(\lambda', t')$ assumes this minimum. Then (λ', t') is a local minimum of Δ, and by Theorem 3.5 this is the unique value $\lambda' = \lambda^*$. The claim follows. ◀

4 Optimality under translations in the plane

Suppose that λ is fixed at some value. A placement (λ, t) is a local minimum of $\Delta(\lambda, t)$ under translation if and only if (λ, t) is a local maximum of $f(\lambda, t)$ under translation. We observed in Lemma 3.1 that if C and P are convex polytopes, both the left and right partial derivatives $\frac{\partial_-}{\partial u} f(\lambda, t)$ and $\frac{\partial_+}{\partial u} f(\lambda, t)$ with respect to any unit vector u are always well defined. This gives a general criterion for a local minimum under translations.

In this section we consider the planar case, and assume that the boundaries of $\lambda C + t$ and P intersect in a finite number of points. This is always true, for instance, when C and P are polygons without parallel edges or when C is a circle and P is a polygon. Under this assumption, the rhizome $\rho(\lambda, t)$ consists of a finite number of intervals of ∂C that we will denote $\rho_i(\lambda, t)$, for $1 \leqslant i \leqslant k$ and some constant k. For each rhizome interval $\rho_i(\lambda, t)$, let its endpoints be p_i and q_i such that $\rho_i(\lambda, t)$ is the counter-clockwise interval on ∂C from p_i to q_i. We define the *rhizome vector* $\vec{\rho}_i(\lambda, t)$ as the vector $q_i - p_i$.

▶ **Lemma 4.1.** *Let C and P be convex polygons in the plane such that no edge of C is parallel to any edge of P. Suppose that $\lambda > 0$ is fixed at some value. If Δ achieves a local minimum under translation at (λ, t), then the rhizome vectors sum to zero: $\sum_{i=1}^{k} \vec{\rho}_i(\lambda, t) = 0$.*

Proof. Let $p_i = (x_i, y_i)$ and $q_i = (x'_i, y'_i)$. Consider first the direction vector $u = (1, 0)$, that is, a translation in the positive x-direction. Let $F \subset \{1, 2, \ldots, k\}$ be the set of indices such that $\rho_i(\lambda, t)$ is part of the front rhizome. Similarly, let $B \subset \{1, 2, \ldots, k\}$ be the set of indices such that $\rho_i(\lambda, t)$ is part of the back rhizome.

For $i \in F$, we have $y'_i > y_i$, while for $i \in B$, we have $y'_i < y_i$. The length of the projection of $\rho_i(\lambda, t)$ on the y-axis is $y'_i - y_i$ for $i \in F$ and $y_i - y'_i$ for $i \in B$. (For $i \notin F \cup B$ we have $y_i = y'_i$.) By Lemma 2.3, the partial derivative $\frac{\partial}{\partial u} f(\lambda, t)$ is well defined, and it should be zero. Thus we have $\sum_{i \in F} y'_i - y_i = \sum_{i \in B} y_i - y'_i$, which is equivalent to $\sum_{i=1}^{k} y'_i - y_i = 0$.

We consider next the direction vector $u = (0, 1)$, that is, a translation in the positive y-direction. Arguing as before, we obtain $\sum_{i=1}^{k} x'_i - x_i = 0$. Putting both statements together the lemma follows. ◀

For the special case where C is a circle, we obtain another elegant characterization:

▶ **Lemma 4.2.** *Let C be a circle, and let P be a convex polygon in the plane. Suppose that $\lambda > 0$ is fixed at some value. If Δ achieves a local minimum under translation at (λ, t), then the centroid of the rhizome $\rho(\lambda, t)$ coincides with the center of C.*

Proof. We can assume that C is the unit circle and the center of C is $(0,0)$. Let α_i and β_i be the angular position of p_i and q_i on the circle. We then have $\vec{\rho_i}(\lambda,t) = q_i - p_i = \big(\cos\beta_i - \cos\alpha_i, \sin\beta_i - \sin\alpha_i\big)$. As in the proof of Lemma 4.1, we have $\sum_{i=1}^{k}(\cos\beta_i - \cos\alpha_i) = 0$ and $\sum_{i=1}^{k}(\sin\beta_i - \sin\alpha_i) = 0$.

On the other hand, the centroid of $\rho_i(\lambda,t)$ is

$$\left(\int_{\alpha_i}^{\beta_i} \cos\theta\,\mathrm{d}\theta,\ \int_{\alpha_i}^{\beta_i} \sin\theta\,\mathrm{d}\theta\right) = (\sin\beta_i - \sin\alpha_i,\ \cos\alpha_i - \cos\beta_i)\,.$$

Therefore, the centroid of the entire rhizome coincides with $(0,0)$, the center of circle C. ◀

5 Exact algorithm

Let (λ^*, t^*) be the global minimum of $\Delta(\lambda,t)$. In Section 2, we showed that one can compute numerical approximations of λ^* and t^* via convex programming for arbitrary convex bodies C and P in $\mathbb{R}^d$. There are efficient convex programming solvers such as CVX and MOSEK, and they would be the solution of choice in practice. However, the convex programming routines are numerical in nature and their running times depend on the input bit complexity and the solution precision required. Therefore, from an algorithmic theory standpoint, it is justifiable to ask whether there is a fast, exact combinatorial algorithm in the RAM model (a common computation model for computational geometry). In this section, we sketch such an algorithm when P and C are convex polygons with n and m vertices, respectively. We assume without loss of generality that the origin lies in C. The details of the algorithm can be found in the full version of our paper.

It is not difficult to design an algorithm with running time polynomial in m and n. For example, when λ is fixed, the achievement by de Berg et al. [7] is to find the best translation in $O((m+n)\log(m+n))$ time despite the complexity of the translation configuration space (for a fixed λ) being $O(m^2n^2)$. Similarly, our goal is to find (λ^*, t^*) in $O((m+n)\mathrm{polylog}(m+n))$ time. Since there is a known algorithm by de Berg et al. [7] for a fixed λ under translation, it is natural to apply parametric search. However, in order to use Meggido's generic parametric search technique as a black box [12], one would need to parallelize the algorithm of de Berg et al., which is rather complicated. To keep a simpler and self-contained description, we present a direct randomized parametric search algorithm.

For simplicity, we assume that no vertex of C or P has the same y-coordinate as another vertex of C or P. This can always be enforced by a slight rotation.

5.1 Overview of the maximum overlap algorithm

Assume that λ is fixed. We outline de Berg et al.'s algorithm for finding the translation that maximizes the overlap (and hence minimizes the symmetric difference). There are two stages and one key subroutine for which we will develop parametric versions later.

Let $a_1, a_2, \cdots, a_n$ and $\lambda b_1, \lambda b_2, \cdots, \lambda b_m$ be the y-coordinates of the vertices of P and λC. There are mn vertical translations $a_i - \lambda b_j$ that align two vertices of P and λC horizontally. These vertical translations correspond to mn horizontal lines $y = \ell_i$ for $j = 1, 2, \cdots, mn$ from top to bottom in the translation configuration space. These lines cut the configuration space into $mn+1$ horizontal strips.

A key primitive of the algorithm is the following **line search**: Let ℓ be a given line in the translation configuration space. Compute the translation $t \in \ell$ that maximizes the overlap of $\lambda C + t$ with P. This can be done in $O(m+n)$ time [4].

The **first stage** of the algorithm localizes the horizontal strip, among the $mn + 1$ candidates in the translation configuration space, that contains the optimal placement of λC. We take $k_{\min} = 1$, $k_{\max} = mn$, $k = mn/2$, and find the optimal translations along the lines $y = \ell_j$ for $j \in \{k_{\min}, k, k+1, k_{\max}\}$. The maximum overlap values then tell us whether we can ignore strips below $y = \ell_{k+1}$ or those above $y = \ell_k$. A binary search will lead us to the desired horizontal strip in the translation configuration space in $O((m+n)\log(m+n))$ time.

In the **second stage**, the algorithm starts with the strip S that contains the goal placement of λC. Consider an arbitrary vertex λv of λC. For every horizontal line h within S, the line $\lambda v + h$ intersects the same edge(s) of P (at most two). Let uw be such an edge. Then, the translations that move λv onto uw form the line segment that joins $u - \lambda v$ and $w - \lambda v$ in the translation configuration space. Similarly, for every vertex w of P, the line $w + h$ intersects the same edge(s) of λC for any horizontal line h within S, and w and these edge(s) of λC induce at most two line segments in the translation configuration space. By taking the supporting lines of these line segments, we get an arrangement $\mathcal{A}$ of $O(m+n)$ lines in the configuration space. If we can identify which cell σ of $\mathcal{A}$ contains the optimal placement, then for all translations $t \in \sigma$, the combinatorial structure of $(\lambda C + t) \cap P$ is fixed. That is, each vertex of $(\lambda C + t) \cap P$ is the intersection of two fixed edges of λC and P, and thus we can obtain its coordinates as linear functions in t. The area of $(\lambda C + t) \cap P$ is a quadratic polynomial in t whose maximum value can be calculated using standard calculus. One can zoom into σ as follows. Compute a $(1/r)$-cutting for $\mathcal{A}$ for some appropriate constant $r > 1$. By solving the one-dimensional optimal placement of λC along the supporting line of each edge in the cutting, we zoom into one cell of the cutting and then retrieve the $(m+n)/r$ lines of $\mathcal{A}$ that intersect this cell. Afterwards, we recurse on these $(m+n)/r$ lines. We spend $O(m+n)$ time at each recursion level, resulting in a total of $O((m+n)\log(m+n))$ time to find the cell σ.

We extend de Berg et al.'s algorithm to a parametric search version in which λ is not fixed. We will also go through the same two stages. Since λ is not fixed, there is an extra dimension to the configuration space (the configuration space is now three-dimensional). We need to work with an arrangement of planes in three dimensions. However, it is time-consuming to deal with a three-dimensional configuration space in its full generality. The trick is to continuously constrain the possible interval of λ values that contains the optimal λ^*.

5.2 Decision algorithm: comparing λ with λ^*

We will need a method to test whether a given value λ' is less than, equal to, or greater than the optimal value λ^*. This decision needs to be done without knowing λ^*. The proof of the following lemma can be found in the full version.

▶ **Lemma 5.1.** *Suppose that the function $t \mapsto \Delta(\lambda', t)$ achieves the minimum at t'. Then, we have*

- *$\lambda' = \lambda^*$ if and only if $(\kappa|C| - |\mathfrak{R}(\lambda', t')|)(\kappa|C| - |\mathfrak{R}(\lambda', t')| + |\mathfrak{R}_0(\lambda', t')|) \leqslant 0$.*
- *$\lambda' < \lambda^*$ if and only if $\kappa|C| - |\mathfrak{R}(\lambda', t')| + |\mathfrak{R}_0(\lambda', t')| < 0$.*
- *$\lambda' > \lambda^*$ if and only if $\kappa|C| - |\mathfrak{R}(\lambda', t')| > 0$.*

Moreover, $\kappa|C| - |\mathfrak{R}(\lambda', t')|$ and $\kappa|C| - |\mathfrak{R}(\lambda', t')| + |\mathfrak{R}_0(\lambda', t')|$ can be computed in $O((m+n)\log(m+n))$ time given λ'.

5.3 Parametric version of searching along a line

The parametric version of the line search primitive is the following: We are given a plane h in configuration space, and search for $(\lambda, t) \in h$ that maximizes $|(\lambda C + t) \cap P|$. We represent the plane h as $l(\lambda)$—for a fixed λ, $l(\lambda)$ is a line in the translation space.

We do not solve the problem in its full generality. Instead, we assume that the input satisfies some conditions that we explain below.

Consider a fixed λ. As we vary $t \in l(\lambda)$, the edges of P that can be hit by a vertex λv of λC are those edges of P intersected by the line $\lambda v + l(\lambda)$. There are at most two such edges, but in general their identities vary depending on the value of λ. The same can be said for the edges of λC that can be hit by the vertices of P as we vary t along $l(\lambda)$. We assume that this does not happen within a range $[\lambda_1, \lambda_2]$ specified in the input. Precisely, as we vary $\lambda \in [\lambda_1, \lambda_2]$, for every vertex λv of λC, the line $\lambda v + l(\lambda)$ does not sweep over any vertex of P, and for every vertex w of P, the line $w - l(\lambda)$ does not sweep over any vertex of λC. Under this condition, we want to find $(\lambda, t) \in l(\lambda)$ that maximizes $|(\lambda C + t) \cap P|$.

There are events at which the combinatorial structure of $P \cap (\lambda C + t)$ changes. These are moments at which a vertex of λC runs into an edge of P or vice versa. Notice that a vertex λv of λC runs into at most two edges of P. The same is true for the events at which vertices of P running into edges of λC. There are $O(m + n)$ such events in total. These events happen in a fixed order as we vary t along $l(\lambda)$ when $\lambda = \lambda_1$. However, when λ is varied linearly within $[\lambda_1, \lambda_2]$, the occurrence time of such an event is a linear function in λ. That is, the occurrence times of the events as functions of λ form an arrangement of $O(m + n)$ lines. Each intersection of two linear functions gives a value of λ. We are only interested in the pair of intersections that are successive in the increasing order of λ value and sandwich λ^*. These two successive values $\lambda_3 \leqslant \lambda_4$ can be identified using expected $O(\log(m + n))$ binary search probes at the vertices of the arrangement by following the framework of randomized slope selection [11]. Each probe involves deciding whether the corresponding λ value is greater than or less than λ^* using Lemma 5.1. Therefore, we can find $[\lambda_3, \lambda_4]$ in $O((m + n) \log^2(m + n))$ expected time. Within $[\lambda_3, \lambda_4]$, the ordering of the events is fixed, and these events/lines divide the vertical strip $[\lambda_3, \lambda_4] \times (-\infty, \infty)$ into $O(m + n)$ *event trapezoids*. Sort these event trapezoids in vertical order in $O((m + n) \log(m + n))$ time.

We perform a binary search on the sorted event trapezoids as follows. Let I be the current event trapezoid that we are probing. Let $t = h_1(\lambda)$ and $t = h_2(\lambda)$ be the two lines that bound I. We first spend $O(m + n)$ time to identify the intersecting pairs of edges of $\lambda C + t$ and P, and this fixes the combinatorial structure of $P \cap (\lambda C + t)$. The area of $P \cap (\lambda C + t)$ is a quadratic function $F(\lambda, t)$ within this event trapezoid. The maximum of F is achieved only when $\partial F/\partial t = 0$, which gives a line $H\colon t = h(\lambda)$. If H does not intersect I within $[\lambda_3, \lambda_4]$, then $\partial F/\partial t$ has a fixed sign within I which tells us how to continue the binary search. If H intersects I within $[\lambda_3, \lambda_4]$, then H intersects H_1 and/or H_2 at one or two values of λ, say λ_5 and λ_6. Using Lemma 5.1, we can decide whether $\lambda_i < \lambda^*$ or $\lambda_i > \lambda^*$ for $i \in \{5, 6\}$. This allows us to constrain $[\lambda_3, \lambda_4]$ to a smaller interval $[\lambda', \lambda'']$ such that, within $[\lambda', \lambda'']$, H lies either completely outside or inside I. In the former case, $\partial F/\partial t$ has a fixed sign within I and it tells us how to continue the binary search. The latter case is the terminating case of the binary search, that is, F achieves its maximum within I, and we can obtain this maximum by substituting $t = h(\lambda)$ into $F(\lambda, t)$ and optimize the resulting quadratic function in λ. In all, we can compute the maximum overlap of $P \cap (\lambda C + t)$ over $t \in l(\lambda)$ and $\lambda \in [\lambda_1, \lambda_2]$ in $O((m + n) \log^2(m + n))$ expected time.

5.4 Parametric version of first stage

In $O(m \log m + n \log n)$ time, we order the vertices of P and C from top to bottom. Let their y-coordinates be $a_1 < a_2 < \cdots < a_{n-1} < a_n$ and $b_1 < b_2 < \cdots < b_{m-1} < b_m$, respectively.

Let $\ell_{ij}(\lambda)$ denote the set of translations that put the vertex of P with y-coordinate a_i and the vertex of λC with y-coordinate λb_j at the same height. Recall that for a fixed λ,

$\ell_{ij}(\lambda)$ is a horizontal line in de Berg et al.'s algorithm. Since λ can vary, $\ell_{ij}(\lambda)$ is a plane in the three-dimensional configuration space.

In the first stage we will identify a region in the three-dimensional configuration space that contains (λ^*, t^*) by binary search among the planes $\ell_{ij}(\lambda)$'s.

Consider a plane $\ell_{ij}(\lambda)$. We first constrain the range of λ to an interval $[\lambda_1, \lambda_2]$ such that $\lambda^* \in [\lambda_1, \lambda_2]$ and, as we vary $\lambda \in [\lambda_1, \lambda_2]$, for every vertex λv of λC, the line $\lambda v + \ell_{ij}(\lambda)$ does not sweep over any vertex of P and for every vertex w of P, the line $w + \ell_{ij}(\lambda)$ does not sweep over any vertex of Q. We determine λ_1 and λ_2 by binary search (using Lemma 5.1) on the critical values of λ. Since we cannot afford to produce the entire list of these roughly mn critical values for λ, we arrange these values in a matrix, and make use of searching a totally sorted matrix in linear time [9].

We are then in a position to apply the algorithm of Section 5.3 to determine the position (λ^*, t^*) with respect to $\ell_{ij}(\lambda)$.

To summarize, in the first stage we identify a range $[\lambda_1, \lambda_2]$ and two planes $\ell_{ij}(\lambda)$ and $\ell_{rs}(\lambda)$ such that $\lambda^* \in [\lambda_1, \lambda_2]$, $\ell_{ij}(\lambda)$ and $\ell_{rs}(\lambda)$ sandwich (λ^*, t^*), and no plane $\ell_{kl}(\lambda)$ in the configuration space lies between $\ell_{ij}(\lambda)$ and $\ell_{rs}(\lambda)$ or intersects one of them within the range $[\lambda_1, \lambda_2]$. The expected run time for this stage is $O((m+n)\log^3(m+n))$.

5.5 Parametric version of second stage

Let R be the region computed by the first stage. It lies within $[\lambda_1, \lambda_2]$ and is bounded by two planes in the configuration space. Consider a vertex v of C and a horizontal line h in the configuration space that stabs R. The first stage guarantees that for all $\lambda \in [\lambda_1, \lambda_2]$, the line $\lambda v + h$ intersects the same edge(s) of P. Similarly, for every vertex w of P, the line $w + h$ intersects the same edge(s) of λC for all $\lambda \in [\lambda_1, \lambda_2]$. Thus, there are at most $E \leqslant 2(m+n)$ events corresponding to translations that put a vertex of λC on an edge of P or vice versa. These events form subsets of planes in the three-dimensional configuration space. We call these planes *event planes*, and we can compute their equations in $O(m+n)$ time.

We compute in $O(m+n)$ time a $(1/4)$-cutting of these E event planes [6], which is a simplicial complex of constant size. Each cell of the cutting intersects at most $E/4$ other event planes. For every supporting plane H of cells in the cutting, since we have the output range $[\lambda_1, \lambda_2]$ of the first stage, we can apply the subroutine in Section 5.3 to find $(\lambda_H, t_H) \in H$ that minimizes Δ and hence maximizes s. This takes $O((m+n)\log^2(m+n))$ expected time. Let L be the supporting plane such that $s(\lambda_L, t_L)$ is maximum among all supporting planes. Due to the concavity of the function s, for every supporting plane $H \neq L$, the side of H that contains (λ^*, t^*) is the side that contains (λ_L, t_L). Let $\mathcal{A}$ be the arrangement of all supporting planes other than L. It means that we can identify the cell σ in $\mathcal{A}$ that contains (λ^*, t^*). The plane L may split σ into two pieces, each of which is contained in some cell in the cutting. Therefore, the number of event planes that intersect σ is at most $2(E/4) \leqslant E/2$. We collect the event planes that intersect σ and then recurse on σ. In $O(\log(m+n))$ rounds we can narrow down to a convex subset K of the configuration space in which there is no event. That is, the combinatorial structure of $P \cap (\lambda C + t)$ is fixed within K, and this structure can be determined in $O(m+n)$ time using any point $(\lambda, t) \in C$. Let $t = (t_x, t_y)$. Then, we can express the area of the overlap within K as a quadratic function in λ, t_x and t_y in $O(m+n)$ time. Finally, we can compute the three partial derivatives, set them to zero to obtain three linear equations in λ, t_x and t_y, and solve the linear system to obtain (λ^*, t^*). The overall expected running time is $O((m+n)\log^3(m+n))$.

▶ **Theorem 5.2.** *Let P and C be two convex polygons in the plane with n and m vertices, respectively. We can compute the optimal scaling factor $\lambda^* \geqslant 0$ and the optimal translation $t^* \in \mathbb{R}^2$ such that the area of $P \cap (\lambda^* C + t^*)$ is maximum in $O((m+n)\log^3(m+n))$ expected time. Hence, the area of $P \triangle (\lambda^* C + t^*)$ is minimum.*

Acknowledgment. We thank Sergio Cabello for suggesting this problem and conjecturing our Theorem 3.6. We also thank Emo Welzl for many helpful discussions during O. C.'s visit to ETH in 2015.

References

1. H.-K. Ahn, S.-W. Cheng, and I. Reinbacher. Maximum overlap of convex polytopes under translation. *Computational Geometry: Theory and Applications*, 46:552–565, 2013.
2. H. Alt, J. Blömer, M. Godau, and H. Wagener. Approximation of convex polygons. In *Automata, Languages and Programming*, volume 443 of *LNCS*, pages 703–716. Springer, 1990.
3. H. Alt, U. Fuchs, G. Rote, and G. Weber. Matching convex shapes with respect to the symmetric difference. *Algorithmica*, 21:89–103, 1998.
4. D. Avis, P. Bose, T. Shermer, J. Snoeyink, G. Toussaint, and B. Zhu. On the sectional area of convex polytopes. *In Communication at the 12th Annual Symposium on Computational Geometry*, page C, 1996.
5. P. Brass and M. Lassak. Problems on approximation by triangles. *Geombinatorics*, 10:103–115, 2001.
6. B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993.
7. M. de Berg, O. Cheong, O. Devillers, M. van Kreveld, and M. Teillaud. Computing the maximum overlap of two convex polygons under translations. *Theory of Computing Systems*, 31:613–628, 1998.
8. R. Fleischer, K. Mehlhorn, G. Rote, E. Welzl, and C. Yap. Simultaneous inner and outer approximation of shapes. *Algorithmica*, 8:365–389, 1992.
9. G. N. Frederickson and D. B. Johnson. Generalized selection and ranking: sorted matrices. *SIAM Journal on Computing*, 13:14–30, 1984.
10. H. Groemer. On the symmetric difference metric for convex bodies. *Beiträge zur Algebra und Geometrie*, 41:107–114, 2000.
11. J. Matoušek. Randomized optimal algorithm for slope selection. *Information Processing Letters*, 39:183–187, 1991.
12. N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of ACM*, 30:852–865, 1983.
13. R. Schneider. *Convex Bodies – the Brunn-Minkowski theory*. Cambridge University Press, 1993.
14. D. Schymura. An upper bound on the volume of the symmetric difference of a body and a congruent copy. *Advances in Geometry*, 14:287–298, 2014.
15. D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.
16. R. Veltkamp and M. Hagedoorn. Shape similarity measures, properties and constructions. In *Advances in Visual Information Systems*, volume 1929 of *LNCS*, pages 467–476. Springer, 2000.

Interactive Geometric Algorithm Visualization in a Browser*

Lynn Asselin[1], Kirk P. Gardner[2], and Donald R. Sheehy[3]

1 **University of Connecticut, Storrs, USA**
`lynn.asselin@uconn.edu`
2 **University of Connecticut, Storrs, USA**
`kirk.gardner@uconn.edu`
3 **University of Connecticut, Storrs, USA**
`don.r.sheehy@gmail.com`

Abstract

We present an online, interactive tool for writing and presenting interactive geometry demos suitable for classroom demonstrations. Code for the demonstrations is written in JavaScript using `p5.js`, a JavaScript library based on Processing.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems – Geometrical problems and computations

Keywords and phrases Computational Geometry, Processing, JavaScript, Visualisation, Incremental Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.64

Category Multimedia Contribution

1 Motivation

At CG Week 2013 in Rio de Janeiro, Suresh Venkatasubramanian presented at the Workshop on Geometric Computing Challenges with a strong plea for more and better tools for producing interactive pedagogical geometry demos. In his abstract[1] he describes the situation quite clearly:

> The early days of the Web were a goldmine for geometric algorithm demos. The visual and interactive nature of geometric algorithms, coupled with the proliferation of lightweight Java applets, made the HTML canvas a fantastic portable framework for teaching and demonstrations.
> Fast forward to 2013. We have a blizzard of Javascript frameworks, sophisticated HTML5 based in-browser animations, and a host of new platforms on which to demo algorithms. We also have a mature geometric computing platform in the form of CGAL. But we no longer have geometric demos that work well (or at all in many cases).

Here in 2016, the situation is not much different. In this paper, we describe a tool that fills this gap. It is informed by experience collected from teaching an introductory course on

* This work was partially supported by the National Science Foundation under grant CCF-1525978.
[1] `http://zeus.mat.puc-rio.br/~socg2013/index.php/workshops/geometric-computing-challenges#abstract-suresh`


© Lynn Asselin, Kirk P. Gardner, and Donald R. Sheehy;
licensed under Creative Commons License CC-BY

Computational Geometry at the University of Connecticut in which students produced such demos as course projects. In most cases, the challenges of **dealing with interaction and visualization dominated the geometric ideas**.

Our interactive geometric algorithm visualizer is designed with the following goals.

1. It should be possible to visualize a geometric algorithm without writing any explicit visualization code.
2. It should be possible to produce an interactive (incremental) demonstration without writing any explicit code to deal with interaction.

Satisfying these two goals implies that all of the visualization and interaction code is abstracted away from the user, who is presented with a simple editor and a canvas. When more complex interaction or visualization is desired, access to these elements is available.

The project is in active development, and the source code is available at `https://github.com/compugeom/compugeom.github.io`. A live, interactive sandbox is available at `http://compugeom.github.io`.

2 Geometric Primitives

Drawable classes take the form of primitives as well as data structures, and can be instantiated as geometric objects complete with visualization specifications, as well as any support functions needed to concisely represent the underlying mathematics. These provide a foundation for other data structures and algorithms to be built upon, as well as a natural way to abstract away visualization. The main objects currently provided are:

- Vertices, Edges and Faces - Objects which encapsulate `p5.js` primitives such as *point* and *line* that can be instantiated, visualized, and modified.
- Point sets - In particular, we allow for dynamic point sets which are generated by user input.
- Polygons - Our polygon implementation provides a cyclic Edge list data structure, and allows for easy indexing into the vertices, automatically wrapping indices around using modular arithmetic.
- Planar Straight Line Graphs - A simple half-edge data structure is used to store and traverse the Edges of an embedded planar graph.

As our objective is to construct a means for computational geometers, students, and developers to visualize algorithms in a way that mirrors the elegance of the underlying mathematics, we adopt a hierarchical paradigm in order to minimize redundancy within the framework. These objects therefore make extensive, but not unnecessary use of lower dimensional primitives and data structures, with visualizations and functionality building naturally off of their component parts. In addition to the classes listed above, we provide standard linear predicates such as triangle orientation tests (i.e. CCW tests).

3 Technical Details

The main library used is `p5.js` [1], a JavaScript implementation of the Processing software sketchbook. We also use the Ace code editor [2] to provide interactive visualizations in the web-page.

There is often a tradeoff between power and ease of use. In order to give a good balance between the two, we attempted to focus on our target users, computational geometry students and educators. Thus, the technical decisions are motivated by pedagogical concerns.

The p5.js[2] library provides a pure JavaScript implementation of the Processing software sketchbook: an API for rendering graphics such as shapes, lines and ellipses in 2D and 3D. These objects are rendered by placing them in an event loop, however there are no geometric objects provided by the p5.js library. To implement geometric algorithms in p5.js/Processing the provided visualization primitives are wrapped by classes that encapsulate geometric primitives. These primitives are then packaged alongside linear predicates and geometric data structures to provide data abstraction and visualization.

The first step in constructing such a framework is, therefore, to handle the instantiation and visualization of drawable objects. This is accomplished by wrapping the event loop in a data structure that maintains the order of execution and animation of our algorithm visualizer. Thus the CG_Environment class serves to abstract away as much of the embedded event loop as possible without restricting the user's access to the p5.js library. We have also created geometric primitives as objects, each containing relevant (and natural) properties and functions, as well as the ability to draw themselves when instantiated by a CG_Environment object. This is achieved by keeping matrix of objects CG_Environment.things sorted by dimension that are to be drawn in the event loop. Objects can also be used strictly as data structures without visualization by instantiating them directly from the library, as in Figure 1

4 Example: Incremental Convex Hull

As a proof of concept, we give an example of a incremental algorithm for computing the convex hull of a set of points. Usually, the term *incremental algorithm* means that the input is processed one element at a time. The primary invariant maintained after processing each element is that the state of the algorithm is a correct output if the current element were the last one. That is, the correctness condition stands in as a loop invariant. An alternative view is that the input arrives online and as each new input arrives, we update the output. This is the more interactive version and it's the perspective we take.

The CG_DynamicPointSet class defines a point set for which new points are generated with a mouse click on the canvas. Upon user input the update function is called, passing the new point as a parameter. Thus, to write a program, one need only instantiate a new DynamicPointSet, a new Polygon (for the output), and write the update function. Some example code is given below.

```
// this algorithm dynamically computes the convex hull of a point set.
demo.P = demo.DynamicPointSet();
var H = demo.Polygon();
// dynamically compute the convex hull of the point set
demo.P.updateFunction = function(q) {
    if (H.VertexList.length < 2) {
      H.newVertex(q);
      return;
    }
    var left = -1, right = -1;
    for (var i = 0; i < H.VertexList.length; i++) {
      if (H.orient(i, i+1, q) && H.orient(i, i-1, q)){ right = i; }
      if (H.orient(i+1, i, q) && H.orient(i-1, i, q)){ left = i; }
    }
    // if q is outside the hull remove vertices and insert q
    if (left != -1 && right != -1) { H.replace(q, left, right); }
};
```

[2] https://p5js.org/

reset run

```
1  // this algorithm dynamically computes the convex hull of a point set.
2  demo.P = demo.DynamicPointSet();
3  var H = demo.Polygon();
4  // dynamically compute the convex hull of the point set
5  demo.P.updateFunction = function(q) {
6      if (H.VertexList.length < 2) {
7        H.newVertex(q);
8        return;
9      }
10     var left = -1, right = -1;
11     for (var i = 0; i < H.VertexList.length; i++) {
12       if (H.orient(i, i+1, q) && H.orient(i, i-1, q)){ right = i; }
13       if (H.orient(i+1, i, q) && H.orient(i-1, i, q)){ left = i; }
14     }
15     // if q is outside the hull remove vertices and insert q
16     if (left != -1 && right != -1) {
17       var f = new CG_Face(new CG_Polygon([H.v(left), q, H.v(right)]));
18       demo.things[2] = [f];
19       H.replace(q, left, right);
20     }
21 };
22
23
```

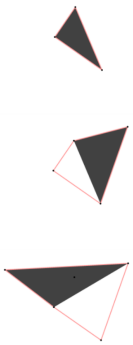


Figure 1 A screen capture from the live webpage (left) and several steps (right) of an augmented convex hull algorithm that visualizes the edited region of the polygon. It first constructs a face from `q` and the left and right-most remaining points, `var f = new CG_Face(new CG_Polygon([H.v(left), q, H.v(right)]))`, and sets it as the only 2D object that is to be drawn, `demo.things[2] = [f]`.

The demo object is an instance of the `CG_Environment` class, and is used here to instantiate a `DynamicPointSet` to handle user input and a `Polygon`, the convex hull. The rest of the code is intended to be representative of the algorithm specification by making use of predicates and functions that are natural to the data types in question.

Figure 1 illustrates a simple augmentation of the convex hull algorithm that makes use of the `CG_Face` class, taken directly from the web-page. In contrast to the factory function provided by the `CG_Environment` class, here we directly access the matrix, `CG_Environment.things`, of drawn objects in order to ensure we only have one 2D object: the face that represents the most recent change in the hull. This is how the user can have more direct access to the underlying drawing, when it is desired.

5 Future Work

There are many new features in active development. Among the most immediately useful are methods for visualizing geometric predicates. The main appeal of this project is the ability to quickly and easily implement and visualize geometric algorithms. Predicate visualizations would allow users to not only implement and view the algorithm's output, but also step through them visually.

In addition to predicate visualization we are in the process of integrating support for 3D algorithms and visualizations. By using the perspective and camera controls of `p5.js` we hope to provide an intuitive and painless interface for 3D input processing and visualization. We are also experimenting with other methods for stepping through code for non-incremental algorithms. Our goal is that this project will be beneficial to educators and students alike, allowing a clear interactive visual aid to procedures that would otherwise be presented in a textbook or lecture slide.

References

1 p5.js. Free Software Foundation, version 2.1. URL: `http://p5js.org/`.
2 Ace code editor. Free Software Foundation, version 2.1., 2010. URL: `https://ace.c9.io`.

Geometric Models for Musical Audio Data

Paul Bendich[1], Ellen Gasparovic[2], John Harer[3], and Christopher Tralie[4]

1 **Department of Mathematics, Duke University, USA; and Geometric Data Analytics, Inc., Durham, USA**
bendich@math.duke.edu
2 **Department of Mathematics, Union College, Schenectady, USA**
gasparoe@union.edu
3 **Department of Mathematics and Electrical and Computer Engineering, Duke University; and Geometric Data Analytics, Inc., Durham, USA**
harer@math.duke.edu
4 **Department of Electrical and Computer Engineering, Duke University, Durham, USA**
chris.tralie@gmail.com

Abstract

We study the geometry of sliding window embeddings of audio features that summarize perceptual information about audio, including its pitch and timbre. These embeddings can be viewed as point clouds in high dimensions, and we add structure to the point clouds using a cover tree with adaptive thresholds based on multi-scale local principal component analysis to automatically assign points to clusters. We connect neighboring clusters in a *scaffolding* graph, and we use knowledge of stratified space structure to refine our estimates of dimension in each cluster, demonstrating in our music applications that choruses and verses have higher dimensional structure, while transitions between them are lower dimensional. We showcase our technique with an interactive web-based application powered by Javascript and WebGL which plays music synchronized with a principal component analysis embedding of the point cloud down to 3D. We also render the clusters and the scaffolding on top of this projection to visualize the transitions between different sections of the music.

1998 ACM Subject Classification H.5.5 [Sound and Music Computing] Modeling, I.5.3 [Clustering] Algorithms

Keywords and phrases Geometric Models, Audio Analysis, High Dimensional Data Analysis, Stratified Space Models

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.65

Category Multimedia Contribution

1 Music Features

We are interested in automatically finding structure in musical audio data by first converting it to a point cloud and then invoking geometric tools we have developed for building structure on top of point clouds sampled from stratified spaces. Musical audio data is typically sampled around 44100 hz, so its raw form is often unwieldy, though it is possible to summarize some of the most important perceptual information at a much lower data rate which is more amenable to structural analysis. A variety of lossy audio features have been hand-designed


© Paul Bendich, Ellen Gasparovic, John Harer, and Christopher Tralie;
licensed under Creative Commons License CC-BY

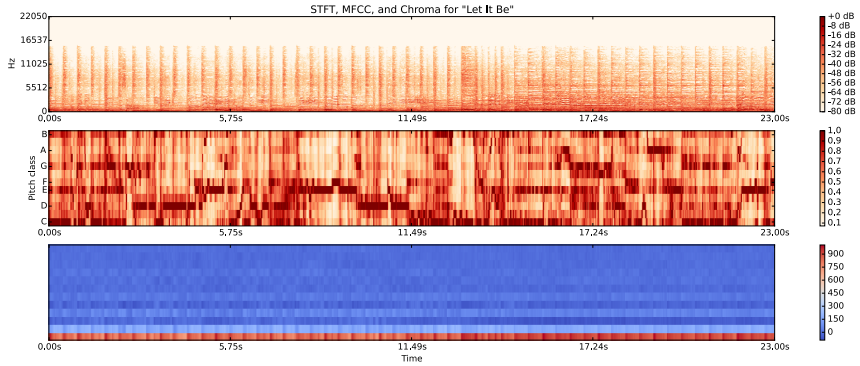


Figure 1 Audio features on top of a 23 second excerpt of "Let It Be" by The Beatles. The top figure shows the log Short-Time Fourier Transform, the middle figure shows chroma features, and the bottom figure shows MFCC features. These features were computed with the help of the "librosa" library in Python (`https://github.com/bmcfee/librosa`). [5]

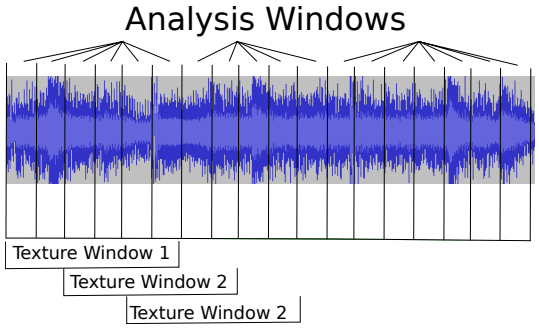


Figure 2 Features are extracted in short time 30ms windows of audio called analysis windows, and the features in blocks of these windows are summarized in texture windows.

on top of the Short-Time Fourier Transform (SFTF) to pull out perceptual information about small chunks of audio in non-overlapping 30 millisecond "analysis windows," and a sequence of these features may then be used in place of the original audio. Two important complementary sets of such features are the Mel-Frequency Cepstral Coefficients (MFCC) [4] and chroma features [1]. MFCC retains coarse information about the entire smoothed spectrum of each time frequency window, while chroma summarizes pitch in 12 equivalence classes of frequencies across all octaves in a time frequency window, one feature for each halfstep in the Western scale. (Note that other spacings are also possible.) There are a number of other features in addition to chroma and MFCC that can be used to summarize information from the STFT. For an overview of these "timbral features," see [6].

If each feature is viewed as a dimension, then the features in each time frequency window can be thought of as a point in some Euclidean space. But since all of these features are computed over very short time windows, it is possible that many windows will be similar even in very different sections of the song. This problem is readily addressed by using much larger overlapping blocks of analysis windows called "texture windows" to capture more time evolution of sound in each audio chunk, as in [6]; see Figure 2. We simply take the mean and standard deviation of each feature over all of the analysis windows in a texture block. In particular, we take 5 timbral features, 12 MFCC coefficients, and 12 chroma features in each analysis window, for a total of 29 features, which we then summarize over 7 second

blocks with their mean and standard deviation (233 30ms analysis windows for each 7 second texture window). We also add one "low energy" feature [6] for a total of 59 features so that the points live in $\mathbb{R}^{59}$. Finally, since we are fusing a collection of heterogeneous features, we normalize each dimension so that its standard deviation is 1 over the data cloud.

2 Geometric Models

Now that we have a scheme for turning audio into a point cloud, we turn to techniques we have developed for inferring stratified space structure from sampled points. More details can be found in [2], but we briefly summarize our techniques here. We first build a cover tree [3] on our point cloud X, which is a multi-scale way of organizing the point cloud. Each node in the tree is associated with a particular "center," which is a point in X. Each level l of the tree has an associated subset X_l of X and a radius $R_l = R_0/2^l$ so that the union of the balls of radius $2R_l$ centered at points in X_l cover all of the points in X. This condition makes it so that the subset of points at each level l goes from coarse to fine (and eventually includes all points in X). To ensure that the tree is balanced and that the collection of points of X at each level are evenly distributed, it is also the case that two centers at a particular level are each at least $2R_l$ apart from each other.

For a more parsimonious representation of our data, we choose a subtree of the cover tree so that the leaf nodes summarize geometrically homogeneous regions. In order to automatically choose which nodes to keep, we perform a version of principal component analysis (PCA) on the set of points within a certain radius of each center point at each level, and we do this at many radii. This process is known as *multi-scale local PCA* (MLPCA). Using a criterion based on the "eigenmetric" introduced in [2], for each node, we decide to either continue moving down the tree constructing subtrees, or to stop subdividing if the associated points have sufficiently similar eigenvalue profiles at multiple scales. The result of this adaptive process is a set of nodes at varying levels whose member points are geometrically similar to one another.

Our next step is to build a *scaffolding* graph on the resulting nodes, so that an edge is drawn between two nodes if the Euclidean distance between them is below some distance threshold. The distance threshold can be chosen automatically or manually in various ways; see [2]. Then, we use a local dimension estimation process based on MLPCA to estimate the intrinsic dimension locally near each one of the clusters. Namely, for each cluster, we perform PCA on the set of all points in that cluster together with all points in neighboring clusters connected to it in the scaffolding graph. Then, we compute the square roots of the eigenvalues we obtain from PCA, and use the largest gap between successive eigenvalues as an indication of what to choose as an initial dimension guess for that cluster. This initial guess may not be accurate, so we use knowledge from stratified space theory to inform and refine our local dimension estimate. A description of this refinement process may be found in [2]. Although this is the technique we chose to use in [2] and the one we used to create the examples in our Javascript demo, we point out that our framework for building the geometric models is flexible enough so that one can instead employ any local dimension estimation method of one's choice.

3 Javascript Demo

We created a GUI in Javascript/WebGL that enables interactive exploration of the geometric models we built, which can be run live at `http://www.ctralie.com/Research/`

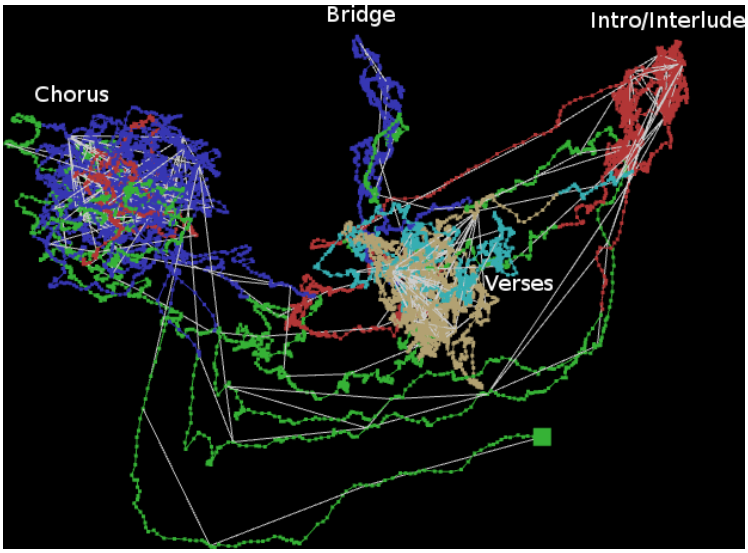


Figure 3 A screenshot from our Javascript GUI depicting our 3D visualization of the embedding of Michael Jackson's "Bad." The large green point indicates the visualization is at the beginning of the song, and this point moves through the model as the song plays.

`GeometricModels`. We show a projection of the 59 dimensional point cloud onto the subspace determined by the first three principal components. Although information is lost in the projection (e.g., in the examples we made available, about 60% of the variance is explained by the first three principal components), we can still render the geometric model that we computed in high dimensions, as well as the dimension estimates of each cluster. The GUI plays music synchronized with the geometric model, highlighting the point that has its first analysis window at the current time as it goes along, which causes it to trace out a trajectory through the geometric model over time.

Figure 3 shows a screenshot of our GUI on Michael Jackson's "Bad," with labels superimposed to mark different sections of the song. White line segments show edges in the scaffolding graph, i.e., connections between different clusters found in the cover tree. Each cluster is colored by its dimension estimate; specifically, green is dimension 1, blue is dimension 2, red is dimension 3, cyan is dimension 4, and tan is dimension 6. We notice in this example that the verses, interlude, and the chorus demonstrate areas of higher dimensionality, and transitions between them are clearly visible as 1 dimensional paths. Additionaly, a bridge, or a deviation from the main pattern of the song, constitutes its own cluster.

Acknowledgements. The first/third authors were partially supported by NSF award BIG-DATA 1444791. The first was also partially supported by NSF award WBSE 3331753. The fourth author was supported by an NSF Graduate Fellowship NSF DGF 1106401. The second author thanks the Department of Mathematics at Duke University for hosting her during fall 2015.

References

1 Mark A. Bartsch and Gregory H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pages 15–18. IEEE, 2001.

2 Paul Bendich, Ellen Gasparovic, John Harer, and Christopher Tralie. Scaffoldings and spines: Organizing high-dimensional data using cover trees, local principal component analysis, and persistent homology, 2016. http://arxiv.org/abs/1602.06245.

3 Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104. ACM, 2006.

4 Bruce P. Bogert, Michael J.R. Healy, and John W. Tukey. The quefrency analysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In *Proceedings of the symposium on time series analysis*, volume 15, pages 209–243. chapter, 1963.

5 Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, 2015.

6 George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

Visualizing Scissors Congruence

Satyan L. Devadoss[1], Ziv Epstein[2], and Dmitriy Smirnov[3]

1 **Mathematics Department, Williams College, Williams, USA**
`satyan.devadoss@williams.edu`
2 **Computer Science Department, Pomona College, Claremont, USA**
`ziv.epstein@pomona.edu`
3 **Computer Science Department, Pomona College, Claremont, USA**
`dmitriy.smirnov@pomona.edu`

Abstract

Consider two simple polygons with equal area. The Wallace–Bolyai–Gerwien theorem states that these polygons are scissors congruent, that is, they can be dissected into finitely many congruent polygonal pieces. We present an interactive application that visualizes this constructive proof.

1998 ACM Subject Classification I.3.5 [Computational Geometry and Object Modelling] Geometric Algorithms, languages and systems, K.3.1 [Computer Uses in Education] Computer-assisted instruction

Keywords and phrases polygonal congruence, geometry, rigid transformations

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.66

Category Multimedia Contribution

1 Introduction

At the dawn of the 19th century, William Wallace and John Lowry [1] posed the following:

> Is it possible in every case to divide each of two equal but dissimilar rectilinear figures, into the same number of triangles, such that those which constitute the one figure are respectively identical with those which constitute the other?

This sparked an active area of research, which culminated in the discovery of the following theorem, independently by Wallace-Lowry [1], Wolfgang Bolyai [2] and Paul Gerwien [3].

▶ **Theorem 1** (Wallace–Bolyai–Gerwien). *Any two simple polygons of equal area are scissors congruent, i.e. they can be dissected into a finite number of congruent polygonal pieces.*

David Hilbert himself recognized the importance of this theorem, including it as "Theorem 30" in his *The Foundations of Geometry* [4]. Furthermore, he posed a three-dimensional generalization of Wallace's question as number three of his famous 23 problems [5]: Given any two polyhedra of equal volume, can they be dissected into finitely many congruent tetrahedra? This problem was solved by Hilbert's own student Max Dehn, who provided (unlike the 2D case) a negative answer by constructing counterexamples [6].

The beauty of the original proof of WBG is that it is constructive: it describes an actual algorithm for constructing the polygonal pieces. To gain a deeper appreciation for this result, we built an interactive application that visualizes the algorithm in an intuitive and didactic manner. Instructors have taught the Wallace–Bolyai–Gerwien procedure using physical materials [7], and this application provides a digital analog.


© Satyan L. Devadoss, Ziv G. Epstein, and Dmitry Smirnov;
licensed under Creative Commons License CC-BY

Figure 1 Visual representation of the algorithm: using scissor cuts, a triangle becomes a rectangle, which equidecomposes to another rectangle of the fixed height.

2 Algorithm

Indeed, the original proof demonstrates that any two simple polygons of equal area are "scissors congruent." We restate the theorem in a different manner that is more suited for visualization.

▶ **Corollary 2.** *Given two simple polygons of equal area S and T, there exists a finite sequence of cuts and rigid transformations that when applied to S result in T.*

This restatement motivates a constructive proof that can be formulated with an algorithm to rigidly transform S to T:

1. Compute some triangulation of S.
2. For each triangle in the trangulation, equidecompose that triangle into a rectangle.
3. For each rectangle generated above, equidecompose that rectangle into a rectangle of some fixed width w. If the starting rectangle is wider than $2w$, cut it in half and stack the two smaller rectangles on top of one another.
4. Stack all fixed width rectangles generated above into a single rectangle.
5. Perform the above steps in reverse order to equidecompose the rectangle into some triangulation of T.

Figure 1 shows the visual interpretation of each of these geometric procedures; see [8] for a more in-depth description and analysis.

3 Implementation

Our visualization application is implemented in HTML5 and JavaScript. It runs client-side in the web browser and can be accessed at `http://dmsm.github.io/scissors-congruence/`. The interface allows the user to input her own intial and terminal polygons. It then rescales the polygons so that they are of the same area, by calculating the optimal scaling factor for each polygon such that the following two constraints are satisfied: both polygons are of equal area, and the wider of the two is not too wide that is goes off the screen. Then, according to the above algorithm, it rigidly transforms her initial polygon into the terminal polygon (see Figure 2). The application takes advantage of the JavaScript libraries jQuery, Two.js, PolyK.js and Math.js in order to render the polygons in a fast and modular way.

References

1 William Wallace and John Lowry. "Question 269". *New Series of the Mathematical Repository 3* (1814). Ed. by Thomas Leybourn, pp. 44–46.

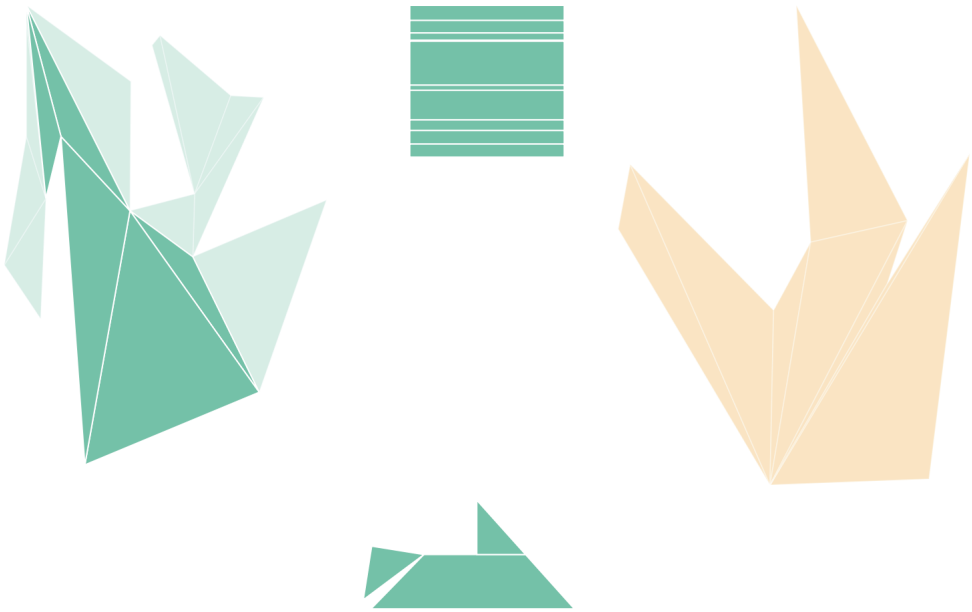


Figure 2 Screenshot of the application performing step 2 of the algorithm: a triangle of the triangulation of the initial polygon (in sea foam green) is being equidecomposed into a rectangle that will eventually be stacked (in the middle) on its way to forming the triangulation of the terminal polygon (in apricot orange).

2 Wolfgang Bolyai. *Tentamen iuventutem studiosam in elementa matheseos puræelementaris ac sublimioris methodo intuitiva evidentiaque huic propria introducendi, cum appendici triplici.* Latin. Ed. by Iosephus Kürschák, Mauritius Réthy and Béla Tőtössy de Zepethnek. 2nd ed. Vol. 2. Budapestini: Sumptibus Academiæ Scientiarum Hungaricæ, 1904.

3 Paul Gerwien. "Zerschneidung jeder beliebigen Anzahl von gleichen geradlinigen Figuren in dieselben Stücke". German. *Journal für die reine und angewandte Mathematik* 1833.10 (1833). Ed. by August Leopold Crelle, pp. 228–234. issn: 0075-4102. DOI: 10.1515/crll.1833.10.228.

4 David Hilbert. *The Foundations of Geometry.* Trans. from the German by E. J. Townsend. Chicago: The Open Court Publishing Company, 1902. vii+143. Trans. of Grundlagen der Geometrie. German. Leipzig: B. G. Teubner, 1899.

5 David Hilbert. "Mathematical problems". *Bulletin of the American Mathematical Society* 8.10 (1902), pp. 437–480. issn: 0002-9904. 10.1090/S0002-9904-1902-00923-3.

6 Max Dehn. "Ueber den Rauminhalt". Mathematische Annalen 55 (3):465–478. 1901. DOI: 10.1007/BF01448001.

7 Szilárd András and Csaba Tamási . "Teaching geometry through play." 2014. `http://www.fisme.science.uu.nl/toepassingen/28205/documents/2014_andras_geometry.pdf`

8 Ryan Kavanagh. "Explorations on the Wallace-Bolyai-Gerwien Theorem". `https://ryanak.ca/files/papers/wallace-bolyai-gerwien.pdf`

Visualization of Geometric Spanner Algorithms

Mohammad Farshi[1] and Seyed Hossein Hosseini[2]

1 **Combinatorial & Geometric Algorithms Lab., Department of Computer Science, Yazd University, Yazd, Iran**
mfarshi@yazd.ac.ir

2 **Combinatorial & Geometric Algorithms Lab., Department of Computer Science, Yazd University, Yazd, Iran**
hoseini.seyedhosein@gmail.com

Abstract

It is easier to understand an algorithm when it can be seen in interactive mode. The current study implemented four algorithms to construct geometric spanners; the path-greedy, gap-greedy, Θ-graph and Yao-graph algorithms. The data structure visualization framework (`http://www.cs.usfca.edu/~galles/visualization/`) developed by David Galles was used. Two features were added to allow its use in spanner algorithm visualization: support point-based algorithms and export of the output to Ipe drawing software format. The interactive animations in the framework make steps of visualization beautiful and media controls are available to manage the animations. Visualization does not require extensions to be installed on the web browser. It is available at `http://cs.yazd.ac.ir/cgalg/AlgsVis/`.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases geometric spanner networks, geometric spanner algorithms animations.

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.67

Category Multimedia Contribution

1 Introduction

Algorithm visualization is a wonderful tool for educational and research purposes. It helps teachers transfer key ideas and the structure of even complex algorithms in a short period of time. In research, providing examples of the output of an algorithm for a specific input can encourage deeper understanding of the algorithm and allow use of the output in presentations.

A geometric t-spanner ($t \geq 1$) for a set of points $S \subset \mathbb{R}^d$ is an undirected Euclidean graph $G = (S, E)$ such that, for any two points $p, q \in S$, there is a path in G between p and q with a length of at most $t|pq|$ where $|pq|$ is the Euclidean distance between p and q. The length of a path is the sum of all edge lengths on the path.

Several algorithms, when given point set S and $t \geq 1$ as input, compute a t-spanner on S, see [7]. The applications offered by geometric spanners means that they are commonly-used to solve problems in computational geometry as well as in fields such as bio-informatics [8].

To the best of our knowledge, there are two tools to visualize some geometric spanner algorithms:

- The applet by Specht and Smid [9] makes the Θ-graph and the geometric neighborhood graph for a given point set. This applet mainly shows the t-path and the shortest path between points in a graph.


© Mohammad Farshi and Seyed Hossein Hosseini ;
licensed under Creative Commons License CC-BY

- The applet by Aschner [3] visualizes three spanner algorithms: path-greedy, Θ-graph, and Yao-graph algorithms.[1]

The visualization presented herein consists of four spanner algorithms: path-greedy, gap-greedy, Θ-graph and Yao-graph (see [5]). This visualization had the following specifications:

- The visualization animates the steps of the algorithms. The speed of animation is adjustable.
- The visualization is browser-independent and it does not require extension to be installed on a web browser. One can use it on any modern browser, including iOS devices like the iPhone and iPad, android devices such as smart watches and TVs, and even the web browser on Kindle.
- It can export the network generated by the algorithms to .ipe format. One can open and edit the output in Ipe drawing editor, commonly-used software that is well-known in the computational geometry community.

2 Algorithms

This section briefly explains the visualized algorithms.

2.1 Path-greedy algorithm

This algorithm uses a set of points and $t \geq 1$ as input and constructs a t-spanner of the input point set. It begins with a graph with the input point set as its vertex set and an empty edge set. In the first step, it sorts all pairs of points in non-descending order of distance. It then checks all pairs of points in sorted order. For each pair (u, v) of points, if the length of the shortest path between u and v in the current graph is greater than $t \times |uv|$, then (u, v) is added to the edge set of the graph. For more details one can see [1, 7].

2.2 Θ-graph and Yao-graph algorithms

These algorithms use a point set and $t \geq 1$ (or, equivalently, integer k) as input. For any point u in the input point set, the plane is divided into k non-overlapping cones with an apex at u. The number of cones, or k, is the smallest integer such that $t \leq 1/(\cos\theta - \sin\theta)$ for $\theta = 2\pi/k$. The Yao-graph algorithm connects u to the point nearest to u in each cone. The Θ-graph algorithm, for each cone, considers a fixed line through u in the cone. It projects all the points in the cone to this line and connects u to the point for which projection to the line through u is closest to u. For more details about the Θ-graph and the Yao-graph algorithms see [4, 10].

2.3 Gap-greedy algorithm

This algorithm uses a point set and two real numbers θ and w as input. The value of θ and w must be chosen such that $0 < \theta < \pi/4$ and $0 \leq w \leq (\cos\theta - \sin\theta)/2$ where t is $1/(\cos\theta - \sin\theta - 2w)$. The gap-greedy algorithm, as for the path-greedy algorithm, sorts all ordered pairs of points non-descending order and processes them in this order. It begins with a graph with the input point set as its vertices and an empty edge set. For each pair (u, v), if there is an edge in the current graph such that the angle between (u, v) and the edge is less than θ

[1] We could not run the applet to see how it works.

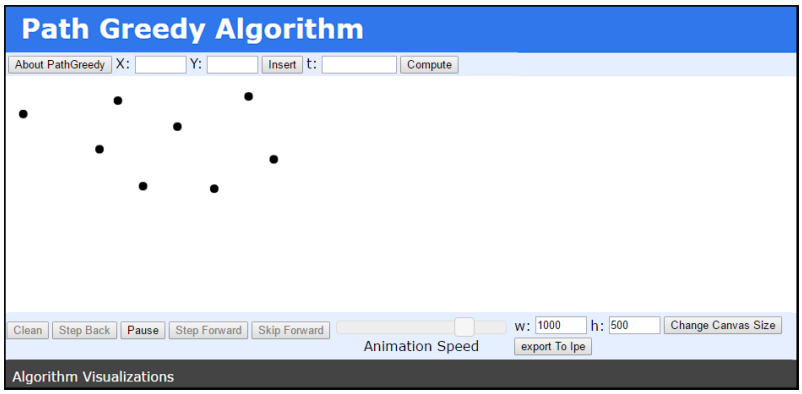


Figure 1 Screen before clicking *compute* button.

and u is close to the source of the edge or v is close to the sink of the edge, then it does not add the edge to the graph. For more details on the gap-greedy algorithm see [2, 7].

3 Usage and Implementation

This section, provides a brief description of the use of visualization and its properties and then describes the implementation of visualization.

3.1 How visualization is used

The visualization of an algorithm is based on interactive animation of the steps of the algorithm. The visualization screen includes two main control parts (Figures 1 and 2).

Algorithm controls: The user can add a point by left-clicking on the position of the point on the canvas or by inserting the coordinates of the point at the top of the page in front of X: and Y: and then clicking on the "insert" button. After adding all points, the user should enter t, k, θ, or w based on the algorithm and afterward click on the "compute" button. The steps for construction of the spanner using the algorithm appear one-by-one. One can remove any previously-inserted point by right-clicking on that point.

Animation controls: Animation controls appear at the bottom of the screen with which to manage animation using the following media controls:

- **clean:** reset the input to an empty canvas;
- **play/pause:** play and pause the animation;
- **step back/forward**: go one step backward/forward in the animation;
- **skip forward:** go to the final step of the algorithm.

A slider control is provided with which to manage animation speed and a button by which to export the output to Ipe drawing software. The user can change the size of the main screen (width and height) by changing the values and clicking on the "Change Canvas Size" button. Animation controls are the same for all algorithms.

3.2 Implementation details

Visualization uses the "structure visualization framework" developed by David Galles [6]. Any algorithm visualization set includes an HTML file and a javascript file. The algorithm

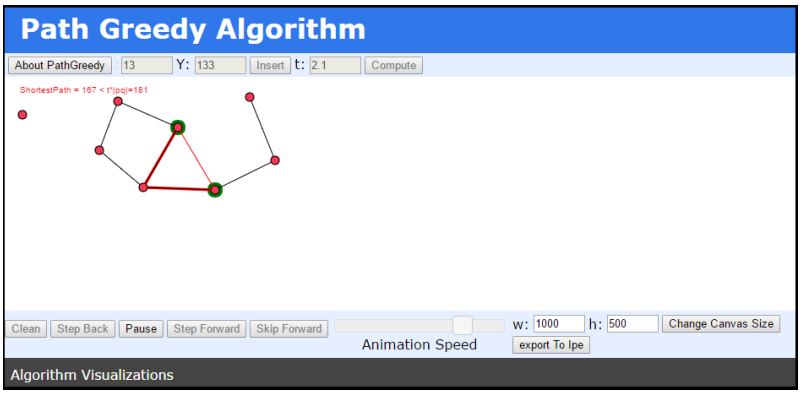


Figure 2 Screen after clicking *compute* button.

is coded in the javascript file. The body of programming is based on the canvas tag in html5 that is manipulated using javascript instructions. The canvas tag can be used to draw a shape and for animation via javascript. Paths, boxes, circles, and texts can be drawn in canvas and images can be added.

Two features have been added to the framework to allow its use in spanner algorithm visualization: support of point-based algorithms and export of the output to Ipe Drawing software format. The first is used to obtain the input point set from the user and the second to export the result of the algorithm to the input point set as an .ipe file.

Acknowledgements. The authors would like to thank Davood Bakhshesh for discussions on both the algorithms and their implementation.

References

1 Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.
2 Sunil Arya and Michiel Smid. Efficient construction of a bounded-degree spanner with low weight. *Algorithmica*, 17(1):33–54, 1997.
3 Rom Aschner. Geometric spanners applet. `http://www.cs.bgu.ac.il/~romas/greedy.html`.
4 Ken Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. of the 19t Annual ACM Symp. on Theory of Computing*, pages 56–65. ACM, 1987.
5 Mohammad Farshi and Seyed Hossein Hosseini. Geometric spanner algorithms visualizations. `http://cs.yazd.ac.ir/cgalg/AlgsVis/`.
6 David Galles. Data structure visualizations. `http://www.cs.usfca.edu/~galles/visualization/`.
7 Giri Narasimhan and Michiel Smid. *Geometric spanner networks*. Cambridge University Press, 2007.
8 Daniel Russel and Leonidas J. Guibas. Exploring protein folding trajectories using geometric spanners. *Pacific Symposium on Biocomputing*, pages 40–51, 2005.
9 Petra Specht and Michiel Smid. Visualization of spanners. `http://isgwww.cs.uni-magdeburg.de/tspanner/spanner.html`.
10 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.

Path Planning for Simple Robots using Soft Subdivision Search*

Ching-Hsiang Hsu[1], John Paul Ryan[2], and Chee Yap[3]

1 **Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, USA**
chhsu@nyu.edu
2 **Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, USA**
john.ryan@nyu.edu
3 **Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, USA**
chee.yap@nyu.edu

Abstract

The concept of ε-exact path planning is a theoretically sound alternative to the standard exact algorithms, and provides much stronger guarantees than probabilistic or sampling algorithms. It opens the way for the introduction of **soft predicates** in the context of subdivision algorithm. Taking a leaf from the great success of the **Probabilistic Road Map** (PRM) framework, we formulate an analogous framework for subdivision, called **Soft Subdivision Search** (SSS). In this video, we illustrate the SSS framework for a trio of simple planar robots: disc, triangle and 2-links. These robots have, respectively, 2, 3 and 4 degrees of freedom. Our 2-link robot can also avoid self-crossing. These algorithms operate in realtime and are relatively easy to implement.

1998 ACM Subject Classification F2.2 Geometrical problems and computations, I.2.9 Autonomous vehicles

Keywords and phrases Path Planning, Configuration Space, Soft Predicates, Resolution Exactness, Subdivision Search

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.68

Category Multimedia Contribution

1 Introduction

Path Planning is a fundamental task in robotics [4, 1]. There are three main approaches: first, we have the Exact Approach which can, in principle, solve any algebraic planning problem. But practitioners tend to implement exact algorithms using machine approximations; then it is no longer clear what the guarantees of exact algorithms mean. The book [2] shows how path planning (Section 9.3), among other exact algorithms of computational geometry, may be implemented exactly (Section 1.3). Exact algorithms are impractical except for the simplest cases. Another approach is based on sampling: we refer to **Probabilistic Road Map** (PRM) [3] as the main representative. This Sampling Approach has dominated the field in the last twenty years, but its central problem is inability to halting (couched as "narrow passage problem"). Finally, we have the Subdivision Approach. This is the oldest among

* This work was partially supported by NSF Grant #CCF-1423228.

© Ching-Hsiang Hsu, John Paul Ryan, and Chee Yap;
licensed under Creative Commons License CC-BY

the three approaches, and remains popular with practitioners: our work falls under this category. In [6, 7, 8], we introduced the concept of **resolution-exactness** as the theoretical foundation for path planning that side-steps exact computation. This opens the way for the introduction of **soft predicates**, replacing the usual predicates that control the logic of all geometric algorithms (e.g., [2]).

Subdivision methods share with Sampling methods many advantages over Exact methods. They both lead to algorithms that are easy to implement and to modify. Implementability is highly valued in a practical area like robotics. Modifiability is also important in practice because we typically deploy such algorithms in systems where non-algorithmic considerations must be accounted for. Both methods allow the possibility of discovering paths *before* the entire free configuration space has been fully explored.

This video is a demonstration of these properties. Specifically:

(a) Our algorithms are resolution-exact: in particular, it will halt on all inputs. When it returns `NO-PATH`, it guarantees there is no path of a certain specified clearance.

(b) Our algorithms are real-time: there is no pre-processing of the inputs, nor any implicit parameter selections (unlike PRM which need some additional parameters).

(c) The flexibility of our framework is seen in its ability to support a variety of global search search strategies. We implemented these strategies: random, Breadth First Search (BFS), Greedy Best First (GBF), "Voronoi heuristic".

(d) The same framework supports three distinct robots (disc, triangle, 2-link). Moreover, each of these robots is parameterizable: we can freely change the radius r_0 of the disc, the lengths (a, b, c) of the triangle, the lengths (ℓ_1, ℓ_2) of the two links.

(e) The performance of our algorithms is adaptive (not controlled by the worst case complexity). It can easily handle arbitrarily complex environments, e.g., environments involving hundreds of triangles.

Limitations. Our current software is implemented using standard machine double precision. It is not hard to extend our software to allow arbitrary precision by incorporating any standard big-float number package (this is especially easy within our Core Library [9]). It is well-known that such a step would take a performance hit. Thus our experimental claims are all based on machine precision computation, within the "normal limits" of experimental validation. What are these normal limits? Typically, the physical environment lies in a 512 square (or cube) domain. Thus our claim of "realtime performance" is within such limits. This is consistent with current practice in robotics algorithms. In a future work, we plan to carry out the error analysis to show the extent to which our results can be guaranteed with machine precision. Although the 3 robots in this video fall within our SSS framework, they remain separate pieces of software. In the future, we plan to create a common SSS framework in which all 3 robots (among others) could be supported.

2 FindPath using Soft Subdivision Search

Here we briefly summarize the SSS Framework [7]. Assume a fixed robot R_0 in physical space $\mathbb{R}^k$ (typically $k = 2, 3$) with configuration space $C_{space} = C_{space}(R_0)$. In this video review, we will see three kinds of robots R_0 as illustrated in Figure 1: their configuration spaces have 2, 3 and 4 (respectively) degrees of freedom.

Our demos will show two kinds of 2-link robot, depending on whether we allow or forbid the 2 links to cross each other. A configuration may be given by $(x, y, \theta_1, \theta_2) \in \mathbb{R}^2 \times \mathcal{T}^2 = C_{space}$ ($\mathcal{T}^2 = S^1 \times S^1$ is the torus). Non-crossing means that $\theta_1 = \theta_2$ is forbidden. More generally,

Figure 1 Three simple robots.

we can forbid a band $|\theta_1 - \theta_2| \leq \delta$. This geometry is interesting since such bands do not disconnect the torus, and seems novel.

The (exact) **path planning problem** for R_0 is this: *given any polyhedral obstacle set* $\Omega \subseteq \mathbb{R}^k$*, and start and goal configurations* $A, B \in C_{space}(R_0)$*, to find an* Ω*-avoiding path from* A *to* B *if one exists, and return* `NO-PATH` *otherwise.* In **resolution-exact path planning**, we are given two additional input: a resolution $\varepsilon > 0$ and a region-of-interest $B_0 \subseteq C_{space}(R_0)$. There is a constant $K > 1$ independent of the inputs such that the algorithm always halts and either outputs `NO-PATH` or output an Ω-avoiding path, subject to:

(P1) If all Ω-avoiding paths in B_0 have clearance $< \varepsilon/K$, it must return `NO-PATH`.

(P2) If there exists an Ω-avoiding path in B_0 with clearance $> K\varepsilon$, it must return a path.

Note that (P1) and (P2) do not cover all possibilities: if the maximum clearance of an Ω-avoiding path is in the gap $(\varepsilon/K, K\varepsilon)$, our algorithm's output need not be deterministic.

Fix an obstacle set $\Omega \subseteq \mathbb{R}^k$. We define the **clearance** of a configuration $\gamma \in C_{space}$ to be the separation between robot in the "pose" γ and Ω. If the clearance is positive, we say γ is **free**. The **free space** $C_{free} = C_{free}(R_0, \Omega)$ comprises all such free configurations. We say γ is **semi-free** if it is on the boundary of C_{free}.

The search for path is restricted to B_0, which will be recursively split into subboxes $B \subseteq B_0$. We focus on box predicates $\widetilde{C} : B \mapsto \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$. The box predicate $\widetilde{C}$ is **conservative** if

$$\begin{cases} \widetilde{C}(B) = \texttt{FREE} & \text{implies} \quad B \subseteq C_{free} \\ \widetilde{C}(B) = \texttt{STUCK} & \text{implies} \quad B \cap C_{free} = \emptyset \end{cases}$$

A maximally conservative predicate is trivial – it always outputs `MIXED`. A minimally conservative predicate is called **exact** – it outputs `FREE` or `STUCK` whenever possible. We say $\widetilde{C}$ is **convergent** if for any monotone sequence $\{B_i : i \geq 0\}$ that converges to a point $p = \lim_{i\to\infty} B_i$, we have $\widetilde{C}(B_i) = C(p)$ for i large enough. Here, $C(p)$ denotes the exact predicate with $C(p) = \texttt{MIXED}$ iff p is semifree. We say $\widetilde{C}$ is a **soft predicate** if it is conservative and convergent. The design of soft predicates is of considerable interest, and illustrated in [6, 8]; there, we further show how soft predicates with the additional property of **effectivity** leads to resolution-exact path planners, called **SSS planners**. Briefly, the main loop of SSS planners is controlled by a priority queue Q containing `MIXED` boxes. While Q is non-empty, we remove a box B from Q, subdivide B into children, classify them and put the `MIXED` boxes back into Q. We maintain the connected components of all the free boxes using the well-known **Union-Find** data structure; two boxes are unioned if they are adjacent. More precisely, the SSS planner has three plug-in subroutines:

(S1) A soft predicate $\widetilde{C}(B)$.

(S2) A global search strategy to determine the priority of boxes in Q: this is encoded in the Q.`getNext`() method which returns a box from Q. Correctness of SSS does not depend on the global strategy.

(S3) A subroutine `Expand`(B) that could fail: first, remove B from Q. If B has width $< \varepsilon$, then `Expand`(B) fails. Otherwise, we subdivide B into a set S$plit(B)$ with two or more subboxes. For each $B' \in$ S$plit(B)$, we compute $\widetilde{C}(B')$. If $\widetilde{C}(B') =$ `FREE`, we add B' into a Union-Find structure, and union B' with each adjacent box B'' already in the structure. If $\widetilde{C}(B') =$ `MIXED`, we put B' into Q.

Finally, we put together these 3 subroutines. Let $Box(\alpha)$ denote a smallest subdivision box that contains α. We can keep track of $Box(\alpha)$ and $Box(\beta)$.

SSS FindPath:
Input: Configurations α, β, tolerance $\varepsilon > 0$, box $B_0 \subseteq C_{space}$.
Output: Either path from α to β, or `NO-PATH`.
Initialize $Q = \{B_0\}$.
1. While ($Box(\alpha) \neq$ `FREE`)
   If (`Expand`$(Box(\alpha))$ fails), Return(`NO-PATH`).
2. While ($Box(\beta) \neq$ `FREE`)
   If (`Expand`$(Box(\beta))$ fails), Return(`NO-PATH`).
3. While ($Find(Box(\alpha)) \neq Find(Box(\beta))$)
   If Q is empty, Return(`NO-PATH`)
   `Expand`$(Q.getNext())$
4. Compute a channel P from $Box(\alpha)$ to $Box(\beta)$.
   Return a path in this channel.

The **channel** in Step 4 is a sequence $(B_1, \ldots, B_m)$ of free boxes where B_i, B_{i+1} are adjacent. The expansion technique for our 2-link robot is non-standard in order to achieve our performance (see [5]).

Acknowledgments. The original software for the disc and triangle robot was implemented as part of Cong Wang's PhD thesis and reported in [6]. The 2-link robot was implemented in Luo's Masters thesis and reported in [5]. The original graphics was written in OpenGL 2.1 using GLUT/GLUI libraries. In the summer of 2015, Bryant Curto and John Ryan (supported by a departmental Undergraduate Summer Research Fellowship) re-implemented the disc and 2-link software in the Qt IDE (OpenGL 4.x) resulting in much faster graphics. Ching-Hsiang Hsu re-implemented the triangle software in Qt.

References

1 H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion.* MIT Press, Boston, 2005.

2 Dan Halperin, Efi Fogel, and Ron Wein. *CGAL Arrangements and Their Applications.* Springer-Verlag, Berlin and Heidelberg, 2012.

3 Lydia Kavraki, P. Švestka, Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, 1996.

4 Steven M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, 2006.

5 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)*, vol. 107 of *Springer Tracts in Advanced Robotics (STAR)*, pp. 353–370, 2015.

6 Cong Wang, Yi-Jen Chiang, and Chee Yap. On Soft Predicates in Subdivision Motion Planning. *Comput. Geometry: Theory and Appl.*, 48(8):589–605, September 2015.

7 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. Best Paper Award. In arXiv:1402.3213.

8 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. Invited. 9th FAW. Guilin, China. Aug 3-5, 2015.

9 Jihun Yu, Chee Yap, Zilin Du, Sylvain Pion, and Herve Bronnimann. Core 2: A library for Exact Numeric Computation in Geometry and Algebra. In *Proc. 3rd ICMS*, pages 121–141. Springer, 2010. LNCS No. 6327.

Exploring Circle Packing Algorithms*

Kevin Pratt[1], Connor Riley[2], and Donald R. Sheehy[3]

1 **University of Connecticut, Storrs, USA**
kevin.pratt@uconn.edu
2 **University of Connecticut, Storrs, USA**
connor.riley@uconn.edu
3 **University of Connecticut, Storrs, USA**
don.r.sheehy@gmail.com

Abstract

We present an interactive tool for visualizing and experimenting with different circle packing algorithms. The source code can be found at: https://github.com/interl0per/CirclePacking

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Geometrical problems and computations

Keywords and phrases Computational Geometry, Processing, Javascript, Visualization, Incremental Algorithms

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.69

Category Multimedia Contribution

1 Introduction

The Koebe Embedding Theorem provides a wonderful link between questions of geometry, topology, combinatorics, and complex analysis. It states that every planar graph can be realized as the intersection graph of a collection of interior disjoint circles in the plane. That is, the vertices correspond to circles and two circles are tangent if and only if their corresponding vertices share an edge. Such a graph representation is called a circle packing. They have found application in computational geometry such as in a proof of the planar separator theorem [4] or for bounding eigenvectors of graph Laplacians [3].

Technically, circle packings exist for any planar graph, but it is often simpler algorithmically to work with 3-connected, triangulated planar graphs and we will assume such graph throughout. There are several known algorithms for computing a circle packing from a given planar graph. Early proofs of the existence of circle packings were non-constructive, but the work of Colin de Verdiere[2] and others [5, 1] showed that circle packings can be computed by convex optimization, ushering in several different iterative algorithms. A good introduction to the mathematical and algorithmic aspects of circle packings can be found in the book by Stephenson [6].

As shown in the work of Bobenko and Springborn, there are several different energy functionals that one could minimize to arrive at a circle packing [1]. However, there are simple heuristics that are not known to give correct algorithms, yet still seem to produce good circle packings. We present a tool designed to make it easy to mix and match circle

* This work was partially supported by the National Science Foundation under grants CCF-1525978 and CCF-1464379.

© Kevin Pratt, Connor Riley, Donald R. Sheehy;
licensed under Creative Commons License CC-BY

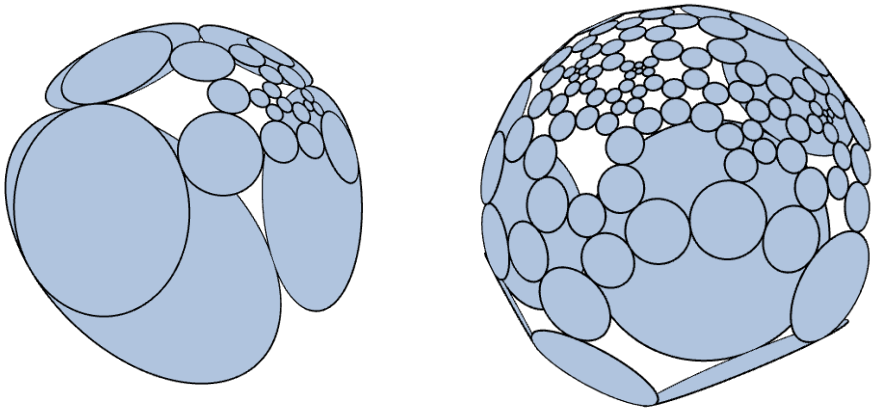


Figure 1 Stereographic projection of a circle packing.

packing heuristics. One can quickly generate a triangulated planar graph and then apply different heuristics to move it towards a circle packing representation.

2 Background

Weighted Delaunay Triangulation. Circle packings are closely related to several other well-known topics in computational geometry. The most notable, being *weighted Delaunay triangulations*. This is the dual to the so-called power diagram, a Voronoi diagram on the points where the (squared) distance to a circle c with center p and radius r is defined to be $\pi_c(x)^2 := \|x - p\|^2 - r^2$. A circle packing representation of a graph realizes the graph as the weighted Delaunay triangulation where the vertices are the centers of the disks and the weights are the radii.

Equilibrium Stresses. A *stress* is an assignment of real numbers to edges that may be interpreted as spring constants. By Hook's Law, the magnitude of the force exerted by a spring is the spring constant times its length. A stress is an *equilibrium stress* if for each vertex, the sum of the forces exerted on it by all its incident edges is exactly zero.

Weighted Delaunay triangulations have another interpretation as projections of convex polyhedra in $\mathbb{R}^3$. The Maxwell-Cremona Correspondence implies that projections of convex polyhedra always admit an equilibrium stress that is nonnegative on all edges except those of the convex hull. Thus, an alternative (dual) perspective on circle packings is that rather than looking for the correct radii, one could look for the correct equilibrium stress. Given a stress, one finds the embedding by Tutte's algorithm. This involves solving one system of linear equations derived from the graph.

Stereographic Mappings and Möbius Tranformations. Weighted Delaunay triangulations can also be realized as stereographic projections of convex polyhedra. In the case of circle packings this polyhedron is sometime called the Koebe polyhedron. It has the interesting property that the edges are all tangent to a sphere.

A Möbius Transformation of a plane can be obtained by performing the stereographic projection of the plane onto a sphere, then rotating or moving the sphere and then performing

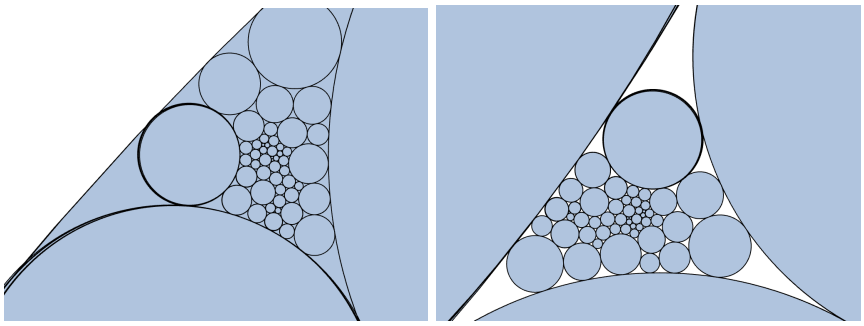


Figure 2 Two Möbius transformations of a circle packing.

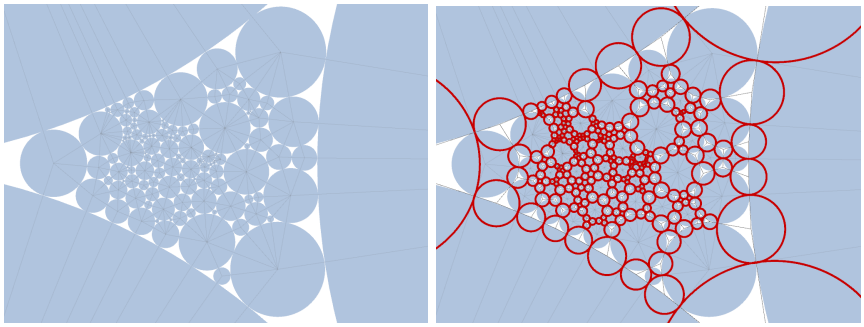


Figure 3 A packing and it's dual.

the stereographic projection back onto the plane. Formally, a Möbius Transformation is a rational function defined on the extended complex plane $\hat{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ of the form:

$$f(z) = \frac{az + b}{cz + d} \tag{1}$$

where $z \in \hat{\mathbb{C}}$ is a complex variable and $a, b, c, d \in \hat{\mathbb{C}}$ are complex numbers such that $ad - bc \neq 0$. Circle packings in the plane are unique up to Möbius Transformation.

Dual Packings. Circle packings of triangulations always admit a *dual packing* (Fig 3). By pressing the `'d'` key, the user can see the dual packing.

3 Interactive Input

One immediate challenge to working interactively with circle packings is the need for input triangulations. We would like to be able to produce triangulations with a minimum of effort. The most natural approach is to draw circles and produce the weighted Delaunay triangulation of the circles. Input is given by clicking a center and dragging to set the radius of an input disk. As new points are added, we incrementally compute the weighted Delaunay triangulation. An advantage of this approach is that it is very easy to draw approximate circle packings. This allows one to experiment with examples where one has a clear idea of the expected output.

As a Möbius transformations will give new circle packings of the input graph, we allow the user to transform the plane to explore different packings. These transformations can be viewed by hitting the `'spacebar'` key twice to change views and then moving the mouse.

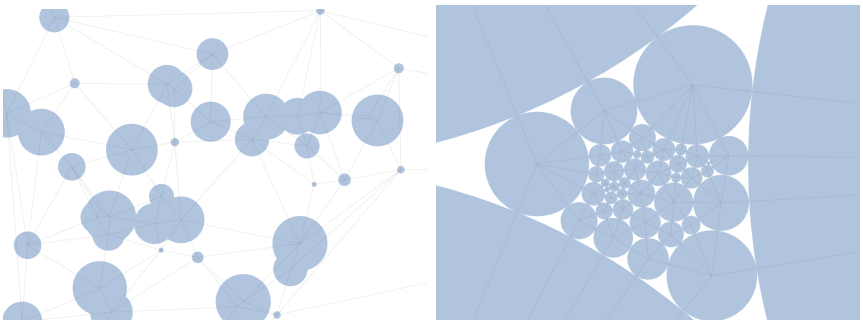


Figure 4 An input of circles (left) and the circle packing output (right).

If the user presses the `'spacebar'` key again, then dragging with the mouse will instead rotate a 3D view of the circle packing mapped stereographically onto a sphere.

4 Algorithms

Currently we have two different algorithms implemented. The first is based on the algorithm from Stephenson [6]. Given any set $\{r_1, \ldots, r_n\}$ of radii, it is possible to compute the angles of the triangles using the law of cosines. The final radii are those for which the angles at any vertex sum to exactly 2π. Thus, the algorithm searches for the radii of the disks by making small incremental updates to the radii, increasing the radius if the angle sum is more than 2π and decreasing the radius of the angle sum is less than 2π.

The second algorithm is based on spring embeddings of graphs such Tutte's algorithm. In this algorithm, the embedded graph is interpreted as a system of springs. This "force-directed" algorithm is not known to terminate, but it seems to be quite effective on small examples.

5 Future Work

We are actively developing new features to provide both new ways to visualize the packings as well as new heuristics. The current implementation supports two of the major paradigms in circle packing algorithms. Next, we will consider algorithms based on discrete Ricci flow.

References

1 Alexander I. Bobenko and Boris A. Springborn. Variational principles for circle patterns and koebe's theorem. *Transactions of the American Mathematical Society*, 356(2):659–689, 2003.

2 Yves Colin de Verdière. Un principe variationnel pour les empilements de cercles. *Inventiones mathematicae*, 104(1):655–669, 1991.

3 Jonathan A. Kelner. Spectral partitioning, eigenvalue bounds, and circle packings for graphs of bounded genus. *SIAM J. Comput.*, 35(4):882–902, 2006.

4 Gary L. Miller, Shang-Hua Teng, William Thurston, and Stephen A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *J. ACM*, 44(1):1–29, 1997.

5 Bojan Mohar. A polynomial time circle packing algorithm. *Discrete Mathematics*, 117(1–3):257–263, 1993.

6 Kenneth Stephenson. *Introduction to Circle Packing.* Cambridge University Press, New York, New York, 2005.

The Explicit Corridor Map: Using the Medial Axis for Real-Time Path Planning and Crowd Simulation

Wouter van Toll[1], Atlas F. Cook IV[2], Marc J. van Kreveld[3], and Roland Geraerts[4]

1 **Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands**
W.G.vanToll@uu.nl
2 **Information and Computer Sciences Department, University of Hawaii at Manoa, Hawaii**
acook4@hawaii.edu
3 **Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands**
M.J.vanKreveld@uu.nl
4 **Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands**
R.J.Geraerts@uu.nl

Abstract

We describe and demonstrate the Explicit Corridor Map (ECM), a *navigation mesh* for path planning and crowd simulation in virtual environments. For a bounded 2D environment with polygonal obstacles, the ECM is the *medial axis* of the free space annotated with nearest-obstacle information. It can be used to compute short and smooth paths for disk-shaped characters of any radius. It is also well-defined for multi-layered 3D environments that consist of connected planar layers. We highlight various operations on the ECM, such as dynamic updates, visibility queries, and the computation of paths (*indicative routes*).

We have implemented the ECM as the basis of a real-time *crowd simulation framework* with path following and collision avoidance. Our implementation has been successfully used to simulate real-life events involving large crowds of heterogeneous characters. The enclosed *demo application* displays various features of our software.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, I.6.8 Types of Simulation

Keywords and phrases Medial axis, Navigation mesh, Path planning, Crowd simulation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.70

Category Multimedia Contribution

1 Introduction

Many simulations, games, and other applications feature virtual characters that need to plan and traverse paths through a complicated environment in real-time. Our research focuses on *path planning and crowd simulation* for disk-shaped characters that move along walkable surfaces. A *navigation mesh* subdivides these surfaces into regions for path planning purposes. We describe the Explicit Corridor Map (ECM) [1, 8], a navigation mesh that allows path


© Wouter van Toll, Atlas F. Cook IV, Marc J. van Kreveld, and Roland Geraerts;
licensed under Creative Commons License CC-BY

planning for characters of an arbitrary radius. This document describes the theoretical background of our demo application, as well as a number of implementation details. Various parts of this text have been extracted from a journal article that is currently in review.

2 Definitions

Let a *2D environment* $\mathcal{E}$ be a bounded two-dimensional planar environment with polygonal obstacles. The *obstacle space* $\mathcal{E}_{obs}$ is the union of all obstacles, including the boundary of the environment. The complement of $\mathcal{E}_{obs}$ is the *free space* $\mathcal{E}_{free}$. The *complexity* of $\mathcal{E}$ is the number of vertices n required to define $\mathcal{E}_{obs}$ using interior-disjoint simple polygons.

The Explicit Corridor Map (ECM) is based on the *medial axis* (MA) [6], which is related to the Voronoi diagram of line segment sites [5]. Various definitions of the MA exist; we define it as the closure of all points in $\mathcal{E}_{free}$ that have at least two distinct equidistant nearest points in $\mathcal{E}_{obs}$, in terms of 2D Euclidean distance. An example is shown in Figure 1a.

The ECM is an MA annotated with nearest-obstacle information. More precisely, it is an undirected graph $G = (V, E)$ where V is the set of MA vertices of degree 1, 3, or higher. Each edge $e_{ij} \in E$ is a sequence of MA arcs between two vertices v_i and v_j in V. An edge is represented by $n' \geq 2$ *bending points* $bp_0, \ldots, bp_{n'-1}$ where $bp_0 = v_i$, $bp_{n'-1} = v_j$, and $bp_1, \ldots, bp_{n'-2}$ are the degree-2 MA vertices inbetween. Each bending point bp_k on an edge stores its two nearest obstacle points l_k and r_k on the left and right side of the edge. An example of an ECM is shown in Figure 1b.

The bending points and their annotations induce a subdivision of $\mathcal{E}_{free}$ into polygonal *ECM cells*. Therefore, the ECM serves as a navigation mesh. Similarly to the medial axis, the ECM can be constructed in $\mathcal{O}(n \log n)$ time and requires $\mathcal{O}(n)$ space.

We have extended the MA and ECM to *multi-layered environments* that consist of 2D layers connected by k line segment connections. The MA is based on distances projected onto a common ground plane. The multi-layered ECM can be computed in $\mathcal{O}(kn \log n)$ time by initially treating the connections as obstacles and then opening them incrementally [8]. An article with improved definitions, proofs, and algorithms is currently under submission.

3 Operations

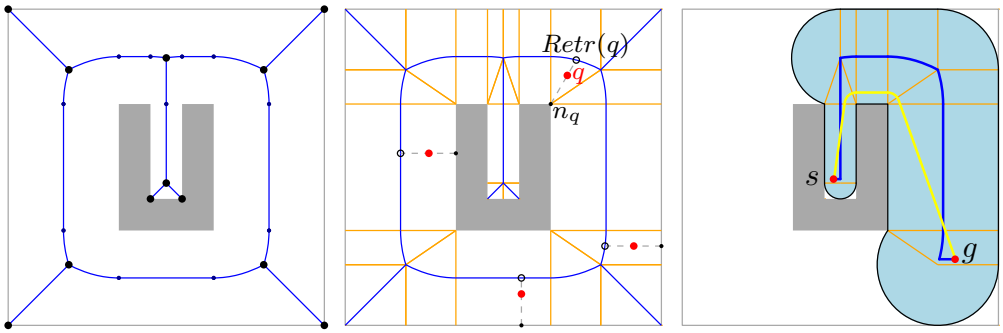
Given a query point q, we can find the ECM cell that contains q in $\mathcal{O}(\log n)$ time using a standard point location data structure. Given this cell, it takes $\mathcal{O}(1)$ time to compute the *nearest obstacle point* n_q to q and the retraction $Retr(q)$ of q, which is the point where the half-line from n_q through q first intersects the medial axis.

When an obstacle is inserted or deleted during the simulation, the ECM can be *dynamically updated* in real-time [9]. Our algorithms for dynamic updates are based on algorithms for site insertions and deletions in Voronoi diagrams. Their running times depend on the number of neighboring obstacles for the dynamic obstacle.

We can also use the ECM to efficiently compute the *visibility polygon* of a query point, or to check if two points are *mutually visible*. These algorithms automatically work in multi-layered environments because they only rely on the connectivity between ECM cells.

4 Path Planning and Crowd Simulation

To compute a global path from a point s to a point g, we retract the query points and then use A* search [2] to find a shortest path from $Retr(s)$ to $Retr(g)$ on the medial axis. Because



(a) Medial axis **(b)** ECM / Retractions **(c)** Path planning

Figure 1 The ECM and its operations in a simple 2D environment. a Obstacles are shown in gray. The medial axis is shown in blue. Degree-2 vertices are shown as small dots; other vertices are shown as large dots. b The ECM adds nearest-obstacle annotations, shown as orange line segments. For any query point q (red dot), we can use the ECM to compute the nearest obstacle point n_q (black dot) and the retraction $Retr(q)$ (circle). c To plan a path from s to g, we first compute a shortest path on the medial axis. This induces a corridor of free space (light blue) in which we can compute e.g. a short route with clearance to obstacles (shown in yellow).

the ECM stores nearest-obstacle information, we can dynamically ignore edges that are too narrow for a character to use. Therefore, we can use the ECM to plan paths for characters of an arbitrary radius. The resulting medial axis path induces a *corridor* of free space around the medial axis. Within this corridor, we can compute various types of geometric paths, such as a shortest path with clearance to obstacles [1]. Figure 1c summarizes this.

A geometric path can be used as an *indicative route* for the character to follow smoothly during the simulation. In each simulation step, the character computes a *preferred* velocity that steers it further along the indicative route. Next, the character converts this preferred velocity to an *actual* velocity that avoids collisions with other characters. For more details on these simulation components, we refer the reader to an overview paper [7].

5 Implementation

We have implemented a framework that combines the ECM with algorithms for path planning, path following, collision avoidance, and other tasks related to crowd simulation. The software was written in C++ in Visual Studio 2013, but the code is platform-independent.

5.1 Computing the ECM

To robustly compute the ECM, we have integrated two different libraries for computing Voronoi diagrams of line segment sites: Vroni [3] and a package of Boost (`http://www.boost.org/`). First, we compute the Voronoi diagram using either library. Next, traverse it while adding nearest-obstacle annotations and removing vertices and edges that are not part of the medial axis. Finally, we remove all connected components that lie inside obstacles. For the multi-layered ECM, we first perform these steps for each layer separately (possibly in parallel), and then we open the connections one by one.

Since the Boost Voronoi library requires integer coordinates as input, we multiply all coordinates by 10,000 and round them to the nearest integer. For convenience and comparative purposes, we use these rounded coordinates in Vroni as well. Since we use meters as units,

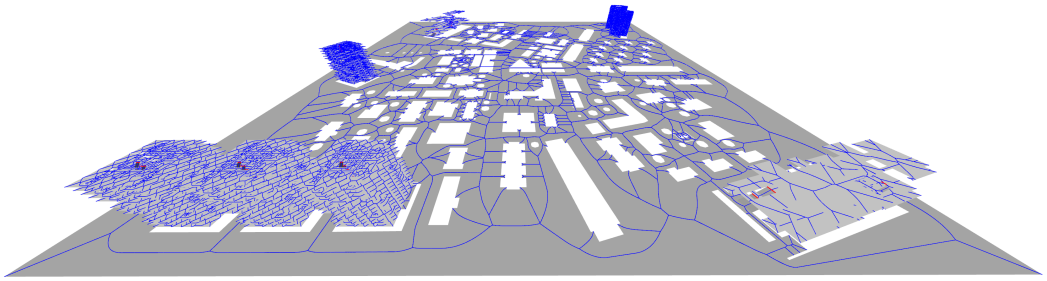


Figure 2 A large multi-layered city and its medial axis, computed using our ECM implementation.

this scaling implies that we represent all coordinates within a precision of 0.1 millimeters.

We have also created an approximating GPU-based construction algorithm [4, 1], but this approach cannot guarantee certain properties such as topological correctness, accurate positions of vertices, and full coverage of the free space. This makes many other ECM algorithms very difficult to implement, most notably our algorithms for dynamic updates and the construction algorithm of the multi-layered ECM.

Experiments show that our Vroni-based implementation is faster than the Boost-based version. However, the advantage of Boost is that it is thread-safe, which allows us to compute multi-layered ECMs very efficiently by computing the ECMs of all layers in parallel. Figure 2 shows an example of a huge multi-layered environment. Its ECM, with over 40,000 vertices and 120,000 bending points, was computed in under 14 seconds using Vroni.

5.2 Demo Application and Other Results

The enclosed demo application automatically loads four 2D environments and computes their ECMs using Boost. In the first environment, it creates a character at a random position and lets it compute and follow an indicative route to a random goal position. Whenever the character reaches its goal, it computes a new route to a new random position. The user can switch between environments, assign new goal positions, change the radius of the character (which may affect the accessibility of certain ECM edges), and change the preferred distance to obstacles (which affects the indicative route, but not the corridor in which it lies).

The full version of our framework can simulate tens of thousands of heterogeneous characters in real-time using multi-threading and an efficient subdivision of the simulation loop into substeps [7]. The framework can also be linked as a DLL to other software, such as the Unity3D game engine (`http://www.unity3d.com/`).

References

1 R. Geraerts. Planning short paths with clearance using Explicit Corridors. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1997–2004, 2010.

2 P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

3 M. Held. VRONI and ArcVRONI: Software for and applications of Voronoi diagrams in science and engineering. In *Proceedings of the 8th International Symposium on Voronoi Diagrams in Science and Engineering*, pages 3–12, 2011.

4 K.E. Hoff III, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast computation of generalized Voronoi diagrams using graphics hardware. *International Conference on Computer Graphics and Interactive Techniques*, pages 277–286, 1999.

5 D.T. Lee and R.L. Drysdale III. Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–87, 1981.

6 F. Preparata. The medial axis of a simple polygon. In *Mathematical Foundations of Computer Science*, volume 53, pages 443–450. Springer, 1977.

7 W. van Toll, N. Jaklin, and R. Geraerts. Towards believable crowds: A generic multi-level framework for agent navigation. In *ASCI.OPEN*, 2015.

8 W.G. van Toll, A.F. Cook IV, and R. Geraerts. Navigation meshes for realistic multi-layered environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3526–3532, 2011.

9 W.G. van Toll, A.F. Cook IV, and R. Geraerts. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds*, 23(6):535–546, 2012.

High Dimensional Geometry of Sliding Window Embeddings of Periodic Videos

Christopher J. Tralie

Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA
chris.tralie@gmail.com

Abstract

We explore the high dimensional geometry of sliding windows of periodic videos. Under a reasonable model for periodic videos, we show that the sliding window is necessary to disambiguate all states within a period, and we show that a video embedding with a sliding window of an appropriate dimension lies on a topological loop along a hypertorus. This hypertorus has an independent ellipse for each harmonic of the motion. Natural motions with sharp transitions from foreground to background have many harmonics and are hence in higher dimensions, so linear subspace projections such as PCA do not accurately summarize the geometry of these videos. Noting this, we invoke tools from topological data analysis and cohomology to parameterize motions in high dimensions with circular coordinates after the embeddings. We show applications to videos in which there is obvious periodic motion and to videos in which the motion is hidden.

1998 ACM Subject Classification I.2.10 Video Analysis, I.5.4 Pattern Recognition: Waveform Analysis, I.4.10 Image Representation: Multidimensional

Keywords and phrases Video Processing, High Dimensional Geometry, Circular Coordinates, Nonlinear Time Series

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.71

Category Multimedia Contribution

1 Video Dynamics And Subspace Geometry

1.1 Video Delay Embeddings

We use a sliding window through time, also known as a "delay embedding" in the dynamical systems literature [3], to capture the dynamics of a periodic video. More precisely,

▶ **Definition 1.** Given a discrete video $X[n] \in \mathbb{R}^{W \times H}$, where $W \times H$ are the dimensions of each frame in pixels and $n \in \mathbb{Z}^+$ is a discrete time index, the video is *periodic* if there exists a $T \in \mathbb{Z}^+$ so that $X[n] = X[n+T]$ for all n.

▶ **Definition 2.** Given a video $X[n] \in \mathbb{R}^{W \times H}$ and a window size M, *delay embedding* $Y[n]$ is formed as

$$Y[n] = \begin{bmatrix} X[n] \\ X[n+1] \\ \vdots \\ X[n+(M-1)] \end{bmatrix} \in \mathbb{R}^{W \times H \times M} \tag{1}$$

As n varies, $Y[n]$ traces out a samples of a 1-manifold in $\mathbb{R}^{W \times H \times M}$, though for a video of F frames, it lies on a $F-1$ dimensional subspace, which we exploit to speed up processing. Figure 1 shows a pictorial depiction of this scheme.

© Christopher J. Tralie;
licensed under Creative Commons License CC-BY
32nd International Symposium on Computational Geometry (SoCG 2016).
Editors: Sándor Fekete and Anna Lubiw; Article No. 71; pp. 71:1–71:5

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

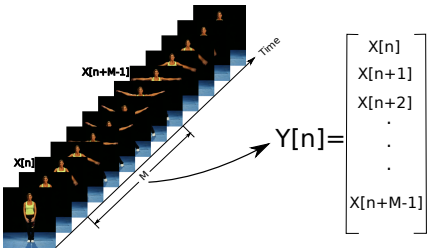


Figure 1 A depiction of a discrete delay embedding of a video of a woman doing jumping jacks with a sliding window embedding of length M.

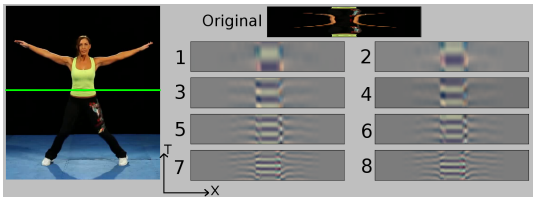


Figure 2 XT slices of the principal components of a sliding window of length 34 on the jumping jacks videos. The green line on the left image shows the X slice that is represented in the plots. Each row corresponds to the two axes of an independent ellipse in the delay embedding.

1.2 Hypertorus Video Model

We now characterize the high dimensional geometry of sliding window embeddings of periodic videos, following a similar analysis to recent on delay embeddings of 1D time series [4] (which we also summarize in our narrated video). We start by assuming very general model for periodic videos. For a period T and for constants A and ϕ, and for an arbitrary function g_i at each pixel X_i, define the grayscale level at pixel X_i as

$$X_i[n] = g_i\left(A\cos\left(\frac{2\pi}{T}n + \phi\right)\right) \tag{2}$$

That is, each pixel is an arbitrary function composed with a scale of the same cosine. Though the function at each pixel may differ, the functions across all pixels are globally *in phase*. This means that the model has mirror symmetry built in. In particular $X_i[n] = X_i\left[T - \left(n + \phi\frac{T}{\pi}\right)\right]$. That is, each pixel repeats itself during the second half of its period, but in reverse, making it impossible to disambiguate "going there" from "coming back." On the other hand, a sliding window size of appropriate length can turn this path into a topological loop by taking a different trajectory in the embedding space during the second half of the period. A similar observation was made in early work on video textures [5]. To see this mathematically, express each pixel as a discrete cosine transform with T terms, which is sufficient to summarize it over its period. Storing all T terms for all N pixels in a period in the $N \times T$ matrix D, all pixels can be combined into a column vector of the following form:

$$X[n] = \sum_{k=0}^{T-1} \cos\left(\frac{2\pi}{T}kn\right) D^k \tag{3}$$

where D^k is the k^{th} column of the matrix of DCT coefficients. A sliding window of length M then takes the following form:

$$Y[n] = \sum_{k=0}^{T-1} \begin{bmatrix} D^k\cos\left(\frac{2\pi}{T}kn\right) \\ D^k\cos\left(\frac{2\pi}{T}k(n+1)\right) \\ \vdots \\ D^k\cos\left(\frac{2\pi}{T}k(n+M-1)\right) \end{bmatrix} \tag{4}$$

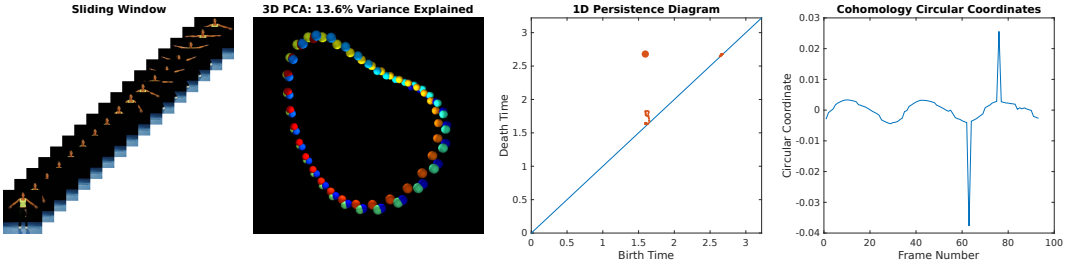


Figure 3 Sliding window embedding of a woman doing jumping jacks.

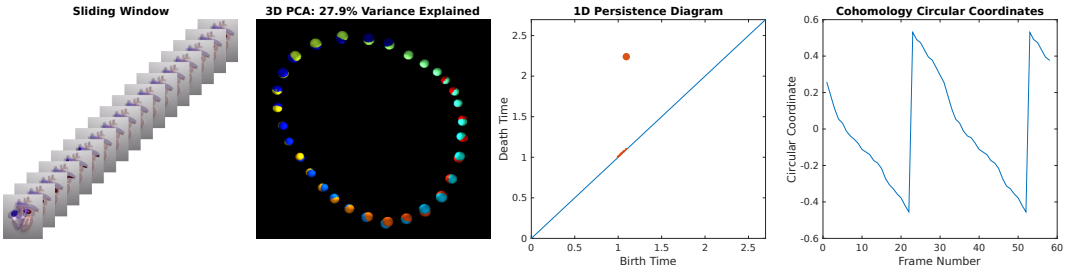


Figure 4 Sliding window embedding of a heartbeat animation.

using the cosine sum identity, this can be rewritten as

$$Y[n] = \sum_{k=0}^{T-1} \cos\left(\frac{2\pi}{T}kn\right) \begin{bmatrix} D^k \\ D^k \cos\left(\frac{2\pi k}{T}\right) \\ \vdots \\ D^k \cos\left((M-1)\frac{2\pi k}{T}\right) \end{bmatrix} + \sin\left(\frac{2\pi}{T}kn\right) \begin{bmatrix} 0^N \\ D^k \sin\left(\frac{2\pi k}{T}\right) \\ \vdots \\ D^k \sin\left((M-1)\frac{2\pi k}{T}\right) \end{bmatrix} \tag{5}$$

Hence, the path that is traced out by varying n is the sum of $d \leq T/2$ independent ellipses, each spanned by a plane. Such paths live on a topological d-torus, corresponding to $2d$ nonzero columns in D, with 2 dimensions for each independent ellipse.

Let us now empirically examine the embedding space of these motions. Let A be a matrix containing all sliding windows, with the j^{th} column of A containing $Y[j]$, and take the k eigenvectors of the matrix AA^T with the largest eigenvalues, sorted in descending order by eigenvalue. These are known as the first k *principal components* of the sliding window point cloud, and they capture the maximum variance in the data over all possible k dimensional subspaces in the embedding space. Note that vector in the embedding space in sliding window videos is itself a video with M frames. Figure 2 shows the first 8 principal component vectors in a real video, pulling out a line of pixels along the x-axis in each principal component and plotting its evolution over the M frames. Consistently with the vector part of Equation 5, lower frequency ellipse axes correspond to smooth sinusoidal motions, while higher axes correspond to higher harmonics.

2 Persistent Homology And Circular Coordinates

Since sliding window embeddings of videos lie on a highly curved topological loop on a hypertorus, high dimensional data analysis tools are necessary. We invoke 1D persistent homology to measure the geometric prominence of the loops [2]. Briefly, homology is an

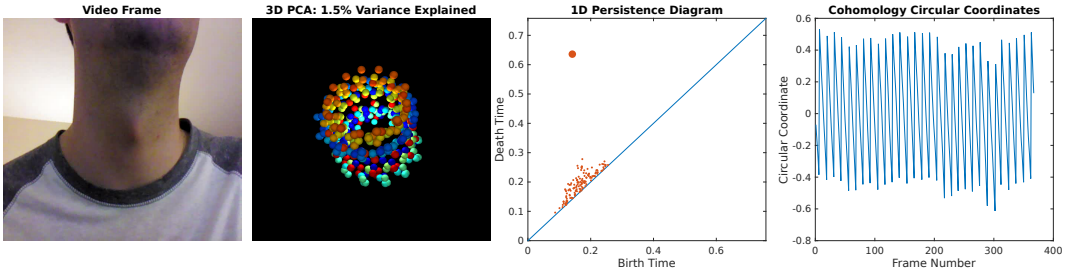


Figure 5 Sliding window embedding of a video of a person sitting still, which has hidden periodic motion due to the person's heartbeat.

algebraic framework for describing equivalence classes of loops in a fixed space, where equivalence is defined up to a boundary in that space (loops in the same class can be stretched into each other along the space without gluing or tearing). *1D Persistent homology* is an adaptation of the theory to point cloud data in which a sequence of simplicial complexes are constructed on top of the point cloud, and classes of loops are tracked as edges and triangle are added to the complex. In a *Rips filtration*, edges are added between points in ascending order of distance, and a triangle between three points is added the moment all three edges have been added. The "birth time" of a class is the distance in the Rips filtration at which the loop class first appears, and the "death time" is the distance at which it "fills in" (i.e. is expressible as a boundary of triangles). In this way, persistent homology is a mix of topological and metric information about the loop, as loops which are born earlier are more finely sampled, and loops which die later are rounder and/or geometrically larger. Birth/death pairs are plotted for all classes in a "persistence diagram," as shown in Figures 3, 4, 5. In all of these examples, there is one clear loop which has a much higher "persistence" (death - birth) than all other loops, and this loop corresponds to the periodic motion encoded geometrically with the sliding windows. We also use a related theory of 1D persistent cohomology to find maps of a point cloud to the circle for a loop class, thereby parameterizing the motion of the video embeddings with circular coordinates [1].

Figure 3 shows an example of applying 1D Rips and persistent cohomology to extract circular coordinates for the jumping jack example. The loop is visible after projecting the data onto its first three principal components, but little of the variance is explained since this video has sharp transitions from foreground to background, which need to be represented with many harmonics, as shown in our video. Circular coordinates, on the other hand, are able to parameterize the motion in high dimensions. A similar pattern is visible for a synthetic beating heart video in Figure 4. We can also apply our techniques to videos with very subtle motions. Figure 5 shows such an example with a person sitting still in front of a camera. Hardly any motion is visible, but the persistence diagram and circular coordinates indicate the presence of a cycle. In fact, this cycle corresponds to twice the person's heartbeat, which exists as a low magnitude vibration in the video. To show that there indeed is a periodic process going on, we use phase-based video motion amplification ([6]) to amplify all motions within the frequency band consistent with the parameterization found.

Acknowledgements. The author was supported by an NSF Graduate Fellowship NSF DGF 1106401.

References

1 Vin De Silva, Dmitriy Morozov, and Mikael Vejdemo-Johansson. Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, 45(4):737–759, 2011.

2 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

3 Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

4 Jose A Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15(3):799–838, 2013.

5 Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498. ACM Press/Addison-Wesley Publishing Co., 2000.

6 Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T Freeman. Phase-based video motion processing. *ACM Transactions on Graphics (TOG)*, 32(4):80, 2013.

Introduction to Persistent Homology

Matthew L. Wright

Department of Mathematics, Statistics, and Computer Science, St. Olaf College, Northfield, USA
wright5@stolaf.edu

Abstract

This video presents an introduction to persistent homology, aimed at a viewer who has mathematical aptitude but not necessarily knowledge of algebraic topology. Persistent homology is an algebraic method of discerning the topological features of complex data, which in recent years has found applications in fields such as image processing and biological systems. Using smooth animations, the video conveys the intuition behind persistent homology, while giving a taste of its key properties, applications, and mathematical underpinnings.

1998 ACM Subject Classification F.2.2 Geometrical problems and computations

Keywords and phrases Persistent Homology, Topological Data Analysis

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.72

Category Multimedia Contribution

1 Persistent Homology

In recent years, persistent homology has become a popular mathematical tool for discerning the geometric structure of complex, often high-dimensional data. Not only has persistent homology been a subject of intense theoretical interest [4, 7] and computational work [6, 11], but it has also been applied to applications in digital imaging [2, 8], biological systems [9], neuroscience [1], and more.

Persistent homology detects topological features—such as clusters, holes, and voids—of data. Generally speaking, the data is often presented as a set of points (perhaps in a high-dimensional space), values of a function at a set of sample points, or a discrete metric space. The basic methodology involves associating to the data a sequence of topological objects, usually simplicial complexes, constructed from the data for an increasing sequence of scale parameters (real numbers). Homology is computed for each object in this sequence, and topological features that persist across a range of scale parameters are identified. Thus, persistent homology is useful when one is unable to predict the scale at which features of interest will appear, or when features exist at various scales.

Persistent homology provides invariants, often called *barcodes*, that represent the persistent topological features of data. A barcode is a collection of intervals of real numbers, often drawn as parallel line segments. Yet a barcode is really a visualization of an algebraic structure called a *persistence module*, which is a graded module over the polynomial ring $k[x]$ for some field k. The structure theorem for finitely-generated modules over principal ideal domains says that such a module decomposes into a direct sum of interval modules. These intervals give the bars in the barcode.

Persistence barcodes are computable via standard algorithms involving linear algebra. The worst-case runtime of these algorithms is $O(n^3)$, where n is the number of simplices in the simplicial complex built from the data, although the computation is rarely worst-case. In


© Matthew L. Wright;
licensed under Creative Commons License CC-BY

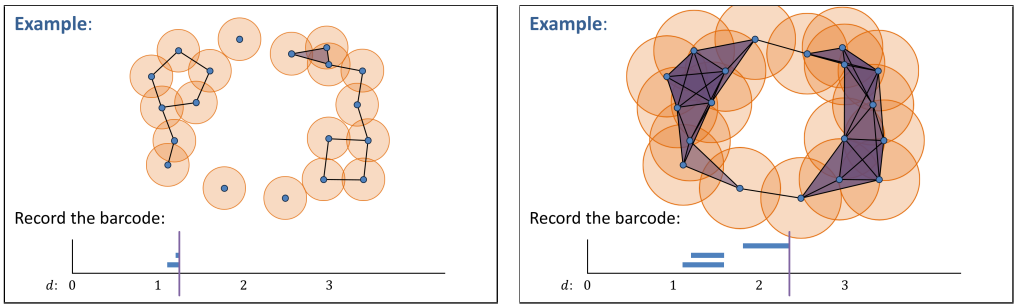


Figure 1 Two frames from the video, showing simplicial complexes constructed from point-cloud data for different scale parameters. The barcode is drawn as the scale parameter increases.

addition, clever data structures and topological simplification can speed up the computation significantly.

Persistent homology has been successfully applied to various problems. For example, Carlsson and others studied the collection of high-contrast patches, 3-by-3 pixels in size, from a large set of grayscale photographs. Treating the patches as 9-dimensional vectors and applying persistent homology, they found that the resulting point cloud was close to the surface of a Klein bottle embedded in the higher-dimensional space [2]. This observation helped create a dictionary useful for image-recognition applications [8].

For another example, Perea and others used persistent homology to detect periodicity in time-series data. Using sliding-window embeddings, the researchers converted measurements of a periodic signal into a point cloud in a high-dimensional space. Persistent homology, applied to the point cloud, detects cycles, which correspond to periodicity in the original signal. Applying this methodology to gene expression time-series data, the researchers detected periodicity that was missed by other methods—especially periodicity in the presence of damping [9].

Persistent homology has also been applied to neuroscience [1], bone morphology [10], virus evolution [3], and sensor networks [5], and more.

2 The Video

Given the increasing interest in persistent homology among mathematicians, scientists, and others, it is desirable to have resources to help teach this subject to those who are new to topological data analysis. The present video is designed to do just that: to make persistent homology accessible to a wide audience. In its first eight months on YouTube, the video has been viewed more than 2500 times by viewers around the globe.

While the mathematics of algebraic topology are formidable, the intuition behind persistent homology is easily conveyed in video format. The author has found that animations depicting the evolution of a simplicial complex and the corresponding barcode as the scale parameter increases, are an effective way introducing persistent homology to non-specialists. Such an animation is found in the present video, beginning at about 4 minutes and 40 seconds after the start of the video; two frames of the animation appear in Figure 1. The author has used this animation, as well as other clips from the video, to teach persistent homology to students, colleagues, and friends.

The video is designed to be accessible and intuitive. It is not the author's intent to minimize the rich mathematical theory of homology, nor to hide the computational challenges

involved in using persistent homology in real applications, as the video points to the theory and challenges that one can appreciate with further study. Though the video does not dwell on technical details, but merely skims the surface of persistent homology, the author has attempted to convey that great and fascinating depths exist under that surface. It is the author's hope that this video will motivate viewers to embark on further study of mathematical theory, computation, and applications—or at least will lead them to a greater appreciation of the elegance and utility of mathematics.

The animations seen in the video were prepared using the animation tools in Microsoft PowerPoint 2013. The audio was captured with a Blue Snowball microphone, and the in-person video was recorded with a Canon Digital Rebel T3i camera. Screen capture and editing were performed using ActivePresenter by Atomi Systems.

Acknowledgements. This video was prepared while the author was a Postdoctoral Fellow at the Institute for Mathematics and its Applications (IMA) at the University of Minnesota, supported by funding from the National Science Foundation. This submission was prepared while the author was a Visiting Assistant Professor at St. Olaf College. The author is partially supported by NSF DMS-1606967. The author thanks the IMA, St. Olaf College, and the NSF for their support of mathematics research and education.

References

1. P. Bendich, J.S. Marron, E. Miller, A. Pieloch, and S. Skwerer. Persistent homology analysis of brain artery trees. *Annals of Applied Statistics*, to appear.
2. G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008.
3. J.M. Chan, G. Carlsson, and R. Rabadan. Topology of viral evolution. *Proceedings of the National Academy of Sciences*, 110(46), November 2013.
4. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete and Computational Geometry*, 37(1):103–120, 2007.
5. V. de Silva and R. Ghrist. Coverage in sensor networks via persistent homology. *Algebraic and Geometric Topology*, 7:339–358, 2007.
6. H. Edelsbrunner and J. Harer. Persistent homology: a survey. In *Surveys on discrete and computational geometry: twenty years later: AMS-IMS-SIAM Joint Summer Research Conference, June 18-22, 2006, Snowbird, Utah*, volume 453, page 257. American Mathematical Society, 2008.
7. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete and Computational Geometry*, 28(4):511–533, 2002.
8. J. Perea and G. Carlsson. A klein-bottle-based dictionary for texture representation. *International Journal of Computer Vision*, 107(1):75–97, 2014.
9. J. Perea, A. Deckard, S.B. Haase, and J. Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16(1), 2015.
10. K. Turner, S. Mukherjee, and Doug M Boyer. Sufficient statistics for shapes and surfaces. *preprint*, 2013.
11. A. Zomorodian and G. Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33(2):249–274, 2005.